**App Functionality**

**Login Page**

The person using your application should have a way of impersonating/logging in as an existing user. (This could be as simple as having a login box that appears at the root of the application that lets the user select a name from the list of existing users. Alternatively, you could create your own account creation process to allow a user to sign up for an account.) Your application should work correctly regardless of which user is selected. Once the user logs in, the home page should be shown.

We always want to make sure we know who the logged in user is, so information about the logged in user should appear on the page. If someone tries to navigate anywhere by entering the address in the address bar, the user is asked to sign in and then the requested page is shown. The application allows the user to log out and log back in.

**Home page**

Once the user logs in, the user should be able to toggle between his/her answered and unanswered polls on the home page, which is located at the root. The polls in both categories are arranged from the most recently created (top) to the least recently created (bottom). The unanswered questions should be shown by default, and the name of the logged in user should be visible on the page.

**Single Q page**

What would be the point of seeing answered and unanswered polling questions if we couldn’t actually vote or see the results? Each polling question should link to the details of that poll. The details of each poll should be available at questions/:question\_id.

When a poll is clicked on the home page, the following is shown:

1. Text “Would You Rather”;
2. Avatar of the user who posted the polling question; and
3. Two options.

For answered polls, each of the two options contains the following:

1. Text of the option;
2. Number of people who voted for that option; and
3. Percentage of people who voted for that option.

The option selected by the logged-in user should be clearly marked.

Since we want to make sure our application creates a good user experience, the application should show a **404 page** if the user is trying to access a poll that does not exist. (Please keep in mind that newly created polls will not be accessible at their url because of the way the backend is set up in this application.) It should also display a navigation bar so that the user can easily navigate anywhere in the application.

So what happens when someone votes in a poll? Upon voting in a poll, all of the information of an answered poll should be displayed. The user’s response should be recorded and clearly visible on the poll details page. Users can only vote once per poll; they shouldn’t be allowed to change their answer after they’ve voted -- no cheating allowed! When the user comes back to the home page, the polling question should appear in the “Answered” column.

**New Question Page**

It would be no fun to vote in polls if we couldn’t post our own questions! The form for posting new polling questions should be available at the /add route. The application should show the text “Would You Rather” and have a form for creating two options. Upon submitting the form, a new poll should be created, the user should be taken to the home page, and the new polling question should appear in the correct category on the home page.

**Leaderboard Page**

But how can we know how many questions each user has asked and answered? Let’s get some healthy competition going here! The application should have a leaderboard that’s available at the /leaderboard route. Each entry on the leaderboard should contain the following:

1. User’s name;
2. User’s picture;
3. Number of questions the user asked; and
4. Number of questions the user answered

Users should be ordered in descending order based on the sum of the number of questions they’ve asked and the number of questions they’ve answered. The more questions you ask and answer, the higher up you move.

The user should be able to navigate to the leaderboard, to a specific question, and to the form that allows the user to create a new poll *both* from within the app and by typing in the address into the address bar. To make sure we’re showing the data that is relevant to *the user*, the application should require the user to be signed in order to access those pages.

## **A Guide for the Planning Stages of Your Project**

1. Identify What Each View Should Look Like
2. Break Each View into a Hierarchy of Components
3. Determine What Events Happen in the App
4. Determine What Data Lives in the Store

**Identify What Each View Should Look Like**

**Home View Requirements (View-1)**

* + Is located at the home route (/)
  + Show user’s **answered and unanswered polls** and toggle between them
  + Show these polls in both categories **sorted from most recently** added at the top, to oldest at the bottom
  + The **unanswered questions** should be shown by **default**
  + Show the name of the **logged in user** visible on the page
  + Each poll will show:
    1. Text “Would You Rather”
    2. **Name and picture of the user who posted** the polling question
    3. **Some text of option**
    4. **View Poll button** to redirect to **either Detail of Poll page or Poll Result Page**

**Single Poll View Requirement (**Details of each poll) **(View-2)**

* + Is located at (/questions/:question\_id)
  + For unanswered polls, each poll will show:
    1. Text “Would You Rather”
    2. **Name and picture of the user who posted** the polling question
    3. **Some text of option**
    4. **Submit button** to redirect to Poll result

**Poll Result (For Answered Polls)**

* + **Name and Avatar of the user who posted** the polling question
  + Each of two options contains the following:
    1. **Text of the option**
    2. **Number of people who voted for that option**; and
    3. **Percentage of people who voted for that option**.
    4. The option selected by the logged-in user should be clearly marked.

**New Poll View Requirements (View-3)**

* + Is located at (/add)
  + show the text “Would You Rather”
  + have a **form for creating two options**.
  + Upon submitting the form, a new poll should be created, the user should be taken to the home page, and the new polling question should appear in the correct category on the home page.

**Leaderboard View Requirement (View-4)**

* + Is located at (/leaderboard)
  + Each entry on the leaderboard should contain the following:
    1. **User’s name**
    2. **User’s picture**
    3. **Number of questions the user asked**
    4. **Number of questions the user answered**

\*\*Users should be ordered in descending order based on the sum of the number of questions they’ve asked and the number of questions they’ve answered. The more questions you ask and answer, the higher up you move.

**Login Requirements**

* + Show **dropdown for user** to select a name from the list of existing users
  + Show **Sign In button**
  + Once the user logs in, the home page should be shown
  + allows the user to log out and log back in.

\*\*If someone tries to navigate anywhere by entering the address in the address bar, the user is asked to sign in and then the requested page is shown. The application should require the user to be signed in order to access those pages.

**404 Requirement**

* + show a **404 page** if the user is trying to access a poll that does not exist.
  + (Please keep in mind that newly created polls will not be accessible at their url because of the way the backend is set up in this application.) It should also display a navigation bar so that the user can easily navigate anywhere in the application.

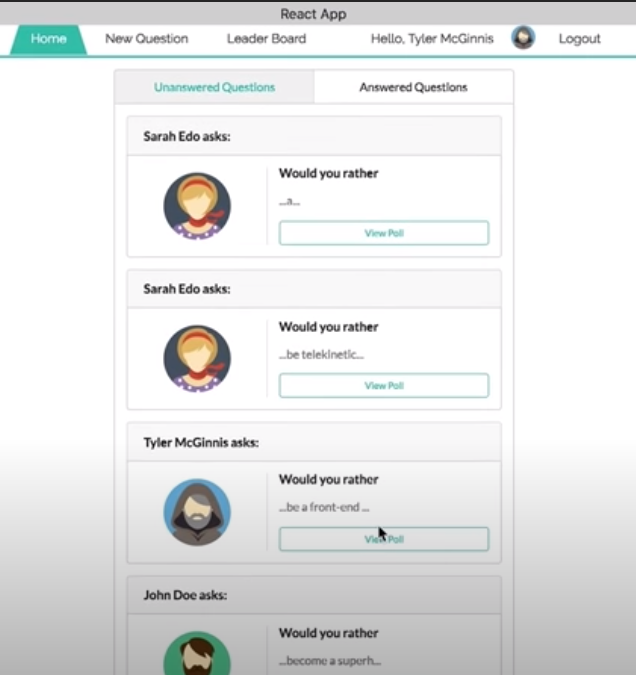
**View Recap**

So, these are the 4 views we need in our app:

* **Home**
* **Single Poll (Poll Detail)**
* **New Poll**
* **Leader board**

# **Break Each View Into a Hierarchy of Components**

**Components for Home Page**



Poll Tabs

Poll List

Poll

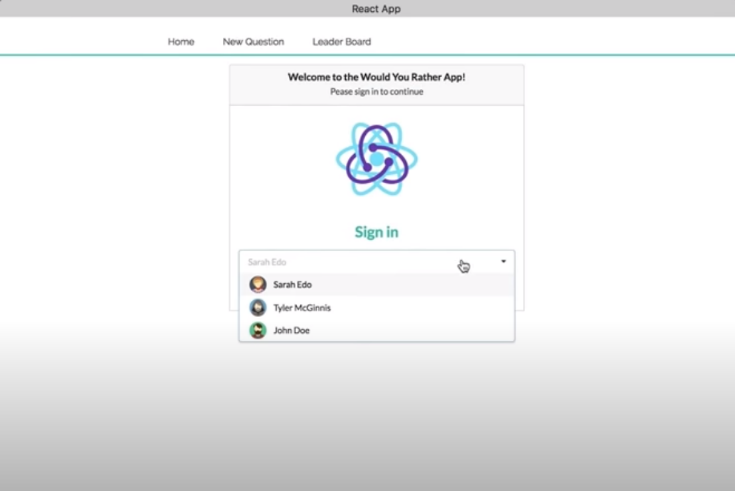
Navigation

Header

APP

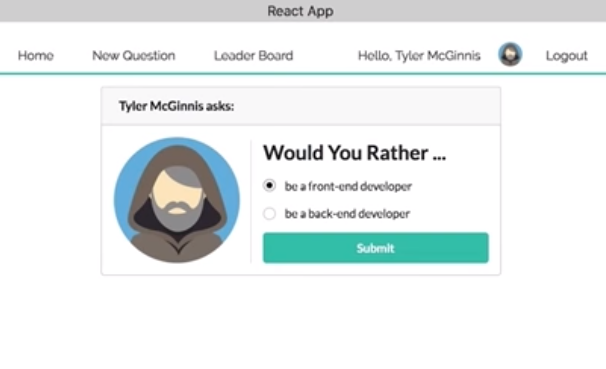
I broke this **Home view** into the following React Components:

* **App** - the overall container for the project
* **Header** – displays navigation, Signed in User, and Logout button)
* **Navigation** - displays the navigation
* **Poll Tabs** – tab to display unanswered questions and answered questions
* **Poll List** - responsible for the entire list of polls
* **Poll** - in charge of display the content for a single poll
* **Sign in Form** – user to log in



Sign In Form

**Components for Single Poll page**

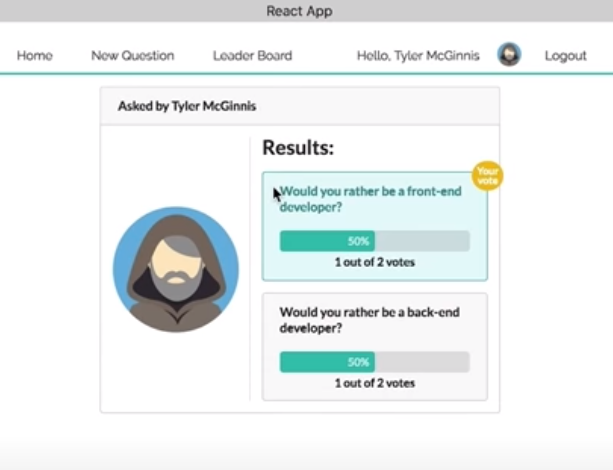
****

Poll Answer Form  
(if unanswered)

Navigation

Header

APP

****

Poll Result

(if answered)

APP

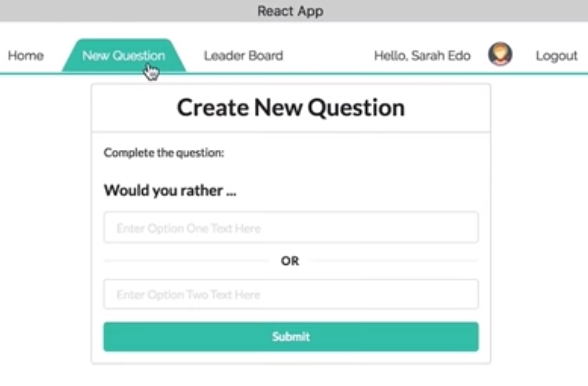
Navigation

Header

I broke this **Single Poll view** into the following React Components:

* **App** - the overall container for the project
* **Header** – displays navigation, Signed in User, and Logout button)
* **Navigation** - displays the navigation
* **Poll Answer Form** – If unanswered
* **Poll Result** – If answered

**Components for New Poll Page**



APP

New Poll

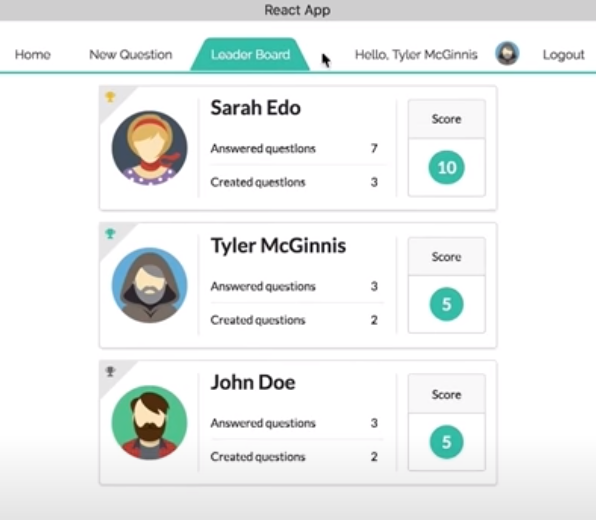
Header

Navigation

I broke this New Poll view into the following React Components:

* **App** - the overall container for the project
* **Header** – displays navigation, Signed in User, and Logout button)
* **Navigation** - displays the navigation
* **New Poll –** form to create a new poll / question

**Components for Leader board Page**

****

User

Navigation

Header

User List

APP

* + Each entry on the leaderboard should contain the following:
    1. **User’s name**
    2. **User’s picture**
    3. **Number of questions the user asked**
    4. **Number of questions the user answered**

\*\*Users should be ordered in descending order based on the sum of the number of questions they’ve asked and the number of questions they’ve answered. The more questions you ask and answer, the higher up you move.

I broke this Leader board view into the following React Components:

* **App** - the overall container for the project
* **Header** – displays navigation, Signed in User, and Logout button)
* **Navigation** - displays the navigation
* **User List** – responsible for the entire list of users
* **User** – in charge of display the content/score of each user

**All Components**

So, from the way I broke things down, the application will have the following components:

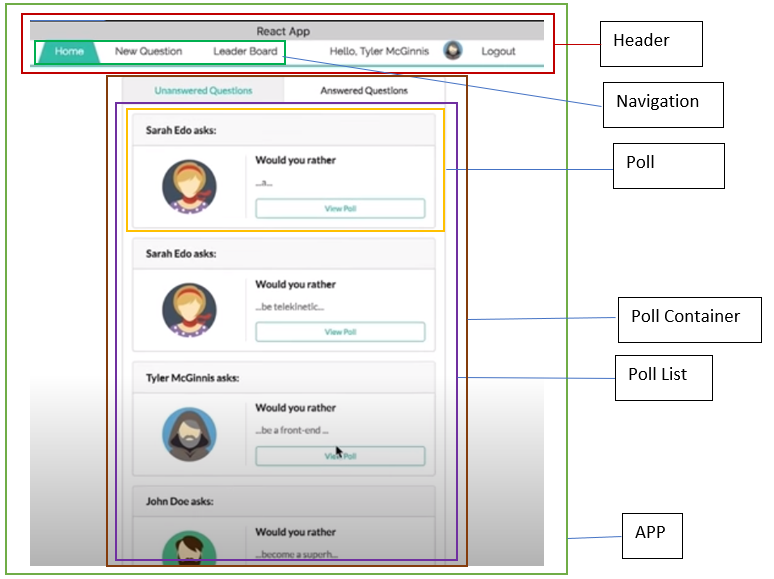
* + **App**
  + **Header**
  + **Navigation**
  + **Poll Tabs**
  + **Poll List**
  + **Poll**
  + **Sign in Form**
  + **Poll Answer Form**
  + **Poll Result**
  + **New Poll Form**
  + **User List**
  + **User**

**Determine What Events Happen in the App**

**Sign in Form**

* + Get a **list of all users**
  + Set the **signedInUser** in store

**Poll List Component**

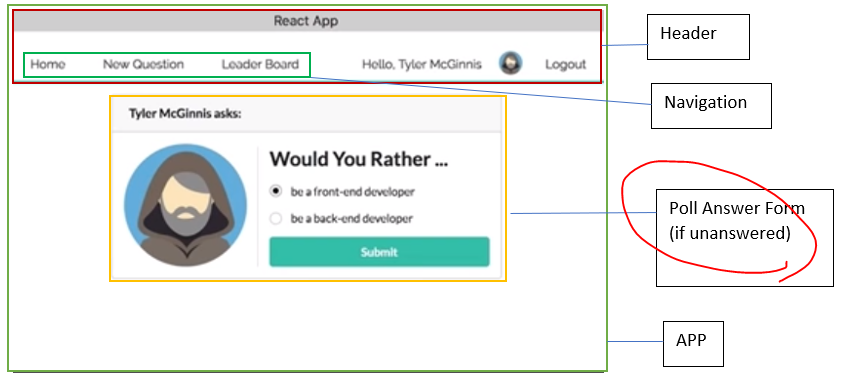


- get **a list of all polls** **(\_getQuestions)**

**Poll Component**

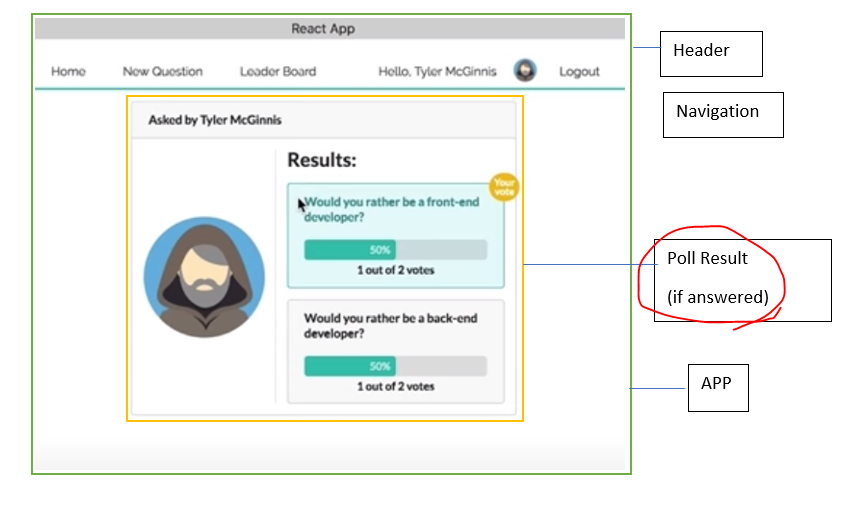
* + Get **particular poll** from a list of polls
  + Get **signedInUser** so user can view/vote the poll
  + Check the polls that user answered / unanswered

**Poll Answer Form**



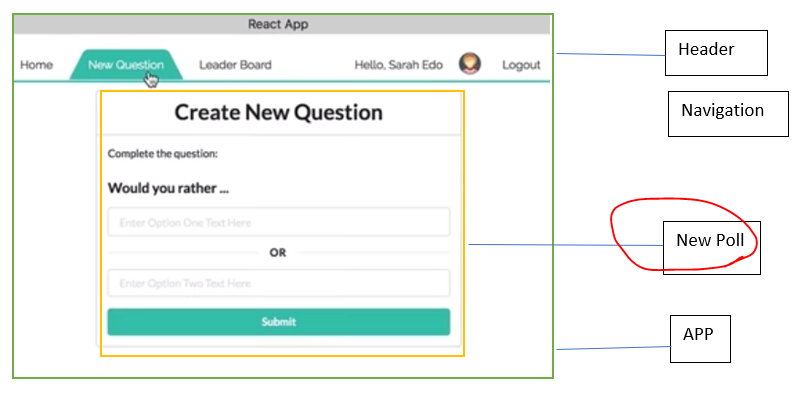
* + Get detail of **particular poll (**to show text of option)
  + Get **author’s picture from user**
  + **Save answer** of the user to this poll (**\_saveQuestionAnswer** ({ authedUser, qid, answer }))

**Poll Result**



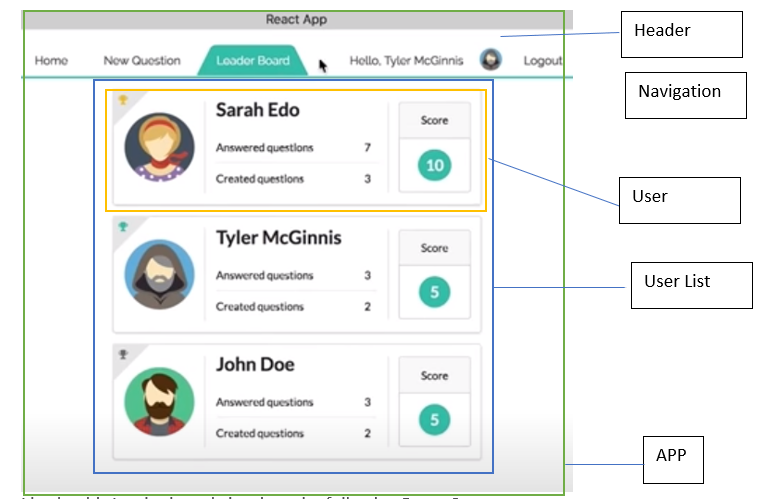
* + Get detail of **particular poll (**to show text of option)
  + Get **author’s picture from user**

**New Poll Form**



* + Get **signedInUser** so user can create a new poll
  + **Save the options of the new poll (\_saveQuestion)**

**User List**



* + Get a **list of all the users** **(\_getUsers)**
  + Get **signedInUser** so user can view the leaderboard

**User**

* + Get **particular user** from a list of users
  + Get **signedInUser** so user can view the leaderboard

**Determine What Data Lives in the Store**

**The Store**

**users (user)** : The result of a users action going through its users reducer.

**polls (poll)** : The result of a polls action going through its polls reducer.

**signedInUser** :The result of an signedInUser action going through its signedInUser reducer.

* + *get* the **users**
  + *get* the **polls**
  + *get* the **signedInUser**

**State of component**

**answer of poll,**

**options of poll**