Frontend Documentation

Start application:

```
cd frontend

npm run start
```

The front end will be started at port 3000. To use a different port, edit package.json:

```
"scripts": { "start": "set PORT=PORTNUMBER && react-scripts start" }
```

Technology Stack

React

React Hooks and routing are used extensively

- {useNaviagte, useState, useEffect}
- Browser Router for URL path management

Axios

Used to make HTTP requests and handle responses from the server.

Redux

Stores global state of user information:

- {useDispatch, useSelector}
- Redux-persist will use local storage.

Redux documentation

We define our user data slice/definition for redux in `userSlice.js`. We store the currently logged-in user's email(default is “”) in the state and have a reducer/method `changeEmail` to modify the storage state.

Then, we define our storage in `store.js` and use redux-persist to only allow component render after the state is loaded.

In `index.js`, we pass the storage as a global state into the application and when rendering, we display the component set as `loading`.

useDispatch

```
import { useDispatch } from 'react-redux'
import { changeEmail } from '../../../reduxStore/userSlice';
export default function Component() {
  const dispatch = useDispatch();     // define the dispatch function.
  dispatch(changeEmail("someemail")); // changes the email stored in
state.
```

useSelector

```
import { useSelector, useDispatch } from 'react-redux'
export default function Component() {
  // Contains the current email stored. Only proceed to next statements
until the email is retrieved.
  const currentUserEmail = useSelector((state) => state.email);
```

Axios

```
axios.defaults.withCredentials = true;                // needed to
retreive session id
axios.PROTOCAL(URL,
    {},
    {
        headers: {
            'Access-Control-Allow-Credentials': true,// needed to send
credentials
            'Access-Control-Allow-Origin': HOSTNAME, // needed to pass
CORS
        },
    }
).then(() => {                                        // Called when the
call is a success
}).catch(function () {                                // Called O.W.
});
```

Custom Components:

MomentieTimeline

Displays one or more vertical timelines. See the demo on App.js.

Attributes: `{ timelineList, setTimelineList, width, height, editMode, isSkill, section }`

- timelineList- useState variable that contains the passed-in data.

- setTimelineList - useState set method to change timelineList.
- width - width of each timeline
- height - height of each timeline
- editMode - display the edit view
- isSkill - when set to true, timeline should display a much simple version.
- section - name of this timeline.

MomentieTimelineItem

Displays a single timeline item.

Attributes: `{ index, timelineItem, width, editMode, deleteItem, editItem, isSkill }`

- index - index position inside the timeline array
- timelineItem - item data
  - Contract: The data should be an object looking like

```
{
    _id: 0, // This is manually added, not from backend. This is to
ensure react mapping
            // gives the correct information
    topic: "experience",
    title: "first",
    content:
"somethingasasddddddddddddddddddddddddddddddddddddddddddddddd",
    startTime: "2022-10-05T04:38:26.022Z",  // ISO stirngs
    endTime: "2022-10-05T04:38:26.022Z",    // ISO stirngs
  }
```

- width - width of the item
- editMode - display the edit view
- deleteItem - takes a topic and an index and deletes it from the view. Will re-render.
- editItem - takes a topic, index, field, and value that sets the item's field by this new value. Will not re-render
- isSkill - when set to true, timeline should display a much simple version.

MomentieTag

Displays a list of tags

Attributes: `{ tagList, setTagList, edit }`

- tagList - useState variable of list of tag data

```
• [{
    current: string
  }]
```

- setTagList - useState set method to change the tags.
- edit - display edit view when true.

Rate

Displays a rating value

Attributes: `{ rating, setRating, read, rate }`

- rating - useState variable a rating integer value
- setRating - useState set method to change the rating value
- read - readOnly if set to true.
- rate - a function called to handle a change in rating

MomentiePost

Displays a list of posts.

Attributes: `{ postList }`

- postList

```
• [{
      email: string
      content: string
  }]
```

MomentieUserList

Displays a list of users. Retrieves the profile image and provides navigation using the user in user list's email.

Attributes: `{ userList, cardPerPage }`

- cardPerPage: default 6. Controls how many cards you want per page.
- userList

```
• [{
      email: string
      username: string (optional)
      like: int <= 5
  }]
```

Miscellaneous

- `constants.js` You should store global constants here. Currently, only backendHost which is the backend hostname is stored