https://github.com/facebook/create-react-app

READYMADE BOILER PLATE WITH THE BASIC REACT APP with REDUX:
https://github.com/codewithbernard/react-redux-boilerplate

FOR THIS EXAMPLE - USE MY GIT:
YOUR FIRST REACT AND REDUX APP with API CALL:
https://github.com/mz75/React_Redux_Axios_BoilerPlate

BASIC STEPS TO FOLLOW – TOTAL 8 STEPS:

1. INSTALL THE NORMAL REDUX PACKAGES + THUNK (THUNK IS THE DISPATCHER) & AXIOS AXIOSTHROUGH NPM

a. ***Install React App***:

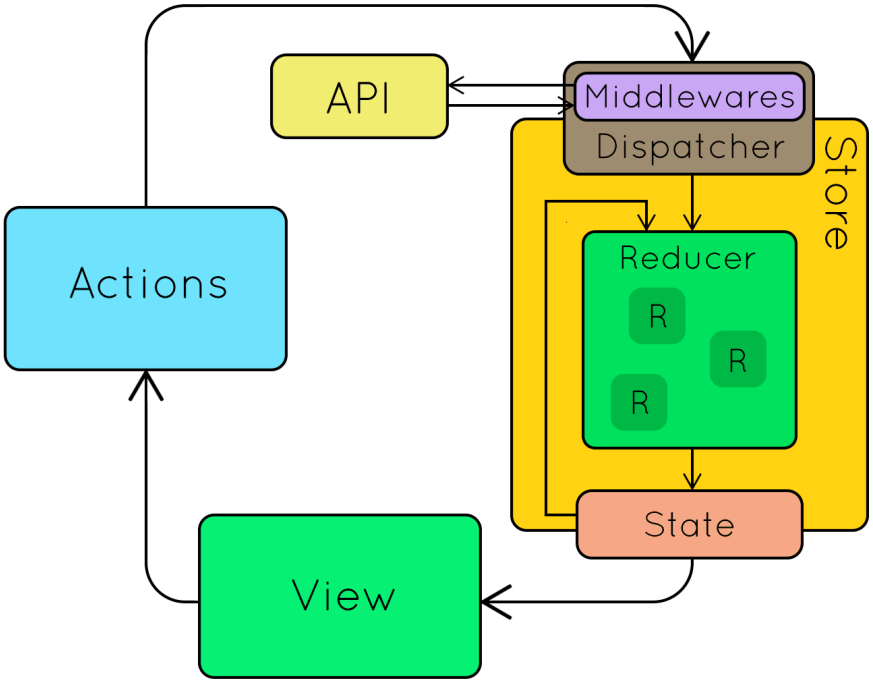
npx create-react-app my-app
// **Reference:** https://github.com/facebook/create-react-app#create-react-app-

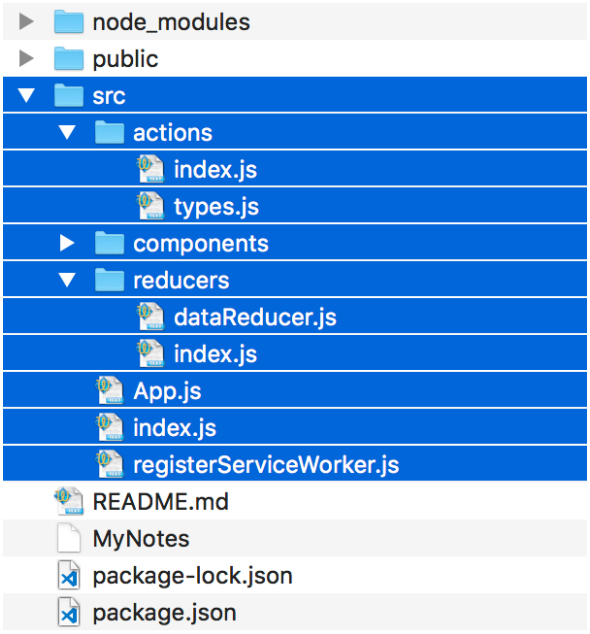
b. ***Install Redux, Thung & axios :***

npm install --save redux react-redux redux-thunk
// **Reference**: https://github.com/gaearon/redux-thunk#redux-thunk

npm install --save axios
// **Reference**: https://www.npmjs.com/package/axios#installing

You need a chrome plugin called “ExtensionAllow-Control-Allow-Origin” for out of domain ajax calls.
// **Reference**: https://chrome.google.com/webstore/detail/allow-control-allow-origi/nlfbmbojpeacfghkpbjhddihlkkiljbi





2. CREATE FOLDERS IN THE SRC FOLDER AKA ROOT FOLDER:

a. actions (The folder where you call the APIs – in our example)

DETAILS:
This folder must have a file called **'index.js'**,
This **'index.js'** is the file where you will make the **API calls** and
check your **self-defined data types** by calling the **'types.js'** file
(important: **'types.js'** only returns a name against an alias – check file).

b. reducers (The folder where you perform actions on APIs – in our example)

DETAILS:
This folder must have a file called **'index.js'** this is the record/collection of all reducers.

c. components (For normal react components – not used in our example)

DETAILS:
This folder has all your components you normally write in React

3. MAKE BASIC CHANGES INTO YOUR ROOT FOLDER'S INDEX.JS FILE (THE ROOT FOLDER VERSION)

```
// NPM TIP: To add npm package run npm install --save <package_name>
// Default – No change
import React from "react";

// Default – No change
import ReactDOM from "react-dom";

// Default – No change
import App from "./App";

// Default – No change
import registerServiceWorker from "./registerServiceWorker";

// NEW: Call the Provider COMPONENT using 'react-redux' -
// This component provides your app with 'store'

import { Provider } from "react-redux";

// NEW: Check the redux diagram–we need to create a store & apply middlewear
// 'createStore'& 'applyMiddleware' are 'functions' that we are able to call once import them from redux.
// GENERALL RULE: If the first letter is capital than its likely to be a component and in other cases its likely to be a function or a variable.

import { createStore, applyMiddleware } from "redux";

// NEW: calling directory will automatically look for index.js & this is a file which imports other reducers and merge them. Mainly in our example it is "DataReducer.JS"
import reducers from "./reducers";

// NEW: Importing reduxThunk [which is the Dispatcher aka middlewear part]
import reduxThunk from "redux-thunk";

// NEW: Finally initiating the store and passing reducers plus middlewear as reduxThunk – because it's a dispatcher.
const store = createStore(reducers, applyMiddleware(reduxThunk));

// Default – No change
ReactDOM.render(

// NEW: Wrap the lone <App /> tag with <Provider store={store}> passing the store as attribute.
<Provider store={store}>
<App />
</Provider>, document.getElementById("root")
);
registerServiceWorker();
```

ACTIONS FOLDER CHANGES:

4. INDEX.JS IS TO BE CREATED INSIDE THE ACTIONS FOLDER.

DETAILS:
This is the file where you will call your main APIs using axios
Here is a sample of the file code with details:

```
// Calling a typical alias/object self-defined data type from types.js inside actions folder.
import { FETCH_DATA } from "./types";

// Calling a axios using axios – for calling unhosted external API
// Here you will require the help from that chrome plug-in for calling external API as it allow you to do that.
```

```
// Common import for both shortcut or longcut code.
import axios from "axios";

// Shortcut of the code
// NOTE– If you find this difficult to understand. I have a longcut version under this.

export const fetchData = () => async dispatch => {
const res = await axios.get("https://swapi.co/api/people");
dispatch({
    type: FETCH_DATA,
    payload: res.data.results
});
};

// Longcut of the code
// NOTE– If you find shortcut difficult to understand.

export const fetchData = () => {
return async dispatch => {
    const res = await axios.get("https://swapi.co/api/people");
    console.log(res);
};
};
```

5. TYPES.JS IS TO BE CREATED INSIDE THE ACTIONS FOLDER.

DETAILS:

This is the file where for creating aliases for self-defined data types, returning a (const) variable only.
Here is a sample of the file code with details:

```
// Plain strings for types
export const FETCH_DATA = "FETCH_DATA";
```

REDUCERS FOLDER CHANGES:

6. DATAREDUCER.JS IS TO BE CREATED INSIDE THE REDUCERS FOLDER.

DETAILS:

Here is a sample of the file code with details:

```
// Calling a typical alias/object self-defined data type from types.js inside actions folder.
import { FETCH_DATA } from "../actions/types";

// state is passed as an array although it should be obj along with 'action'.
// we expect returned state to be an Array obj – hence.
// IMPORTANT: here we are exporting a function, and this is an example of arrow function and that function have one switch statement which evaluate the action type

export default (state = [], action) => {
    switch (action.type) {
        case FETCH_DATA:
        return action.payload;
        default:
        return state;
    }
};
```

7. INDEX.JS IS TO BE CREATED INSIDE THE REDUCERS FOLDER.

DETAILS:

This is the file where you will combine all your reducers, but we have only one in our example "dataReducer.js"
Here is a sample of the file code with details:

```
// Calling reducers, you have just created. So that you don't need to call them individually, but we have only one here 'dataReducer.js'
import dataReducer from "./dataReducer";

// For combining all reducers by using redux, you need to import the object first by using redux.
import { combineReducers } from "redux";

export default combineReducers({

// Assigning the key value and returning it back after combining.
data: dataReducer
});
```

OTHERS:

8. **RUN APPLICATION**

DETAILS:

Write using terminal (inside the "react-redux-boilerplate" folder)

```
npm start
```