Progress

Set Up

Create-React-App

Starting a fresh project is always quite nice. I have the “create-react-app” package installed globally on my machine so it's nice and simple to get started with prepackaged configuration out of the box.

```
yarn add create-react-app
create-react-app alarm.me
```

Ejecting and SCSS

Process

This is new territory for me; I haven't ejected a project before so I followed a tutorial. There were a couple differences in the tutorial and what I did but nothing major. I followed the Webpack install path as I figured I wouldn't need to get any updates to create-react-app for this project. The process boiled down to these steps...

1. Add SASS dependencies
2. Eject
3. Configure Loaders
4. Configure Exclude
5. Modify Imports

Add SASS dependencies

So I have this nice and clean project setup and what is the first thing I go and do? Well you got me, the first thing I did was install a couple dependencies

```
yarn add sass-loader node-sass --dev
```

Eject

But the second thing I did was eject it to get access to config files create-react-app was hiding :(

```
yarn eject
```

Configure Loaders

In the module.exports.module.rules I added

```
//SCSS
{
    test: /\.scss$/,
    include: paths.appSrc,
    loaders: ["style-loader", "css-loader", "sass-loader"]
```

```
}
```

I think this loads files ending with .scss with these specific loaders (the dependencies we installed). I'm not sure if the location of this matters but I added it before the CSS rule

Configure Exclude

Next up was adding SCSS to the exclude list. I'm pretty sure this is necessary to make sure react doesn't process the file and its instead processed as SASS.

```
exclude: [/\.(js|jsx|mjs)$/, /\.html$/, /\.json$/, /\.scss$/]
```

Modify Imports

This is the simple step, just change any files to .scss and change any imports to reflect that

```
import './App.scss';
```

Routing

React-Router v4

Install

To handle routing for the application I went with react router. React router has packages that are split up depending upon platform, the only one I needed was react-router-dom.

```
yarn add react-router-dom
```

Wrap the App

Going into my index.js I then placed a Router component around my app.

```
import { Router } from 'react-router-dom'

ReactDOM.render((
  <Router>
    <App />
  </Router>
), document.getElementById('root'));
```

Defining Routes

```
export const _LANDING = '/'
export const _ACCOUNT = '/account'
export const _HOME    = '/home'
```

Import Routes

```
import * as routes from './constants/routes.js'
```

Switch and Route

After I imported my other components (Landing, Home, and Account), I set up the paths

```
<Switch>
      <Route exact path={routes._LANDING} component={Landing}/>
      <Route exact path={routes._HOME} component={Home}/>
      <Route exact path={routes._ACCOUNT} component={Account}/>
</Switch>
```

Authentication

Auth0 vs Firebase

My previous usage of authentication services was limited to using firebase. While I am tempted to stick with firebase with this project; I have heard a bit of interest in using Auth0. Not one to shun tech that I don't have a grudge against (looking at you Quartus Prime) I'm going to give it a try. If it turns out kinda difficult I will probably just integrate firebase.

Starting Integration

Installation

```
yarn add auth0-js
```

Also add the script into index.html

```
<script type="text/javascript"
src="node_modules/auth0-js/build/auth0.js"></script>
```

Authentication Service?

```
// src/Auth/Auth.js

import auth0 from 'auth0-js';

export default class Auth {
  auth0 = new auth0.WebAuth({
    domain: 'YOUR_AUTH0_DOMAIN',
    clientID: 'YOUR_CLIENT_ID',
    redirectUri: 'http://localhost:3000/callback',
    responseType: 'token id_token',
    scope: 'openid'
  });

  login() {
```

```
    this.auth0.authorize();
  }
}
```

The domain and clientID could be found under account settings

While this did work and take me to a login page it is also giving me an syntax error and Im not entirely sure why...

```
Uncaught SyntaxError: Unexpected token <
```

GraphQL

Difficulties

I had a non-insignificant amount of trouble in setting up GraphQL on the frontend and backend. I'm just gonna go over the path that got it working for now.

Backend

```
yarn add express graphql express-graphql cors
```

Schema

This is what my current schema looks like, it will need to be modified

```
// GraphQL schema
var schema = buildSchema(`
      type Query {
      alarm(id: Int!, userID: Int!): Alarm
      alarms(userID: Int): [Alarm]
      },
      type Mutation {
      createAlarm(userID: Int!, dateTime: String!, title: String!, note:
String!, color: String!): Alarm
      updateAlarm(id: Int!, userID: Int!, dateTime: String!, title:
String!, note: String!, color: String!): Alarm
      deleteAlarm(id: Int!): Alarm
      },
      type Alarm {
      id: Int
      userID: Int
      dateTime: String
      title: String
      note: String
```

```
      color: String
      },
      type User {
      id: Int
      name: String
      }
`);
```

Data

```
let alarmData = [
  {
      id: 1,
      userID: 1,
      dateTime: "2010-10-20 4:30",
      title: "Wake Up",
      note: "Go to class!",
      color: "green"
  }
]
let idCount = alarmData.length;
```

Resolvers

```
var deleteAlarm = ({id}) => {
  alarmData.forEach((alarm, i) => {
      if(alarm.id === id) {
      return alarmData.splice(i,1);
      }
  })
}

var getAlarm = (args) => {
  if(alarmData.length == 0) return null;
  return alarmData.filter(alarm => {
      return alarm.id === args.id && alarm.userID === args.userID;
  })[0];
}

var getAlarms = (args) => {
  if(args.userID) {
      return alarmData.filter(alarm => alarm.userID === args.userID);
```

```
  }
  else {
      return alarmData;
  }
}

var createAlarm = (args) => {
  let newAlarm = {
      id: idCount++,
      userID: args.userIS,
      dateTime: args.dateTime,
      title: args.title,
      note: args.note,
      color: args.color
  };
  alarmData.push(newAlarm);
  return newAlarm;
}

var updateAlarm = (args) => {
  let alarmIndex = alarmData.findIndex((alarm => alarm.id == args.id));
  //dateTime: String!, title: String!, note: String!, color: String!
  if(args.dateTime) {alarmData[alarmIndex].dateTime = args.dateTime;}
  if(args.title) {alarmData[alarmIndex].title = args.title;}
  if(args.note) {alarmData[alarmIndex].note = args.note;}
  if(args.color) {alarmData[alarmIndex].color = args.color;}

  return alarmData[alarmIndex];
}

// Root resolver
var root = {
      alarm: getAlarm,
      alarms: getAlarms,
      deleteAlarm: deleteAlarm,
      createAlarm: createAlarm
};
```

Express

```
// Create an express server and a GraphQL endpoint
var app = express();
```

```
app.use('/graphql', cors(), express_graphql({
      schema: schema,
      rootValue: root,
      graphiql: true
}));
app.listen(4000, () => console.log('Express GraphQL Server Now Running On
localhost:4000/graphql'));
```

Starting the Server

All you need to do to start the server is run

```
node server.js
```

Hot Reloading the server

```
yarn global add nodemon
```

Then to start the server its just

```
nodemon server.js
```

Or do like I did and just add that to the package.json like so

```
"scripts": {
      "start": "nodemon server.js"
  }
```

And all you need to do now is call

```
yarn start
```

Simplistic and uniform with create-react-app's start command :)

Frontend

I ended up struggling a bit here also as I went straight for adding the client Apollo. While that's all well and good, I didn't realize that I would manually specify how the queries would change and update the data. So the DELETE_ALARM mutation needed an update parameter passed to it that would update the cache appropriately. Effectively cutting out the need for another query to be sent as we know what the list should contain anyway.

Installation

```
yarn add react-apollo apollo-boost
```

Provider and Client

In App.js

```
import { ApolloProvider } from "react-apollo";
```

```
import ApolloClient from "apollo-boost";

const client = new ApolloClient({
  uri: "http://localhost:4000/graphql"
});

<ApolloProvider client={client}>
    <div className="App">
        ...
    </div>
  </ApolloProvider>
```

Now the provider automatically passes the client prop down all components so all can query the client.

Query Creation

```
// src/components/Home/gqlQueries.js
import { gql } from 'apollo-boost'

export const GET_ALARMS = gql`
  {
      alarms {
      id
      userID
      dateTime
      title
      note
      color
      }
  }
`;

export const DELETE_ALARM = gql`
  mutation deleteOneAlarm($id: Int!) {
      deleteAlarm(id: $id) {
      id
      }
  }
`;

export const CREATE_ALARM = gql`
  mutation createAlarm($userID: Int!, $dateTime: String!, $title: String!,
$note: String!, $color: String! ) {
```

```
      createAlarm(userID: $userID, dateTime: $dateTime, title: $title,
note: $note, color: $color) {
      id
      userID
      dateTime
      title
      note
      color
      }
  }
`;
```

Sending Queries and Mutations

First import

```
// src/components/Home/home.js
import { Query, Mutation } from "react-apollo";
```

Sorry for the messed up formatting. Here I am executing a query against the server and returning a list of alarms as well as a nexted Mutation for every alarm. This allows for users to delete indivual alarms.

```
<Query
      query={GET_ALARMS}
      >
            {({ loading, error, data }) => {
            if (loading) return <p>Loading...</p>;
            if (error) return <p>Error :(</p>;

            return data.alarms.map((alarm, i) => (
            <div key={i}>
                  {i+1}. <h3>{alarm.title}</h3>
                  <p>{alarm.note}</p>
                  <p>{moment(alarm.dateTime).fromNow()}</p>
                  <Mutation
                  mutation={DELETE_ALARM}
                  update={(store, { data: { alarms } }) =>
                  this._updateStoreAfterDelete(store, alarms, alarm.id)
                  }
                  >
                  {(deleteAlarm, { data }) => (
                  <div>
                        <button type="submit" onClick={() => {
```

```
                        deleteAlarm({ variables: { id: alarm.id } });
                        }}>
                        Delete
                        </button>
                    </div>
                    )}
                    </Mutation>
                </div>
                ));
                }}
    </Query>
```

And here is the code to update the cache after that button is pressed

```
  _updateStoreAfterDelete = (store, alarms, alarmId) => {
      const data = store.readQuery({ query: GET_ALARMS })
      let alarmIndex = data.alarms.findIndex(alarm => alarm.id ===
alarmId);
      let newAlarmData = data.alarms.splice(alarmIndex,1);
      data.alarmData = newAlarmData;
      store.writeQuery({ query: GET_ALARMS, data })
  }
```

History

Installation

yarn add history

Usage

I created another file called history.js to section it off. I realize it was not too necessary but here it is.

```
import createHistory from 'history/createBrowserHistory'

export const history = createHistory();
```

Then simply import history and pass it to the router as a prop

```
import { history } from './history.js'

ReactDOM.render((
  <Router history={history}>
      <App />
```

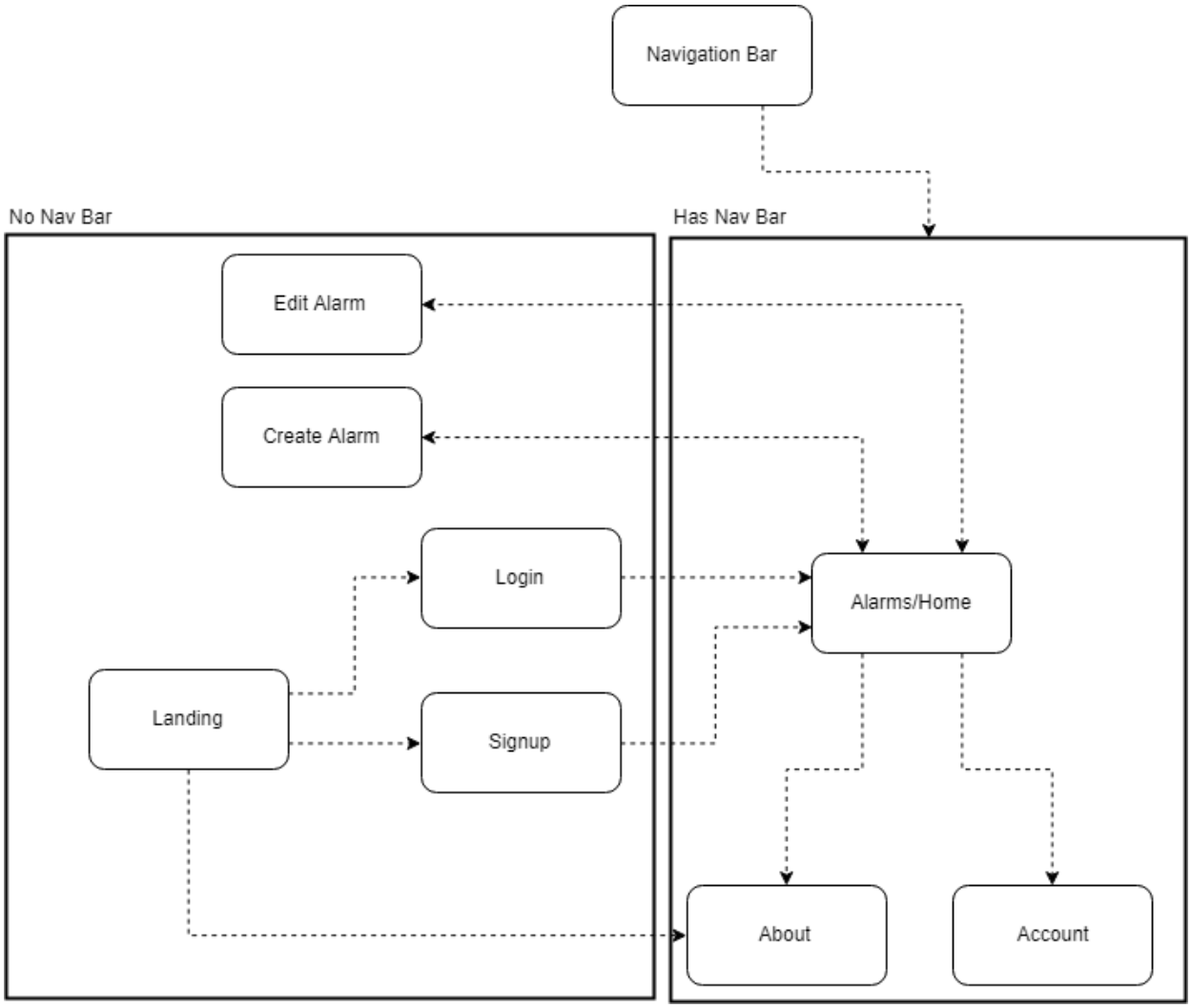
```
  </Router>
), document.getElementById('root'));
```

Now you can do things like push the Home page onto the view, navagating the router to the ‘/home’ path for me here.

```
pushHistory = () => {
      history.push({ pathname: routes._HOME })
  }
```

Late Design Work

Screenflow

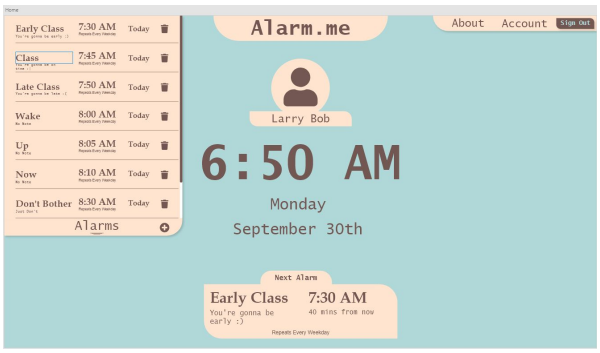


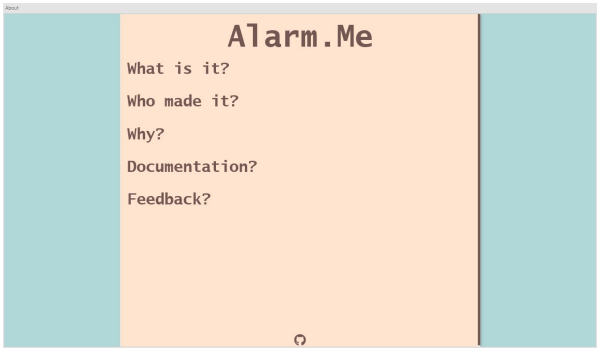
Screen Sketches

Limited Palette

Designs

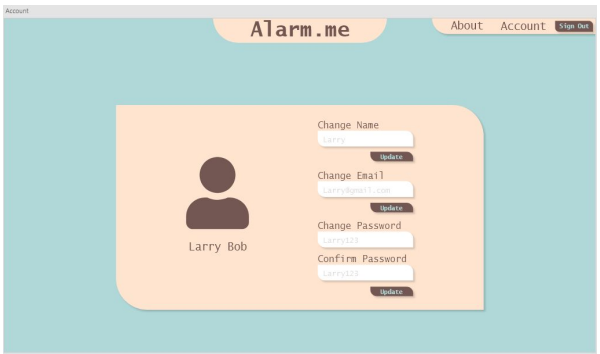

Alarm.me
Sign Up
Log In
About

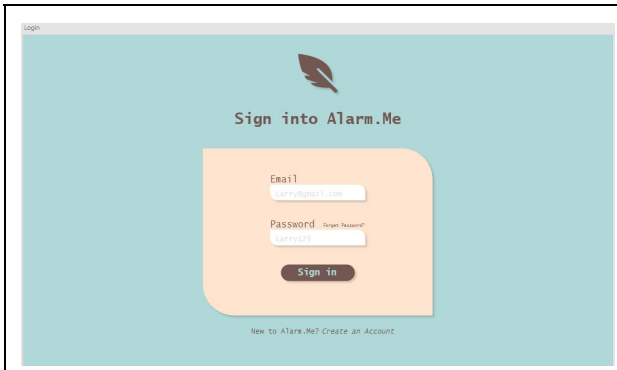

Alarm.me
About
Account
Sign Out
Early Class 7:30 AM Today
Class 7:45 AM Today
Late Class 7:50 AM Today
Wake 8:00 AM Today
Up 8:05 AM Today
Now 8:10 AM Today
Don't Bother 8:30 AM Today
Alarms
Larry Bob
6:50 AM
Monday
September 30th
Next Alarm
Early Class 7:30 AM
You're gonna be early :)
40 mins from now
Repeats Every Weekday

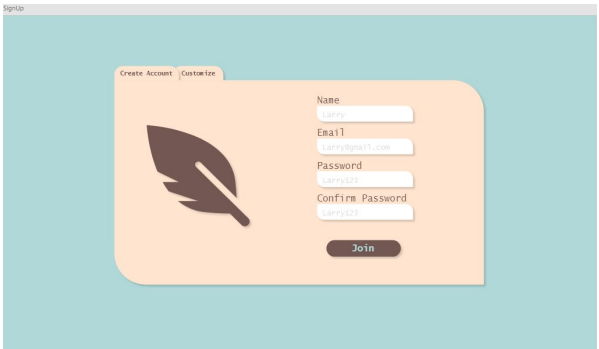

Alarm.Me
What is it?
Who made it?
Why?
Documentation?
Feedback?


Edit Alarm
Title
Note
Date/Time
Repeats
Cancel
Update


Create New Alarm
Title
Note
Date/Time
Repeats
Cancel
Create


Alarm.me
About
Account
Sign Out
Larry Bob
Change Name
Update
Change Email
Update
Change Password
Update
Confirm Password
Update





Getting the server set up

AWS Instance

Node + Express Test

tmux

MySQL

Installing


```
[
    {
        "id": 1,
        "userID": 1,
        "dateTime": "2018-09-30T19:30:07.000Z",
        "title": "TEST",
        "note": "This is a test Alarm",
        "color": "green"
    },
    {
        "id": 2,
        "userID": 1,
        "dateTime": "2018-09-30T19:40:07.000Z",
        "title": "TEST2",
        "note": "This is second test Alarm",
        "color": "blue"
    }
]
```