In this event, our attendee number is quite low. We have just **10** guests. Your responsibility is creating new guests by randomly mixing the guest name by their professions. You should increase this list to **100**.

Guests lists can be fetched at: https://jsonplaceholder.typicode.com/users as JSON.


```
[
    {
            "id": 1,
            "name": "Leanne Graham",
            "username": "Bret",
            "email": "Sincere@april.biz",
            "address": {
            "street": "Kulas Light"
            "suite": "Apt. 556",
            "city": "Gwenborough",
            "zipcode": "92998-3874",
            "geo": {
                  "lat": "-37.3159",
                  "lng": "81.1496"
                  }
            },
            "phone": "1-770-736-8031 x56442",
            "website": "hildegard.org",
            "company": {
            "name": "Romaguera-Crona",
            "catchPhrase": "Multi-layered client-server neural-net",
            "bs": "harness real-time e-markets"
            }
      },
      …
]
```

“name” => is for the **name** field

“catchPhrase” => is for the **profession** field

E.g: First guest is “**Leanne Graham**” and her profession is “**Multi-layered client-server neural-net**”.

The second guest is “**Ervin Howell**” profession “**Proactive didactic contingency**”. We have eight more guests in this list.

The randomly generated guest would be “**Leanne Howell**” and profession “**Multi-layered contingency neural-net**”.

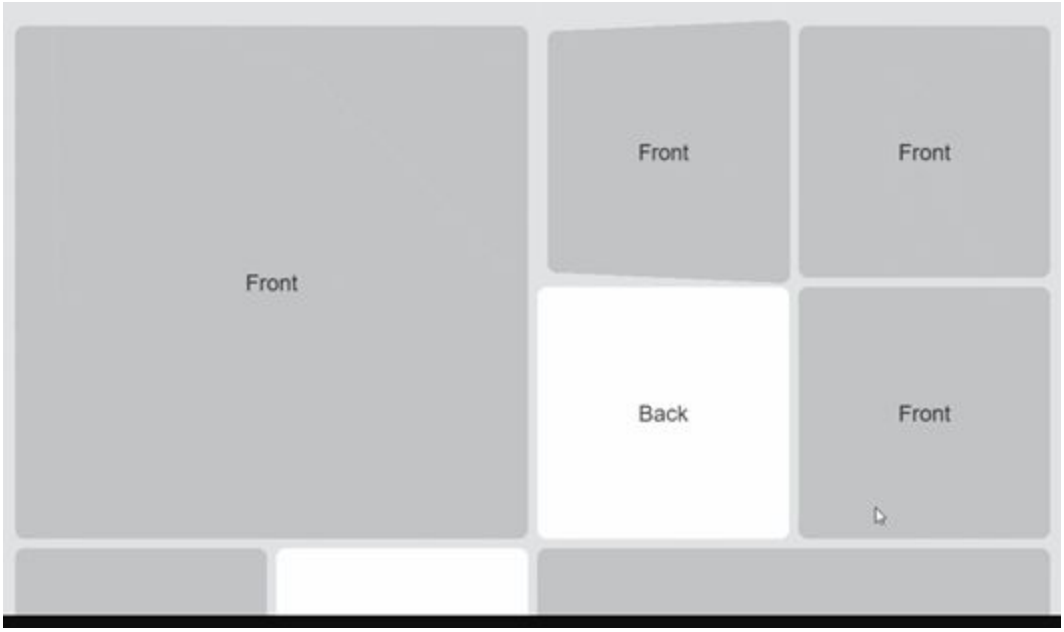
As you can see, we randomly picked words from guest names and words from their profession.

The newly generated guest **names** should be **equal to 2 words** and their **profession** word count should be **equal to 3 words.** (Eg: “Ervin(1) Howell(2) Leanne(3)” is wrong, “Multi-layered(1) client-server(2)” is wrong). Special characters are counting as a part of word too. Only the blank space is a separator of words.

After successfully generating the 90 more guests (10 who already is attending and 90 more to complete the desired 100 guests). All the **guests** should be **displayed** in a **flip card**. **Front side** of the card should display the **name** of the guest and the **back side** of the flip card should display the **profession** of the guests. Click for the flip card example.

The **original guest** who are incoming from the beginning should be **displayed** in a **bigger card** (min 2x bigger). Cards should be **sorted** by **alphabetically**. **To flip** the cards, we should **click on the card**. Also, we should have a **button** in the corner of flipboxes which allows to **delete the card** from the guest grid. By clicking the delete button should be displayed a **modal** which asks the user if he/she is **sure** about the **deletion**. It will prevent the unwanted/wrongly deletions.

Grid layout:



Above of the cards additionally we should **have a checkbox list** which allows selection of **forbidden words** for mixing and randomly generating a new guest. Selected names must not be included in randomly generated new guest names.

Forbidden checkbox list:

Note: Don't write the names statically. All names should be fetched and mapped from the Web API.

The application **initially** displays the **10 real guests** in the flip boxes. Then by selecting the forbidden names there should be a **button** which **fulfils the list to 100 guests**. By randomizing the names we are exposing a possible edge case which in the guest list we may have a DUPLICATE guest. It's a quite low chance to have it but we should take care of that situation too.

After generating and displaying the guests, **above the guest grid** add a field which displays the **duplicate count**. If there is a duplicate render a new button with name DEDUPLICATE and **functionality** on click to **delete the duplicate flipboxes** from the grid.

- Create a **React app** which visually displays the grid of the guests in a flip cards.
- Get the guest data from the link which is provided above.
- Use ES6 syntax including arrow functions, destructing and async/await
- Separate the logic in pieces by following the best practices.
- Add types with Flow or Typescript
- Keep the code clean and avoid repetition
- Add a spinner or skeleton view when information is loading
- Deal with errors coming from the backend
- Have a nice UI using a components library (Bootstrap, Ant Design, Material UI or another?)
- Show common tools used for you daily development environment
- Add unit test to test the critical parts of the application (ex: user generation functionality) **(optional but BIG plus)**
- Add README file or any text file which describes how project should be started.
- Deploy (Publish) the project on any free PaaS (Netlify, Heroku, Github Pages) **(optional but BIG plus)**