Project Report

For WEB 701 Assessment 3 Graduate Diploma in IT

JUNE 2018

DOCUMENTED BY
Frank Fernando
Frank-wickramarathna@live.nmit.ac.nz

Contents

Project Title

"News showcase" is a React, Redux, and Redux Saga web application powered by News API to showcase comprehensive news. The aim of this document is to help you identify the implementation of the project.

Project code is available on Github.
Application link: https://news-showcase.herokuapp.com/

Technologies Used

This web application is built with use of some of the latest front end technologies. Such as

- React - JavaScript library for building user interfaces.
- Redux - A predictable state container for **JavaScript** apps.
- Redux Saga – A library which makes the applications side effects easier to maintain.

React:
ReactJS is an open source Javascript library developed by Facebook and community. Its initial release was in March 2013. The library mainly used to build user interfaces especially for single page applications, widely employed by companies like Instagram, Netflix, Paypal, Apple and more.

It seems like React is more famous for social media applications that we love like Instagram, Facebook, and Netflix etc. But React has wide variety of applications from producing autonomous cars to web applications, command line interfaces, native mobile applications etc which makes this library highly useful language to learn.

Below are some of the important features of React:

- Single way data flow
    - In React, immutable values are passed to the component's renderer. Component cannot directly modify the values
- Simple and easy to understand.
- Easy to learn

- Cross platform support – IOS, Android and Web
- Simple to debug
- High in performance & use of JSX

Redux:

Redux is a state management tool for JavaScript applications. It is also a concept of data communication and storing within an application in the browser. By using Redux in your application you can store all required data for generate the view similar to database which helps to query SELECT, INSERT, and UPDATE the data into JSON database known as single-state tree.

Redux follows three main principles

1. Single source of truth – the state of your whole application is stored in an object tree.
2. State is read only – using an action, it is possible to change the state.
3. Changes are made with pure function – pure reducers are used to specify the state is changed by an action.

Redux is an outcome of following prior arts.

- Flux
- Elm – functional programming language
- Immutable – JavaScript library
- Baobab
- RxJS – way of manage complexity of asynchronous application.

Redux ecosystem is well explained this article.

Redux Saga:

Redux-saga *is a library that aims to make application side effects (i.e. asynchronous things like data fetching and impure things like accessing the browser cache) easier to manage, more efficient to execute, simple to test, and better at handling failures.*

This helps your Redux application to stay in-sync with external APIs. It is a Redux middleware which can be start, pause and cancelled form the main application with Redux actions. Also it has access to the Redux state and can dispatch Redux actions.

It uses an ES6 feature called Generators to make those asynchronous flows easy to read, write and test (similar to async / await, with more extra features).

Saga Effects:

It provides some helper effects to spawn tasks when specific actions are dispatched to the store. Below are some of the effects,

- takeEvery
- takeLatest
- call
- put

Here is the link to original paper done by Hector Garcia-Molina & Kenneth Salem.

Implementation

Framework Implementation:

As we are building a news showcase application, I will explain the implementation initiate with basic react setup, organizing my project workflow, defining routes, setting up Redux and Redux saga finally deployment to Heroku.

- Install package globally - `npm install -g create-react-app`.
- Create news-showcase application - `create-react-app news-showcase`
- Install other dependencies –

  ```
  npm install --save redux redux-saga react-router@2.4 react-redux
  rm -rf src/**  # To clean the default sample app
  ```

After the installation, the React, Redux and Redux saga powered project should follow below workflow.

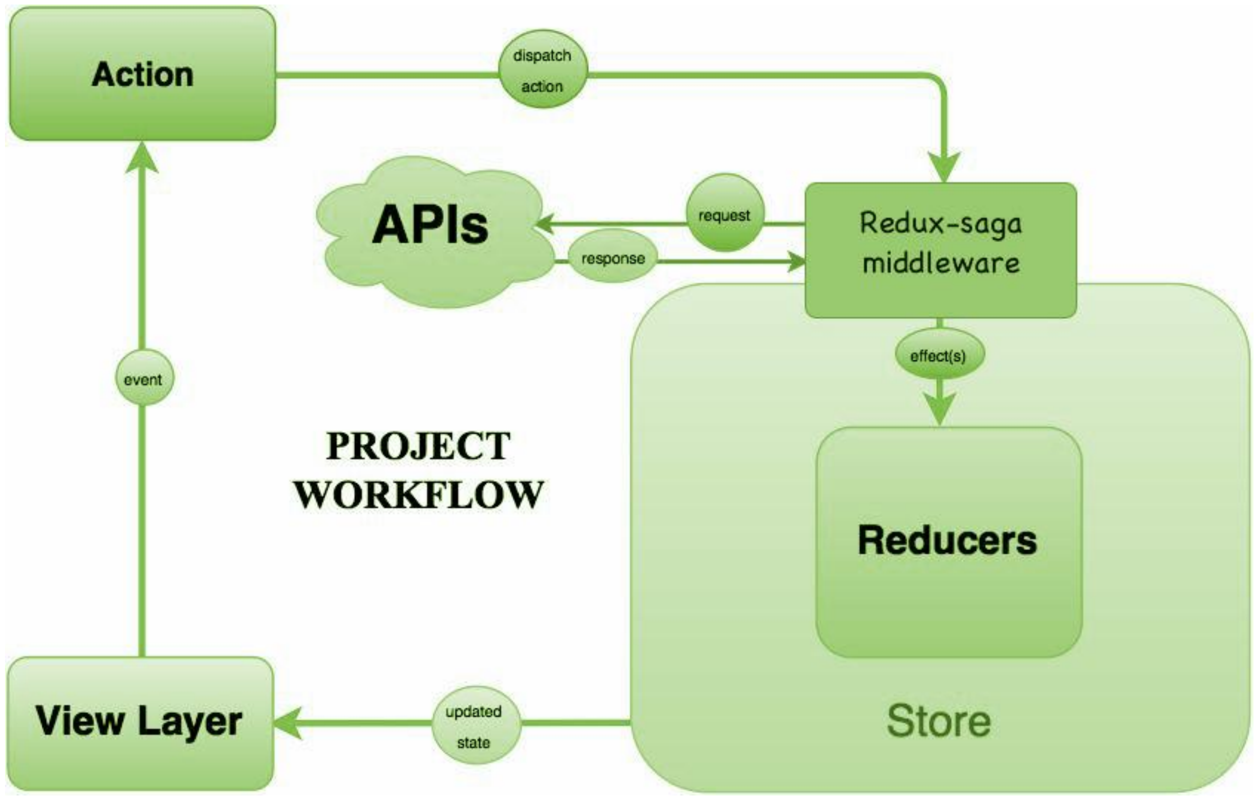


Fig. 1: Application workflow

- **View Layer** is the React component. It makes the request for an action i.e. button click
- **Action** is the action creator which returns a object with the action type and the payload which is optional.
- **Redux-saga** is the Redux middleware which handles all asynchronous News API requests.
- **APIs** are the locations which points to fetch different types of news.
- **Reducers** are the pure functions which construct the state.
- **Store** is the single object which holds the complete state of the application.

Once the store receives an updated state, it transmits the view layer to be updated to the latest data.

Code Implementation:

1. **Initial Setup**
    - App.js

```
import React, { Component, PropTypes } from 'react';

class App extends Component {
  render() {
    return (
      <div className="container-fluid text-center">
        {this.props.children}
      </div>
    );
  }
}
App.propTypes = {
  children: PropTypes.object.isRequired
};
export default App;
```

App component is the parent component of the application. **this.props.children** is the place where other child component is rendered.

- routes.js

```
import React from 'react';
import { Route, IndexRoute } from 'react-router';
import App from './App';
import NewsPage from './components/NewsPage';


export default (
  <Route path="/" component={App}>
    <IndexRoute component={NewsPage} />
    <Route path="news" component={NewsPage} />
  </Route>
);
```

Map components to different routes. The parent components and serve as the entrance to other React components. IndexRoute is the homepage component to NewsPage.

- index.js

```
import ReactDOM from 'react-dom';
import React from 'react';
import { Router, browserHistory } from 'react-router';
import { Provider } from 'react-redux';
import configureStore from './store/configureStore';
import routes from './routes';

// Initialize store
const store = configureStore();

ReactDOM.render(
  <Provider store={store}>
    <Router history={browserHistory} routes={routes} />
  </Provider>, document.getElementById('root')
);
```

index.js is the entrance to our application.

Note: I have included bootstrap CDN in our public/index.html and styles/style.css for custom styling.

- api.js

```
const NEWS_API_KEY = '05677d4958ee4427bf4bce9b88eceb12'
const NEWS_API_URL = 'https://newsapi.org/v2/'


export const anyNews = (searchQuery) => {
    const NEWS_API_ENDPOINT =
`${NEWS_API_URL}everything?q=${searchQuery}&apiKey=${NEWS_API_KEY}`;
    var req = new Request(NEWS_API_ENDPOINT);
    return fetch(req)
        .then(function (response) {
            return response.json();
        });
};

export const topNews = (searchQuery) => {
    const NEWS_API_ENDPOINT = `${NEWS_API_URL}top-
headlines?country=nz&apiKey=${NEWS_API_KEY}`;
    var req = new Request(NEWS_API_ENDPOINT);
    return fetch(req)
        .then(function (response) {
            return response.json();
        });
};

export const sourceNews = (searchQuery) => {
    const NEWS_API_ENDPOINT = `${NEWS_API_URL}top-
headlines?sources=${searchQuery}&apiKey=${NEWS_API_KEY}`;
    var req = new Request(NEWS_API_ENDPOINT);
    return fetch(req)
        .then(function (response) {
            return response.json();
        });
};
```

Here first I have defined the News API key and the API URL then three arrow functions to get news.

- NewsPage.js

This page works as the main layout for the applications which imports ***CategoryPage*** and ***ArticlesPage*** inside it.

2. **Action Creators Setup:**
    - constants/actionTypes.js

```
export const SEARCH_TOP_NEWS_REQUEST = 'SEARCH_TOP_NEWS_REQUEST';
export const SEARCH_TOP_NEWS_SUCCESS = 'SEARCH_TOP_NEWS_SUCCESS';
export const SEARCH_TOP_NEWS_ERROR = 'SEARCH_TOP_NEWS_ERROR';
export const SEARCH_ANY_NEWS_REQUEST = 'SEARCH_ANY_NEWS_REQUEST';
export const SEARCH_ANY_NEWS_SUCCESS = 'SEARCH_ANY_NEWS_SUCCESS';
export const SEARCH_ANY_NEWS_ERROR = 'SEARCH_ANY_NEWS_ERROR';
```

These are constants which are placed in a different file which is a good practice.

- actions/newsActions.js

```
import * as types from '../constants/actionTypes';

export const searchTopNewsAction = () => ({
    type: types.SEARCH_TOP_NEWS_REQUEST,
  });

export const searchAnyNewsAction = (payload) => ({
  type: types.SEARCH_ANY_NEWS_REQUEST,
  payload
});

export const searchSourceNewsAction = (payload) => ({
  type: types.SEARCH_SOURCES_NEWS_REQUEST,
 payload
});
```

Here it returns an action type with the optional payload.

3. Setting up state management system

As shown in the Fig.1 there are some wonderful concepts are implemented in the state management.

The Store keeps the whole state tree. Once an action is dispatched it delegates the reducer by passing the current state alongside the action object. If the reducer returns a new state the store updates.

Reducers are pure function which has no side effects, no mutation, no API calls. It creates the new states and returns it to the store.

- reducers/newsReducer.js

```
import initialState from './initialState';
import * as types from '../constants/actionTypes';

// Handles image related actions
export default function (state = initialState.news, action) {
        switch (action.type) {
        case types.SEARCH_ANY_NEWS_SUCCESS:
            return { ...state, articles: action.news.articles };
        case types.SEARCH_ANY_NEWS_ERROR:
            return { ...state }
        case types.SEARCH_TOP_NEWS_SUCCESS:
            return { ...state, articles: action.news.articles };
        case types.SEARCH_TOP_NEWS_ERROR:
            return { ...state }
        case types.SEARCH_SOURCES_NEWS_SUCCESS:
            return { ...state, articles: action.news.articles };
        case types.SEARCH_SOURCES_NEWS_ERROR:
            return { ...state }
        default:
            return state;
    }
}
```

This is my ***newsReducer*** which returns new state tree to the store.

- store/configureStore.js

```
import { createStore, applyMiddleware } from 'redux';
import createSagaMiddleware from 'redux-saga';
import rootReducer from '../reducers';
import rootSaga from '../sagas'; // TODO: Next step

//  Returns the store instance
// It can  also take initialState argument when provided
const configureStore = () => {
  const sagaMiddleware = createSagaMiddleware();
 return {

...createStore(rootReducer,window.__REDUX_DEVTOOLS_EXTENSION__&&window.__REDU
X_DEVTOOLS_EXTENSION__(),
     applyMiddleware(sagaMiddleware)),
   runSaga: sagaMiddleware.run(rootSaga),

  };
};

export default configureStore;
```

This is the code for creating the store for our web application. First we create our saga middleware, creation of redux store by passing ***rootReducer*** and ***sagaMiddleware*** to ***createStore***.

- sagas/newsSaga.js

```
import { put, call, all } from 'redux-saga/effects';
import { topNews, anyNews, sourceNews } from '../Api/api';
import * as types from '../constants/actionTypes';

export function* anyNewsSaga({ payload }) {
  try {
    const { news } = yield all({
      news: call(anyNews, payload),
    })
    yield [
      put({ type: types.SEARCH_ANY_NEWS_SUCCESS, news }),
    ]
  } catch (error) {}
```

Using sagas now the web application can access the News API data asynchronously which makes the accessibility for side effects much easier and better.

Finally we connect our React component to Redux store by updating ***index.js***.

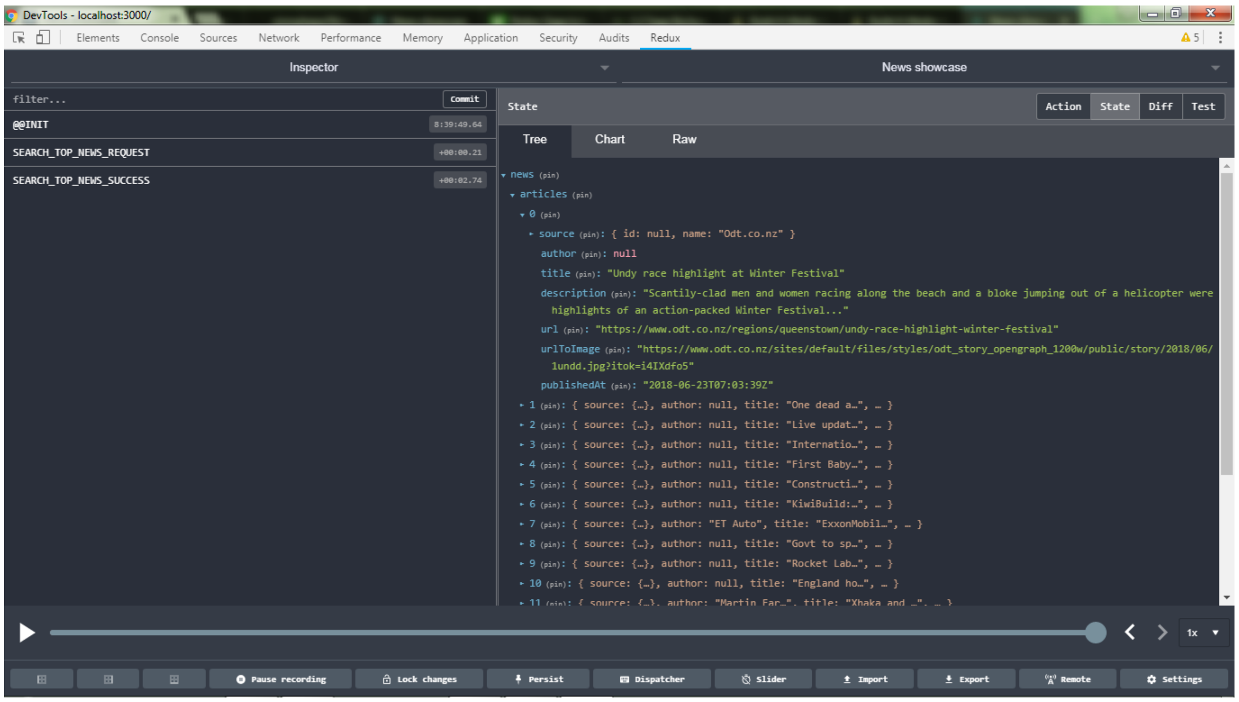
```
import ReactDOM from 'react-dom';
import React from 'react';
import { Router, browserHistory } from 'react-router';
import { Provider } from 'react-redux';
import configureStore from './store/configureStore';
import routes from './routes';

// Initialize store
const store = configureStore();

ReactDOM.render(
  <Provider store={store}>
    <Router history={browserHistory} routes={routes} />
  </Provider>, document.getElementById('root')
);
```

Here we initialize the store. The Provider component react-redux makes the store availability thorough out the system. If you look at our React components with use of connect now we can access our state.

Now start the application by typing npm start. If you install **Redux DevTool extenstion** in the browser you can see the whole store like as below image.



```
@@INIT                          8:39:49.64
SEARCH_TOP_NEWS_REQUEST         +00:00.21
SEARCH_TOP_NEWS_SUCCESS         +00:02.74
```

```
▾ news (pin)
 ▾ articles (pin)
  ▾ 0 (pin)
   ▸ source (pin): { id: null, name: "Odt.co.nz" }
    author (pin): null
    title (pin): "Undy race highlight at Winter Festival"
    description (pin): "Scantily-clad men and women racing along the beach and a bloke jumping out of a helicopter were
      highlights of an action-packed Winter Festival..."
    url (pin): "https://www.odt.co.nz/regions/queenstown/undy-race-highlight-winter-festival"
    urlToImage (pin): "https://www.odt.co.nz/sites/default/files/styles/odt_story_opengraph_1200w/public/story/2018/06/
      1undd.jpg?itok=i4IXdfo5"
    publishedAt (pin): "2018-06-23T07:03:39Z"
  ▸ 1 (pin): { source: {…}, author: null, title: "One dead a…", … }
  ▸ 2 (pin): { source: {…}, author: null, title: "Live updat…", … }
  ▸ 3 (pin): { source: {…}, author: null, title: "Internatio…", … }
  ▸ 4 (pin): { source: {…}, author: null, title: "First Baby…", … }
  ▸ 5 (pin): { source: {…}, author: null, title: "Constructi…", … }
  ▸ 6 (pin): { source: {…}, author: null, title: "KiwiBuild:…", … }
  ▸ 7 (pin): { source: {…}, author: "ET Auto", title: "ExxonMobil…", … }
  ▸ 8 (pin): { source: {…}, author: null, title: "Govt to sp…", … }
  ▸ 9 (pin): { source: {…}, author: null, title: "Rocket Lab…", … }
  ▸ 10 (pin): { source: {…}, author: null, title: "England ho…", … }
  ▸ 11 (pin): { source: {…}, author: "Martin Far…", title: "Xhaka and …", … }
```

4. Deploy to Heroku

Action creators are functions that returns plain JavaScript object of action type and an optional payload. These are pure functions.

Once the application is working fine locally, I need to host it in a server; heroku free plan was the ideal choice.

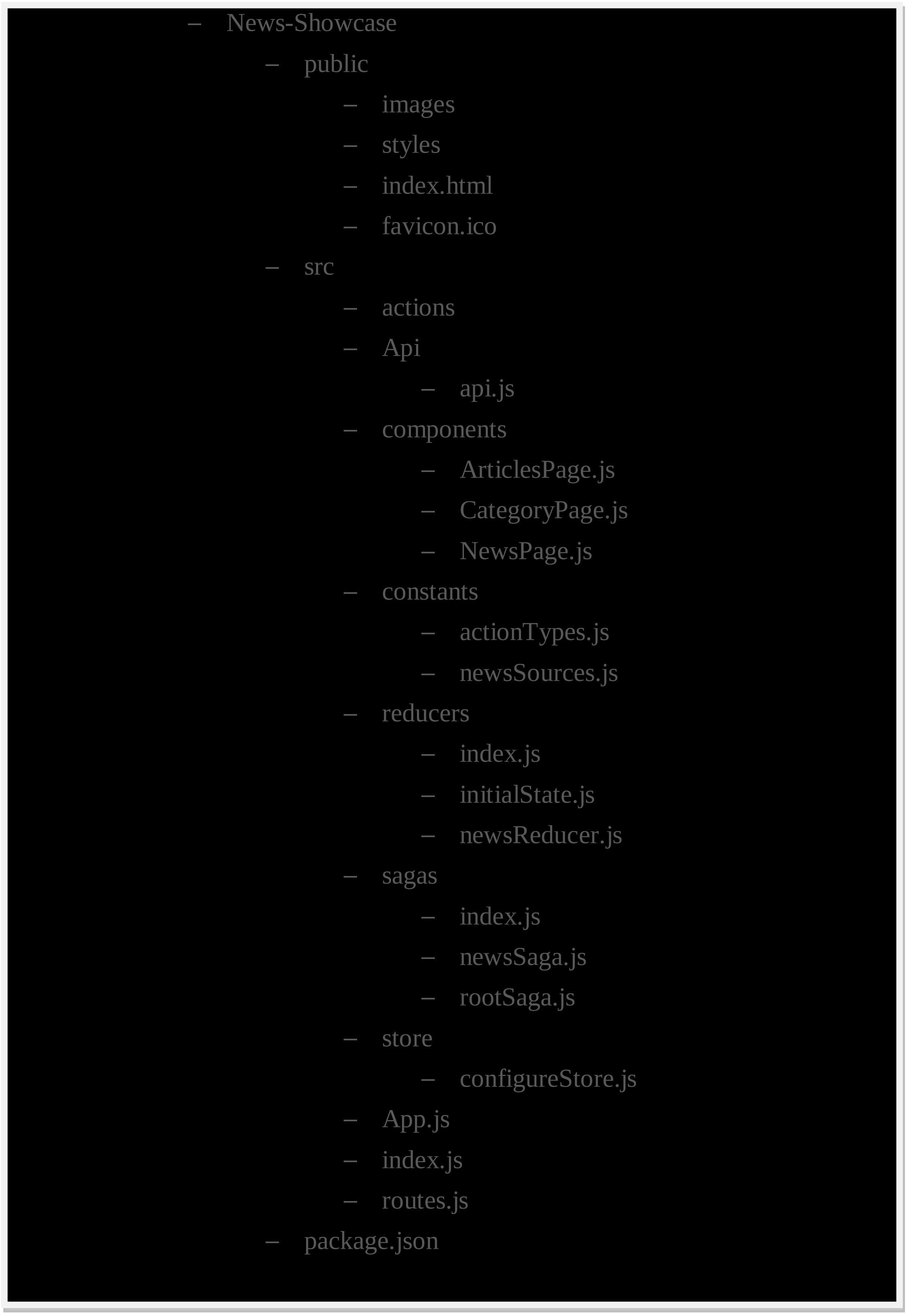
I used create-react-app's build script to bundle the application using below steps.

- Run ***npm run build*** - This will create a build folder in our project directory.
- Add ***"deploy": "npm install -g pushstate-server && pushstate-server build"*** to **package.json**
- create **Procfile** in project's root directory and add***: web: npm run deploy***
- Then complete the below commands

```
$ heroku login # Enter your credentials as it prompts you
$ heroku create <app name> # If  you don't specify a name, Heroku creates one
for you.
$ git add --all && git commit -m "Add comments"  # Add and commit your
changes.
$ git push heroku master # Deploy to Heroku
$ heroku ps:scale web=1 # Initialize one instance of your app
$ heroku open # Open your app in your default browse
```

Project File Structure:

The final structure of the project should look like below.



```
– News-Showcase
    – public
        – images
        – styles
        – index.html
        – favicon.ico
    – src
        – actions
        – Api
            – api.js
        – components
            – ArticlesPage.js
            – CategoryPage.js
            – NewsPage.js
        – constants
            – actionTypes.js
            – newsSources.js
        – reducers
            – index.js
            – initialState.js
            – newsReducer.js
        – sagas
            – index.js
            – newsSaga.js
            – rootSaga.js
        – store
            – configureStore.js
        – App.js
        – index.js
        – routes.js
    – package.json
```

Problems Faced

When it comes to problems while working with React, I found few issues as mentioned below.

- Deprecation of code – i.e. `React.PropTypes`
  - As React is updating, there are many console warnings to clear for deprecated methods.
  - Always need to follow the accurate versions
- Confusion between functions and classes.
- Forgetting another application is running in the same port.
- Confusing curly braces {} with parenthesis ().
- To make setState() to work.
- Implementing masonary effect to the news feed and styling was difficult.
- Connecting store was advanced and time consuming.

Impact of ReactJS

Popularity of ReactJS is increasing rapidly day by day. Today many leading companies uses React for their busiest dynamic websites due to its increasing versatility which makes feature – rich design an affordable option for any size of business.

React JS is a JavaScript based front end library, where I believe its features mainly impact the Fullstatck developers in the industry.

Key feature is its ability to split pages into separate components which can be used multiple times. As a result of this it reduces the development time allowing to good architecture, capability to adapt to future change. This feature mainly impacts the developers to come out from the JavaScript libraries like JQuery.

React is rapidly acquiring a reputation for its flexibility and simplicity. As react has ability to store the current state of an application internally. Therefore this make the application only update the place which changed which provide the user better experience.

When discussing about impact of React, the other important factor is to look at is why developers love React. The main advantage is that the ability to see how individual component render and to edit the code within. The other main factor is reusability of the component results in less development time, and zero code repetition.

The fact that so many main internet players are tuning into React clearly shows the impact and it has a strong future. As it is open source developers starts to upload many new features to the community. Developers enjoy working with React as it gives the user the freedom to experiment, push limits, and produce feature-rich, robust and visually appealing products which I predict soon to become a industry standard for web and mobile application development.