**README.md**

In Scope:

* API for CRUD Operations for ProjectTasks
* React/Redux Front End to support CRUD Operations for ProjectTasks

This project was bootstrapped with [Create React App](https://github.com/facebook/create-react-app).

Create React App is divided into two packages:

* create-react-app is a global command-line utility that you use to create new projects.
* react-scripts is a development dependency in the generated projects (including this one).

You almost never need to update create-react-app itself: it delegates all the setup to react-scripts.

When you run create-react-app, it always creates the project with the latest version of react-scripts so you’ll get all the new features and improvements in newly created apps automatically.

**Folder Structure**

After creation, your project should look like this:

my-app/

README.md

node\_modules/

package.json

public/

index.html

favicon.ico

src/

App.css

App.js

App.test.js

index.css

index.js

logo.svg

For the project to build, **these files must exist with exact filenames**:

* public/index.html is the page template;
* src/index.js is the JavaScript entry point.

You can delete or rename the other files.

You may create subdirectories inside src. For faster rebuilds, only files inside src are processed by Webpack.  
You need to **put any JS and CSS files inside src**, otherwise Webpack won’t see them.

Only files inside public can be used from public/index.html.  
Read instructions below for using assets from JavaScript and HTML.

You can, however, create more top-level directories.  
They will not be included in the production build so you can use them for things like documentation.

If you have Git installed and your project is not part of a larger repository, then a new repository will be initialized resulting in an additional .git/ top-level directory.

Available Scripts

In the project directory, you can run:

### **npm start**

Runs the app in the development mode.  
Open [http://localhost:3000](http://localhost:3000/) to view it in the browser.

The page will reload if you make edits.  
You will also see any lint errors in the console.

### **npm test**

Launches the test runner in the interactive watch mode.  
See the section about [running tests](https://github.com/AgileIntYoutube/freeProjectBoardReactReduxFrontend#running-tests) for more information.

### **npm run build**

Builds the app for production to the build folder.  
It correctly bundles React in production mode and optimizes the build for the best performance.

The build is minified and the filenames include the hashes.  
Your app is ready to be deployed!

See the section about [deployment](https://github.com/AgileIntYoutube/freeProjectBoardReactReduxFrontend#deployment) for more information.

### **npm run eject**

**Note: this is a one-way operation. Once you eject, you can’t go back!**

If you aren’t satisfied with the build tool and configuration choices, you can eject at any time. This command will remove the single build dependency from your project.

Instead, it will copy all the configuration files and the transitive dependencies (Webpack, Babel, ESLint, etc) right into your project so you have full control over them. All of the commands except eject will still work, but they will point to the copied scripts so you can tweak them. At this point you’re on your own.

You don’t have to ever use eject. The curated feature set is suitable for small and middle deployments, and you shouldn’t feel obligated to use this feature. However we understand that this tool wouldn’t be useful if you couldn’t customize it when you are ready for it.

## Installing a Dependency

The generated project includes React and ReactDOM as dependencies. It also includes a set of scripts used by Create React App as a development dependency. You may install other dependencies (for example, React Router) with npm:

npm install --save react-router-dom

Alternatively you may use yarn:

yarn add react-router-dom

This works for any library, not just react-router-dom.

## Fetching Data with AJAX Requests

React doesn't prescribe a specific approach to data fetching, but people commonly use either a library like [axios](https://github.com/axios/axios) or the [fetch() API](https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API) provided by the browser.

The global fetch function allows you to easily make AJAX requests. It takes in a URL as an input and returns a Promise that resolves to a Response object. You can find more information about fetch [here](https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch).

A Promise represents the eventual result of an asynchronous operation, you can find more information about Promises [here](https://www.promisejs.org/) and [here](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise). Both axios and fetch() use Promises under the hood. You can also use the [async / await](https://davidwalsh.name/async-await) syntax to reduce the callback nesting.