Capabilities Test

This test is designed to understand your coding style, your approach to solving problems and your knowledge of React and Flux. You are supposed to write a self containing application that solves the problem below. APIs for data fetching are provided in the sample below along with the instructions to run them. There is no need to develop any server side code. You will only be focussing on React and Flux and using the APIs provided to you below.

Deliverables

You are required to deliver a .zip folder with pre-compiled scripts in the following structure. No server side code (e.g. ASPNET, Ruby, Node etc) is required. **Also, in order to view your solution, we will not compile any of your source code.**

- /dist
  - index.html (this html file will be opened to see your solution in action)
  - app.js (this contains precompiled react and flux solution that is referenced in the index.html)
- /source (your source code should be here)

Scenario

The hypothetical application that you are going to develop is meant for Admin Staff to enter new Students along with their basic information (Name, Nationality, Date of Birth) as well family information (like parents and siblings). Once an Admin Staff submits the application, he/she cannot edit the information and all fields should appear as disabled for him/her.

This information then needs to be approved by the Registrar. The Registrar can edit any information of a student along with adding/editing or even deleting family members.

Both Admins and Registrars see a table where they can see registered students and can click to view their details in a Modal.

There is no login into the system, the role of the user can be changed via a drop down from the top menu on the landing page.

Screens

Landing Page

The landing page contains a table that lists all students. The rows can be clicked and opens the Modal with Student Details (see below). This screen also contains a *Add New Student* Button that will be used to create a New Student. It also contains a dropdown on the top of the page that allows the user to switch roles

Student Modal

This is a Modal that will contain 2 Sections.
Section 1 is Basic Information that contains the First Name, Last Name, Date of Birth (Date Picker) and Nationality (drop down).
Section 2 is Family Information that will contain a section for each family member. Each Family Member Section contains the following fields: Name, Relationship (DropDown with 3 choices Parent, Sibling, Spouse), Nationality (DropDown). This section also contains the *Add Family Member* button which is used to add new Family Members (the user can add as many family members he wishes), as well as a *Delete Family Member* used to delete each Family Member. Also, all fields of the Family Member can be edited by the Registrar.

All Fields in the Modal should be disabled, when the Admin opens the students that are already submitted.

All Changes should be submitted only when the user taps submit and should be discarded when the user taps cancel.

Endpoints

The following endpoints are already available to you in order to fetch and store data. In order to run the endpoint, download the project provided below, and run 'npm install' and then 'node source/app.js'.

> *You are not allowed to modify the server endpoints in any way.*

Server Source Code (Download below)

https://www.dropbox.com/s/mfg5mvt13vkzwtz/interview-server.zip?dl=0

Students

```
http://localhost:8088/api/Students (GET, POST)
```

Available Methods:
GET: Get all Students
Post: Add a new Student with Basic Details Only

```
http://localhost:8088/api/Students/{id} (PUT)
```

Available Methods:
PUT: Updates a Student's Basic Details only

```
http://localhost:8088/api/Students/{id}/Nationality/{id}
```

Available Methods:
GET: Gets the Nationality of a particular student
Put: Updates a Student's Nationality

```
http://localhost:8088/api/Students/{id}/FamilyMembers/
```

Available Methods:
GET: Gets Family Members for a particular Student
POST: Creates a new Family Member for a particular Student **(without the nationality)**

FamilyMembers

```
http://localhost:8088/api/FamilyMembers/{id}
```

Available Methods:
PUT: Updates a particular Family Member
DELETE: Deletes a family member for a particular Student

```
http://localhost:8088/api/FamilyMembers/{id}/Nationality/{id}
```

Available Methods:

GET: Gets a nationality associated with a family member
PUT: Updates a particular Family Member's Nationality

Nationality

```
http://localhost:8088/api/Nationalities (GET)
```

GET: Gets all nationalities in the system