**Task 3**

I recommend use TypeScript for React. This makes development easier if several people are working on the project at the same time.

Tools:

* **Docker** is tool designed to make it easier to deploy and run application. Docker allow a developer to package up an application with all necessary libraries and with other dependencies.
* **Postman** is an API development tool which helps to build, test and modify APIs. It has the ability to make HTTP requests.

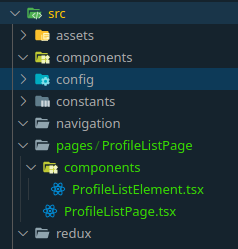
**Framework:**

* **React-bootstrap or material ui**: Simplifies the responsive and user-friendly ui development.

**NPM libs:**

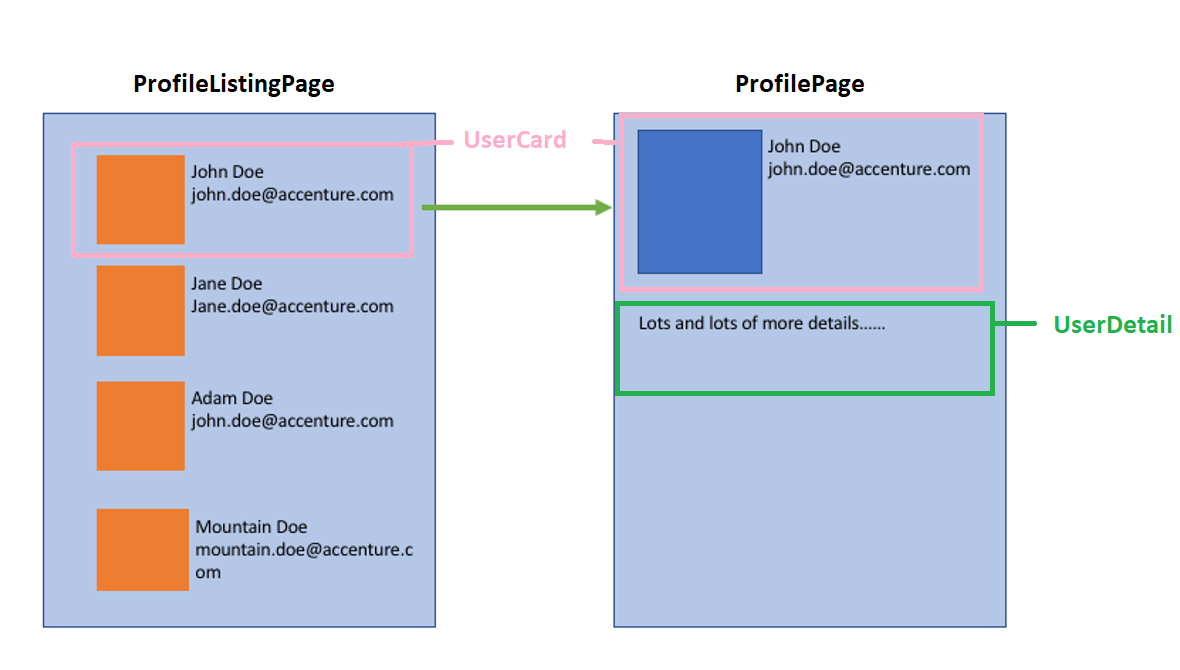
* **Styled-component:** It allows to write css code in your components. It help you write more readable code.
* **Redux or other state management solution:** With redux, the state of your application is kept in a store and each component can access any state that it needs from the store.
* **Axios:** Axios is a lightweight HTTP client which provides better error handling.

**Directory structure**

The components folder is the place of the shared components.

The redux folder holds all the redux resources at one place.

**Components**



The **ProfileListingPage** is component which contain all of the components which belong to the ProfileList.  
The **UserCard** is reusable component. We will use it several times within ProfileListPage with different props. In the ProfilePage we will use it with other style.

The **UserDetail** will contain the more information about the user.

**SPA vs not SPA**

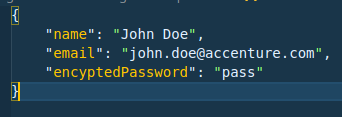
The biggest benefits of the single page apllication is speed load. SPAs are much faster than the regular sites because they are able to load new information into a single page instead of having to link to several other HTML pages. Other advantage is the caching of the SPA applications. SPAs only have to request data from the server one time, upon initial download. Then the site can be accessed even if there is a poor internet connection or if the user is offline.

***Backend***

API uri

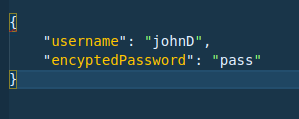
*post: /*user*/registration/*

*User can register.* The frontend will send post request with the following format.



*post: /*user*/login/*

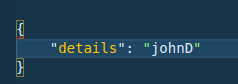
After registration users can login their account.



*get: /*user*/getUsers/*

The frontend will get the registered user list.

  
  
  
  
get: */*user*/getUserDetail/:id*

The frontend will get the details of the user which based on the id.  
  
  
  
  
  
  
**Framework**: I recommend Nestjs framework for the development.