# React, State, Fetch and Mobx

## General part

* Describe the term Single Page Application and why it has become so popular for modern web-applications
* Explain the “recommended” React way of passing data into Components (at the top, or at the bottom or..)
* Explain how JavaScript array methods, like filter, map and (reduce) can be used to generate dynamic HTML structures (tables, ul's etc.)
* Explain about the Observer pattern, and where you have used it, both with Java, JavaScript and Mobx.
* Explain how a library/framework like Mobx fits into the React world, and the actual problems it solves.

## Practical part

**Getting Started:** Clone the start code (exactly as) outlined below to get the right branch (it will provide a React project with support for ES-7 decorators for the MobX part of the exercise):

git clone **-b ejected** https://github.com/Lars-m/reactMoxExExamex3.git

* *Type* ***npm install*** *to fetch dependencies and* ***npm start*** *to run,*  and open the project in your favourite IDE.
* In the *root* of the project, open a new terminal and type **npm run dataserver**. This will start a simple REST/JSON server, which you need for this exercise (leave this window open, and let the server run for the rest of the exercise). Test it via a browser: <http://localhost:4567/api/persons>

The project is *a create-react-app* generated project, modified to provide start code for this exercise. It simulates a site that can show “members” their friends, and eventually members will change in the backend “all the time”.

**1**) Complete the personFactory to fetch data, using fetch, from the REST-api given above.

**2)** Use your updated personFactory and “inject” it into relevant controls so it will be available via props in PersonTable Control

**3**) Add the necessary code to PersonTable to render a table as sketched here.

**4)**

Change the code that generates the Table to provide a comma-separated list of all friends, instead of just the number of friends

**5)**

Change personFactory to use mobX for event-Handling (it must be possible to observe the persons array).

Add the necessary code to other Components (where required) to observe the persons array.

**6)**

Change the URL in personFactory into this one <http://localhost:4567/api/persons_changing>/ , and use JavaScript’s setInterval function to repeatedly update the person's array with fresh data from the server (while developing, fetch every 3 seconds to see changes).

*This new URL provides new (random) data “all the time” to simulate new members are added, members leave, etc*.

How this would be graded:

|  |  |  |
| --- | --- | --- |
|  | **2-4** | **To fall into this range you must:**  Give a minimal to fair performance related to the topics stated in the "General Part"  And  Have completed, ex 1+2 in the practical part with only a few weaknesses |
|  | **4-7** | **To fall into this range you must:**  Give a fair to good performance related to the topics stated in the "General Part"  And  Have completed, ex 1+2 in the practical part with none or only minor weaknesses  Have completed ex-with only minor weaknesses or alternatively 3+4 with some major/minor weaknesses. |
|  |
|  | **7-10** | **To fall into this range you must:**  Give a very good performance related to the topics stated in the "General Part"  And  Have completed, ex-1 +2 in the practical part with none or only a few minor weaknesses  Have completed , ex 3 +4 in the practical part with only minor weaknesses  Have completed 5 with only minor weaknesses or  Have completed 5+6 with some major/minor weaknesses |
|  |
|  | **10-12** | **To fall into this range you must:**  Give an excellent performance related to the topics stated in the "General Part"  And  Have completed, ex-1 +2 in the practical part with no or only a few minor weaknesses  Have completed , all steps in ex-3 +4 in the practical part with only a few minor weaknesses  Have completed 5+6 with only minor weaknesses  Have a working demo, demonstrating all topics above (--> 12) |