Challenge

Create an Address book app.

Create a fictional address book app, where you can search for users' addresses and personal information. The app displays a list of users which you can browse and where you can see personal information for a selected user. There is also a settings page where you can select which nationalities you are interested in.

Minimum requirements 📝

- Display a list of users
- Search field
- Settings page to filter nationalities of the user (CH, ES, FR, GB)
- Unit testing

Show off additional skills 🚀

Extend the application. Take these examples or come up with your own idea!

- Pre-fetch batch of users
- Detail modal with the user's info
- Save nationalities set in the settings page so they'll persist between refreshes

What we value:

- Clear commit history
- Tooling (linting, code formatter…)
- CSS preprocessor usage
- Clear code (easy to read, good variables names)
- Webpack
- Folder structure
- Accessibility and semantic markup

How is the task graded?

We are looking for idiomatic use of JavaScript/TypeScript, and the ability to solve the problems with code that is clean and easy to read. Even though it's very small in scope, please show us how you would use the language and conventions to structure things in a clear and maintainable way.

Documentation is also important. Provide steps to run the project locally that are easily done and require no possible debugging.

Users API
- please read the documentation on the https://randomuser.me, and get users data from there.

Home page (browse with search)
- the page should be located on the root url `/`
- display the users in a grid.
- each user has:
  - a `picture.thumbnail` field,
  - a `name.first` field,
  - a `name.last` field,
  - a `name.username` field,
  - an `email` field,
- the user's grid should automatically load more users as you scroll down, in batches of 50 users, the max length of the catalogue is 1000.
- when the request to load more users (next batch of 50) is done, display an animated "loading..." message while the request is taking place.
- when the end user reaches the end and there are no more users to display, show the message "end of users catalog".
- **extra: (idea to extend the application)** to improve the end user's experience, we recommend always preemptively fetch the next batch of users in advance, making use of idle time. This does not mean that we should show all the users right away, only once the end user has scrolled down the 50 batch of users.

Search
- display a user search on top of the app.
- search field should be case insensitive and should filter the results by `name.first + name.last`.
- when the end user scrolls down, the search should follow the screen so that it is always visible.
- search should filter all visible users and show only those which match the search string.

Remark: Since randomuser.me does not support filtering by user name, it's increasingly hard to combine filtering and infinite list features together on the front-end. We suggest to pause the loading mechanism while the filter is active and indicate that fact somewhere on the screen.

Settings page
- create a different URL named `/settings`, where we will have our settings page.
- here the end user can set, from which nationalities users are fetched for browsing/searching.
  Possible choices should be: CH, ES, FR, GB.
- there should be a button or a link so that the end user can access the settings page and also go back to the search/browse page.
- settings should affect the home page and fetch the users based on selected nationality.
- when setting a new nationality, it should not cause the reloading of the page.

Unit tests
You should cover your whole application with unit tests.

Extra: Details modal (idea to extend the application)
- when the end user clicks on the specific user in the row (can also be a button or an icon in a row), a modal with additional info should open.
- each detail modal should have:
  - a ‘`location.street`’ field,
  - a ‘`location.city`’ field,
  - a ‘`location.state`’ field,
  - a ‘`location.postcode`’ field,
  - a ‘`phone`’ field,
  - a ‘`cell`’ field,
- the modal should be closable so that the end user can browse/search on.