Front-End: React.js

TEST PROJECT GUIDELINES

Introduction

Welcome to TEAM International a software development innovative company us owned and managed. We have operations across the world with delivery centers in the Ukraine, Poland and Colombia. Being a part of TEAM International gives you the chance to work on challenging projects with great professionals, international clients, and the latest technologies. We have a great benefits package that includes: English lessons, Prepaid Medicine, Gym Bonus, and Educational allowance. You will join a great working environment, with chill spaces (hey we got a play station!), you can enjoy our beer Fridays and free snacks at the offices, and we are in an amazing location at the Poblado's Milla de Oro Building.

With this project we will be able to see what would full engagement of you at Team International look and feel like. We expect to receive this project within a week time after you receive it, to prove that you are the right fit with our team and willing to fight hard for the job.

Thanks very much for investing your time in our recruitment process!

Instructions

Create a simple **React.js** app and commit the code to a repository in Github and send back the repo's link to your recruiter. The app must compile and run by using the standard *start* npm script.

- Create the app by using create-react-app
- Use React-MD and configure SASS by using the following link

Good to have

- Components and code organization
- Software patterns and principles
- Presentational and Container components
- A CSS Methodology, such as: BEM (preferable) or SMACSS
- Front-End best practices (Clean Code ones apply as well)
- Redux
- Exact versioning
- Proptyping
- Semantic HTML
- Functional programming
- ESNext

- Linters

Optional
- Any sort of tests
- CSS Modules
- editorconfig

Specification

You are to build a SPA which displays a list of a company's products which are categorized.

You are in charge of creating the skeleton of the application and set the foundations for further development, which means you'll have to create the app from scratch and define the libraries to be used. On top of that, you must create the Products page as well, as an example to follow throughout the entire application.

The app must be linkable, meaning if I want to send the products list filtered by category to a friend or the boss, I must be able to do it. For instance, if a want to send the link of Technological products, there must be a relative URL like: `/products/tech`.

General specs:
- Each page must be as pixel perfect as possible.
- Each rendered page and every component that performs an api call must display a loading icon, as indication that some process is happening behind the scenes.
- API calls, to fetch the Products and to send the Contact form data, must be performed.

Products Page

The contact page should be fully linkable as: `/products`.
The products page must contain the following

Categories panel (see mock below):

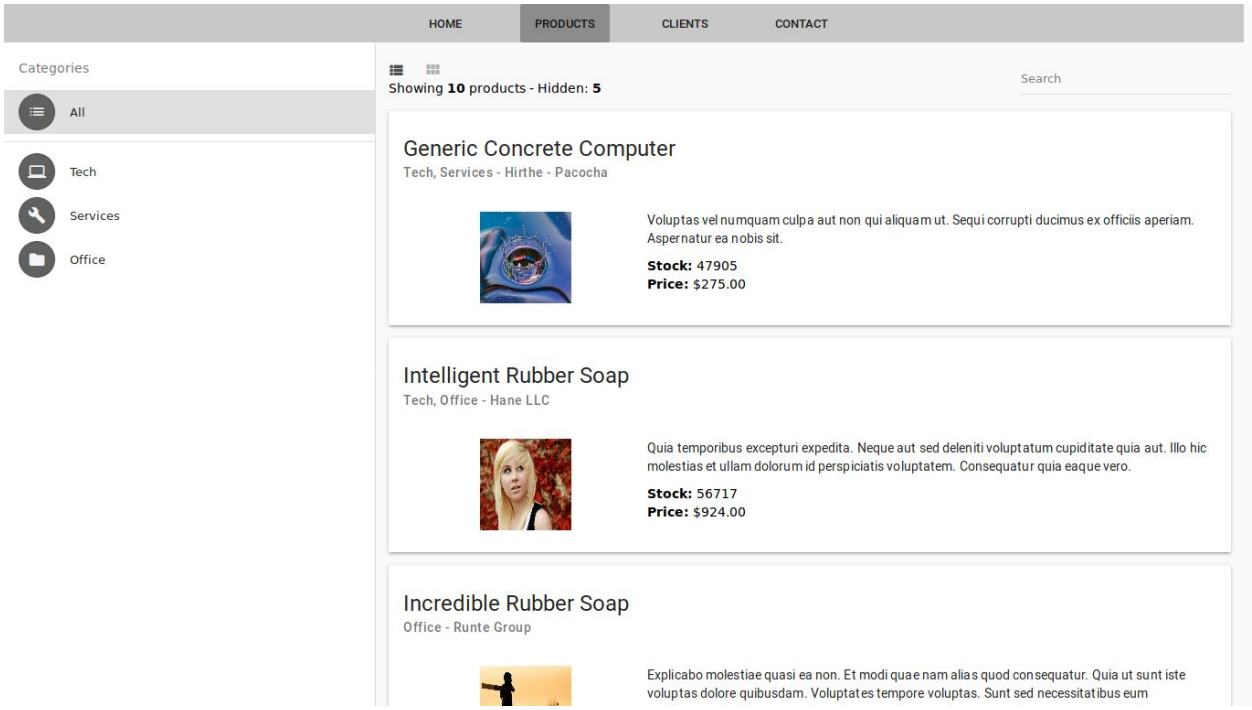
- Contain the 3 categories: **Tech**, **Services**, and **Office**.
- Contain an **All** category which will list the entirety of the products (no pagination need).
- Whenever a category is clicked it must
    - Appear as active (only one at the time)
    - Change the URL (as following) accordingly to the category's name
        - Tech: `/products/tech`

        - Services: `/products/services`
        - Office: `/products/office`
        - All: /products
    - Filter the products list by category. Keep in mind that a single product can have multiple categories and it must be listed in all of its categories
- Must be fixed in the left side, meaning, it will not be scrollable nor will be scrolled with the products list.

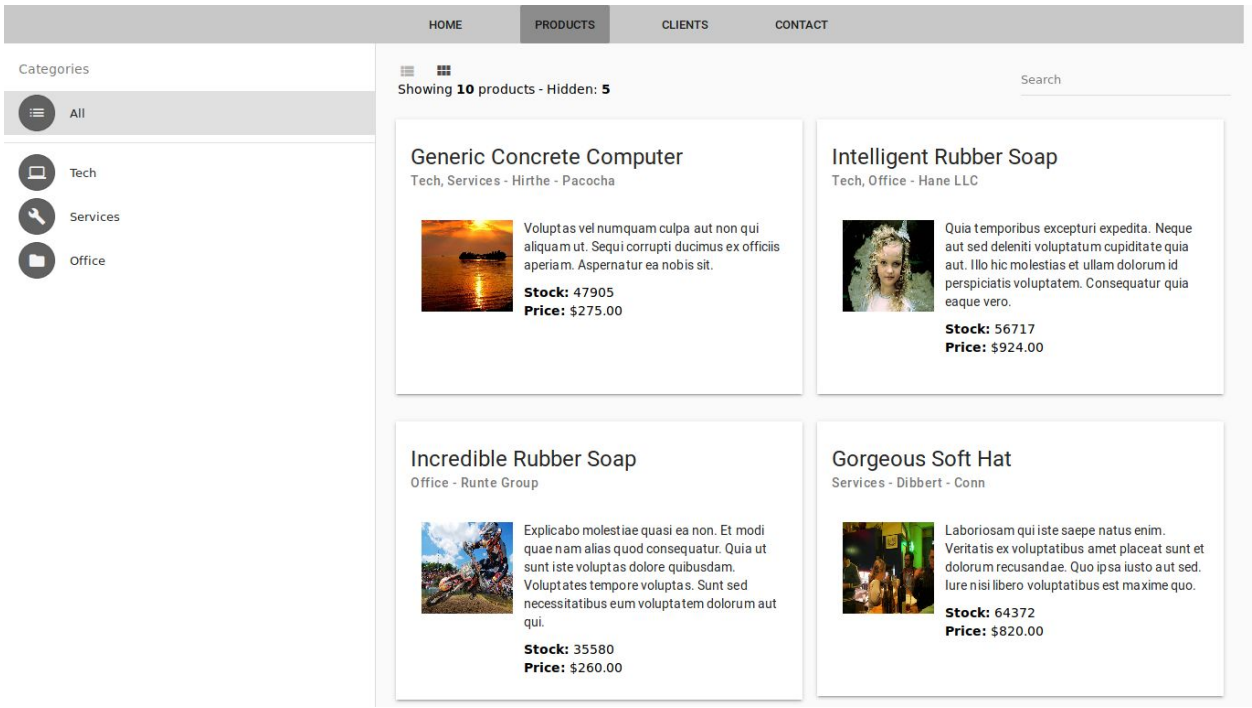
Products List:

- Display how many products are being displayed
    - Total number for **All** category, partial for the other ones
- Display how many products are being hidden from view if a category is selected (Doesn't apply for **All**)
- Each product contains the following data (as defined in the API's data set):
    - `id`: Product's identifier
    - `name`: Product's name
    - `brand`: Brand who manufactured the product
    - `categories`: Product's categories
    - `description`: Product's description
    - `photo`: Image of the product
    - `price`: Product's price
    - `stock`: Number of items in stock
- Toggle views between list and blocks
    - For the blocks view: Display 2 or more per row, as many as they can fit responsively in the horizontal visible space. Max height is 300px
- [Plus] Filter/Search the products
    - Products must be filtered as each word is typed, in client side.
    - When the search filter is deleted/removed, the list must return to the initial unfiltered state.
- [Plus] Infinite scroll. Ten products per render iteration.

List view:



Blocks view:

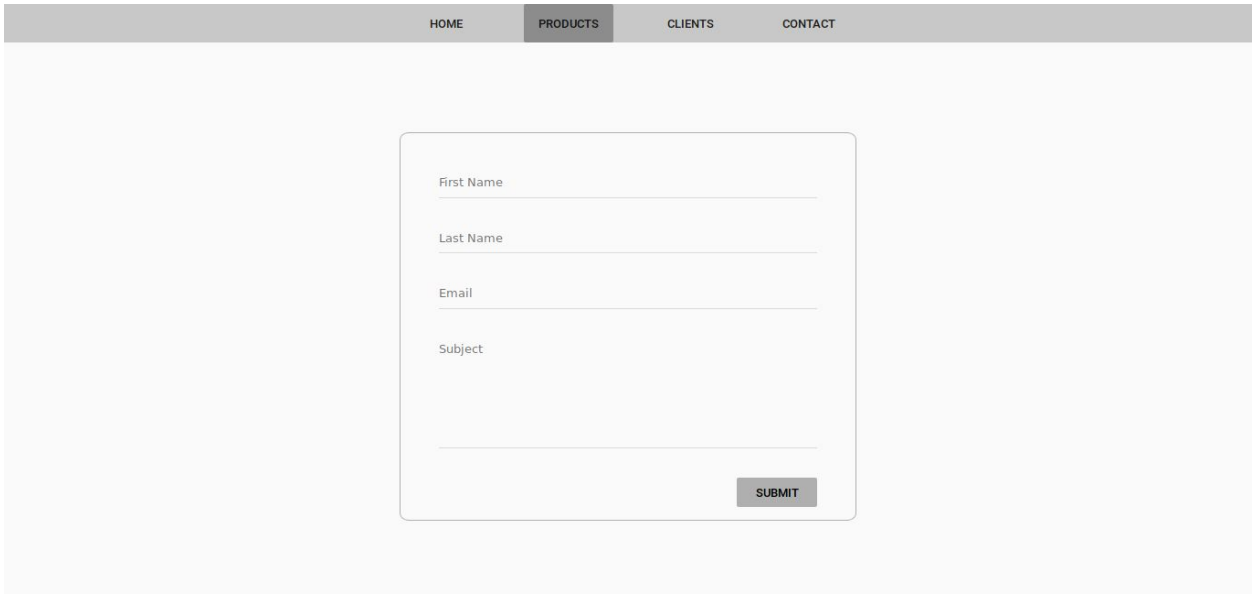


Contact Page

The contact page should be fully linkable as: `/contacts`.

The page must contain the following:

- **First Name** text field:
  - Placeholder: Your name...
  - Maximum length: 255
  - Required
  - Error message: Please write down your first name
- **Last Name** text field:
  - Placeholder: Your last name...
  - Maximum length: 255
  - Required
  - Error message: Please write down your last name
- **Email** text field:
  - Placeholder: Your email address...
  - Maximum length: 255
  - Required
  - Error message: Don't forget to tell us what your email address is
- **Subject** text field:
  - Placeholder: Let us know your concerns!
  - Maximum length: 500
  - Required
  - Error message: Don't forget to write something to use!
- The button submit must be disabled by default and must be enabled only when all fields are entered
- The form must be placed in the middle of the screen, regardless of the resolution
- Error messages must be placed below the text field and change the field's color accordingly (see mock below)



Home & Clients Pages

Each of the the remaining pages must have an `h1` with the name of the page (Home, Clients, Contact) as content, and each page must be fully linkable:

- Home: `/`
- Clients: `/clients`

API

You are to build a small server in Express with NodeJS in which you will serve the content of a JSON file provided in this link for the products list and create a new endpoint that will receive the data from the Contact form and print the data out in the console. Architect the Contact form endpoint as you see fit. This small server must be provided inside your project as a mini project, in its own folder and with a package.json file. The server must be started either by using the `start` npm script or `node index.js` in the console within the mentioned folder.