

---

# Table of Contents

|  |           |
|--|-----------|
| Introduction   | 1.1       |
| Technology We Use  | 1.2       |
| Installation   | 1.3       |
| Folder Structure   | 1.4       |
| Isomorphic [ Dashboard ]   | 1.5       |
| Overview   | 1.5.1     |
| Components   | 1.5.2     |
| Isomorphic [ Hotel ]   | 1.6       |
| Data Provider  | 1.6.1     |
| Auth. Provider   | 1.6.2     |
| Components   | 1.6.3     |
| Routes   | 1.6.3.1   |
| Post Grid  | 1.6.3.2   |
| Google Map   | 1.6.3.3   |
| Map Display  | 1.6.3.3.1 |
| Map AutoComplete   | 1.6.3.3.2 |
| Map Location search  | 1.6.3.3.3 |
| Map InfoWindow   | 1.6.3.3.4 |
| Config   | 1.6.4     |
| Data Format  | 1.6.5     |
| FAQ  | 1.7       |
| How to use this template for simple project structure              | 1.7.1     |
| How to run boilerplate and remove code packages that I don't need? | 1.7.2     |

# Isomorphic - React Redux Admin Dashboard

A react-redux powered single page admin dashboard. Used progressive web application pattern, highly optimized for your next react application.

## Isomorphic Reloaded (isomorphic-v3)

We are calling this Isomorphic Reloaded because of this major update which include a Frontend Hotel site example which is also available both on [React](#) & [Next.js](#) version. Also the Project structure changed into [Monorepo](#) using [Lerna](#) and `Yarn Workspaces` .

### Demo Link

Isomorphic (Dashboard) : <https://isomorphic.redq.io>

Isomorphic (Hotel) : <https://isomorphic-hotel.firebaseio.com>

## Monorepo

The idea behind a monorepo is to store all code in a single version control system (VCS) repository. The alternative, of course, is to store code split into many different VCS repositories, usually on a service/application/library basis.

Welcome to Isomorphic new version Built on mono-repo structure. Check the below link if you want to learn more about yarn workspaces.

1. [Introducing-workspaces/](#)
2. [Workspaces.](#)

### Why we use monorepo

1. To store all code in a single version control system (VCS) repository.
2. **Easier collaboration and code sharing**
3. Code refactors are easy / atomic commits

## Technology Credits

### *For Isomorphic [ Dashboard ]*

- Create React App
- React
- Redux
- Redux-Saga
- React Router 5
- Ant Design
- Google Map
- React Big Calendar
- React Flip Move
- React Google Charts
- Recharts
- React Vis
- React Chart 2

- React Trend
- Echart
- React Grid Layout
- Firebase CRUD
- Authentication Firebase
- Authentication Auth0
- Algolia Search .... etc

### ***For Isomorphic [ Hotel ]***

- Create React App
- React
- React Hooks
- Context API
- Ant Design
- React Router 5
- Formik
- Google Map .... etc

### Installation Process

- Install node & npm
- Install yarn
- Install packages & dependencies
- Yarn Build
- Yarn Start

Isomorphic is based on Create React App, It would be better if you can check their website too. There are a lot of tricks that can help your app, like API connection, Deployment etc.

<https://create-react-app.dev>

### Installing Node & NPM:

To work with `Isomorphic` the first thing you need is to have stable `Node` version installed on your system. To make sure you have already Node.js installed on your system you may follow the below instructions :-

As Node will make sure you have node and npm commands available via command line, just run the below command on your terminal.

```
node -v
```

```
npm -v
```

Note that if you find the npm version less than 5.0.0 you need to update it to the latest version using the below command. you may need to use `sudo` to grant permission

```
npm install npm@latest -g
```

or

```
sudo npm install npm@latest -g
```

### Installing YARN:

You will need to Install [Yarn](#) for the Fast, Reliable, and Secure Dependency Management. Before you start using [Yarn](#), you'll first need to install it on your system. And to make sure it running on your system with latest version run the below command.

```
yarn --version
```

### Installing Packages & Dependencies :

This Product is based on monorepo structure. To provide easy access to different item we provide some command line scripts to begin with -

```
$ yarn
```

**NB: make sure you use yarn for installing packages, dependencies and running script**

### Run the project in dev mode:

To run all the item at same time :

```
$ yarn start:all
```

Which is start development version of every item on different port :

```
hotel: at localhost:3001  
isomorphic: at localhost:3002  
etc.
```

To Run Individual Item:(ex: hotel / isomorphic)

```
$ yarn start:hotel
```

```
$ yarn start:iso-cra
```

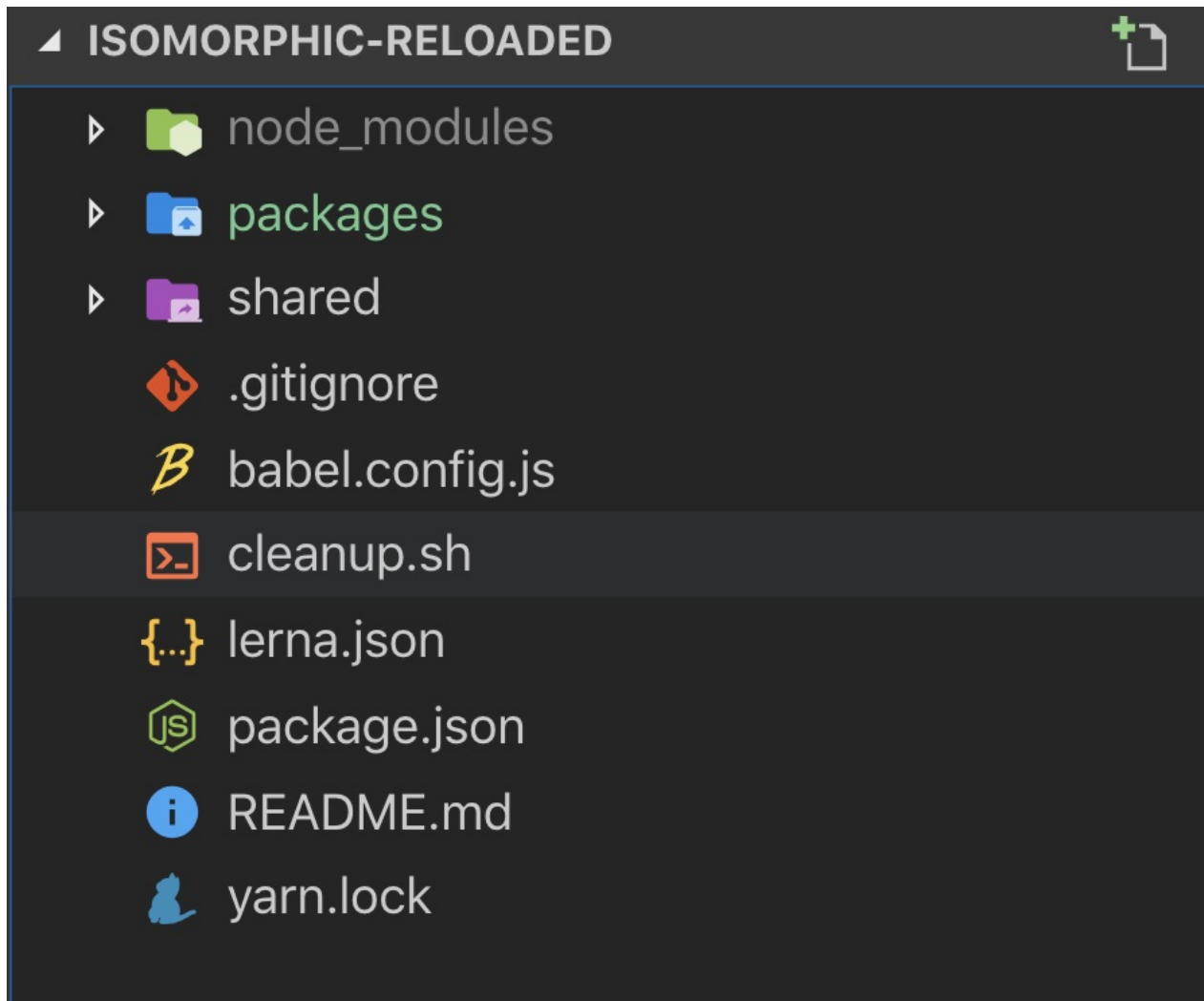
for others scripts just follow this

```
"scripts": {
  "clean": "lerna clean --yes && rimraf node_modules",
  "clean:build": "lerna exec -- rimraf \"{.next,dist,out,build,.docz}\"",
  "start:all": "lerna run --parallel start",
  "start:iso-cra": "yarn workspace isomorphic run start",
  "start:iso-next": "yarn workspace isomorphicnext run dev",
  "start:iso-servers": "yarn workspace isomorphic-servers run start",
  "start:iso-boilerplate": "yarn workspace isomorphic-boilerplate run start",
  "start:iso-gql": "lerna run --parallel start:gql",
  "start:hotel": "yarn workspace hotel run start",
  "start:hotel-next": "yarn workspace hotel-next run dev",
  "build:iso-cra": "yarn workspace isomorphic run build",
  "build:iso-next": "yarn workspace isomorphicnext run build",
  "build:hotel": "yarn workspace hotel run build",
  "build:hotel-next": "yarn workspace hotel-next run build",
  "serve:iso-cra": "yarn workspace isomorphic run serve",
  "serve:iso-next": "yarn workspace isomorphicnext run start",
  "serve:hotel": "yarn workspace hotel run serve",
  "serve:hotel-next": "yarn workspace hotel-next run start"
}
```

NB: We haven't included **packages/isomorphic-boilerplate-single** here because **Isomorphic-boilerplate-single** is for the user who don't want to use the **monorepo structure** and want a simplified structure. so you have to run **Isomorphic-boilerplate-single** by going inside **packages/isomorphic-boilerplate-single** directory. to know more about this check our below section <https://redq.gitbooks.io/isomorphic-reloaded/content/isomorphic-dashboard/overview.html>

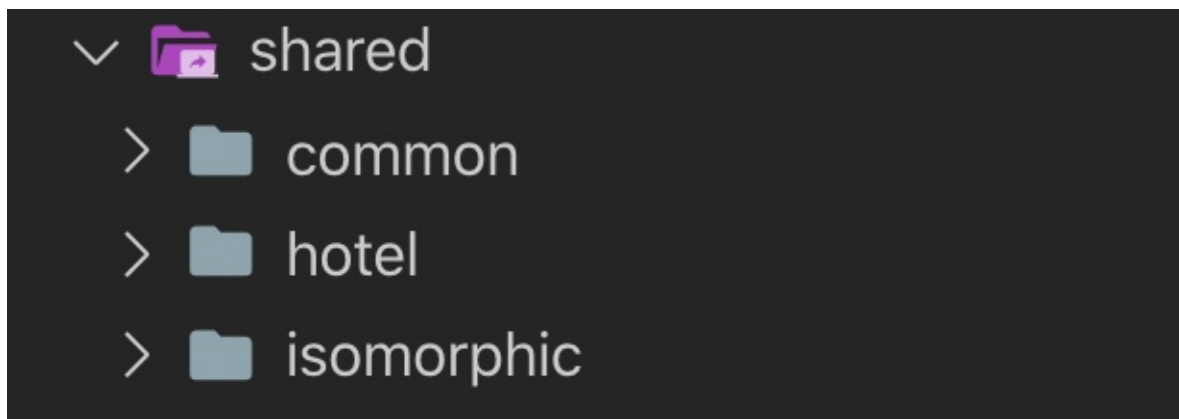
## Folder Structure: (Monorepo Structure using lerna and yarn workspace).

### Top-Level Folders:

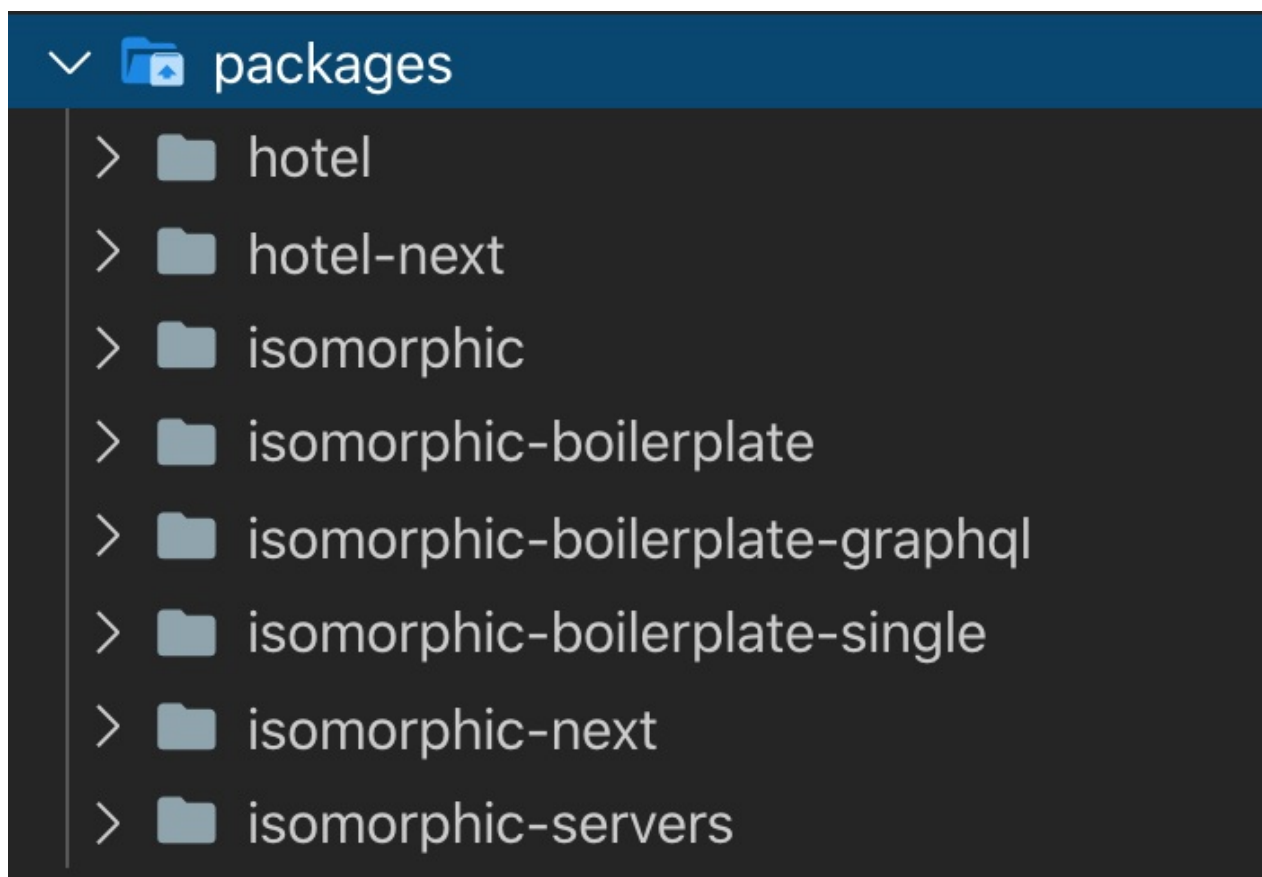


**shared: Contain All the Reuse able Components and Assets.**

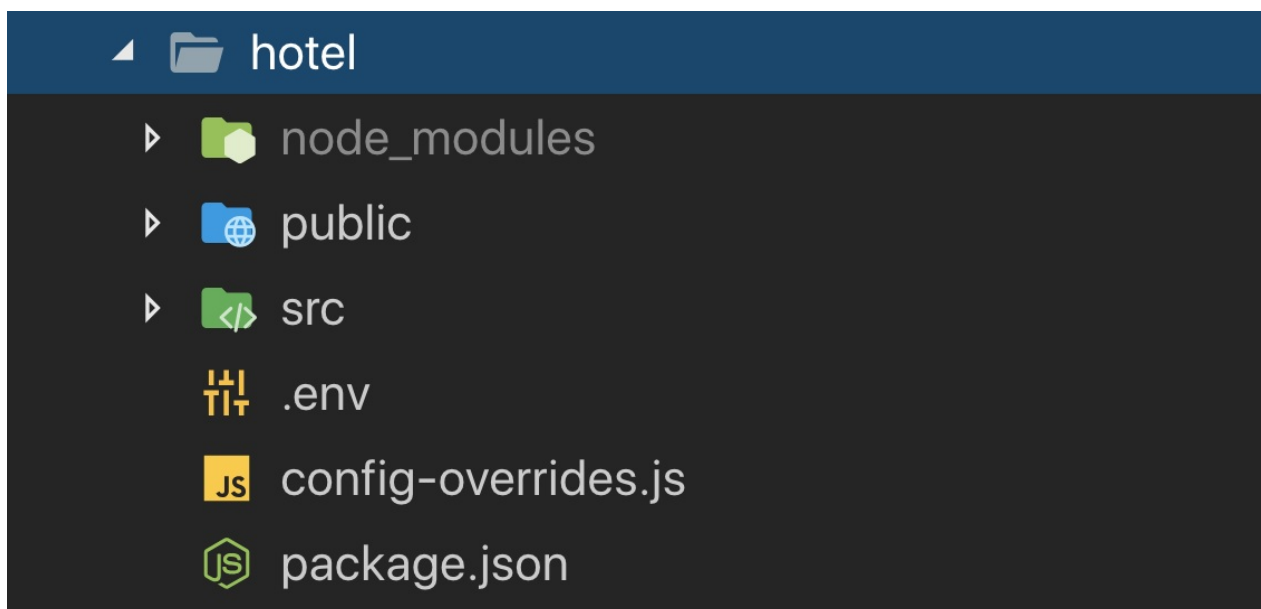




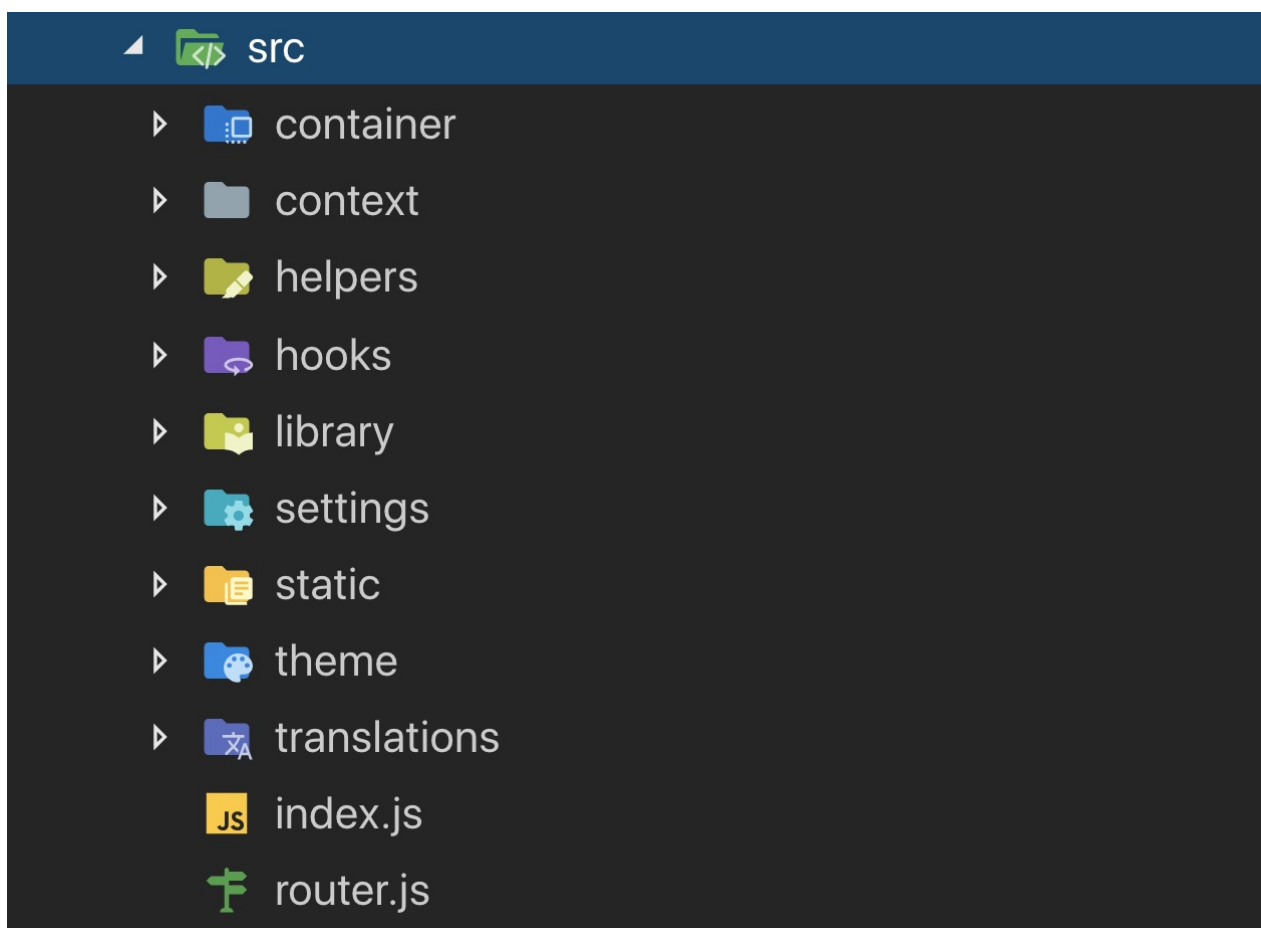
## packages:



## Project within packages (hotel):



## Project's src:



# Isomorphic [ Dashboard ]

In This section We are giving some hint where to start from depending on your purpose and some basic things of Isomorphic [ Dashboard ] that you can be benefited with.

**| So let's get started ...**

# Overview

Isomorphic Comes With multiple boilerplate and configurations for different use cases.

1. isomorphic (build with custom CRA)
2. isomorphic-next (SSR supported by next.js)
3. isomorphic-boilerplate ( if Anyone want to use with monorepo and custom CRA )
4. isomorphic-boilerplate-single (If Anyone want to use isomorphic without monorepo)
5. isomorphic-boilerplate-graphql (CRA with GraphQL server and client)
6. isomorphic-servers (example: of different backend integration)

## How we utilize monorepo structure

By using monorepo we simplified `import`

```
import Component from '../../../components/Component'
import ContainerComponent from '../../../containers/ContainerComponent'

to

import Component from '@iso/components/Component'
import ContainerComponent from '@iso/containers/ContainerComponent'
```

Reuse `components` , `containers` , `redux` , `assets` , `config` between all our `packages`. By moving them in `shared/isomorphic` folder and setup with monorepo. to import things from there you just prefix with `@iso/folderName`

We also move common things like library and ui within `shared/common`

To use them you just need to import things from `shared/common/library` , `shared/common/uiview` by using `@iso/lib` and `@iso/ui`

## If you don't want to use monorepo

then you need to move all the folders and component from `shared/isomorphic` and `shared/common` within your project `src` folder and use our provided config in `isomorphic-boilerplate-single` package.

see example: `isomorphic-boilerplate-single`.

**For the Isomorphic [ dashboard ] please follow up this link.**

Link: <https://redq.gitbooks.io/isomorphic/content/>

You can skip the zip overview section as from **Isomorphic Reloaded** version we are using Monorepo structure. To know about structure check the above folder structure section.

**In This section We are giving some hint where to start from depending on your purpose and some basic things of Isomorphic [ Hotel ] that you can be benefited with.**

**So let's get started ...**

## Data Provider

**To get your require data from a rest end-point use useDataApi hook as**

```
import useDataApi from '../hooks/useDataApi';

const Posts = () => {
  const { data, error, loading } = useDataApi('url');
  if (!data || loading) {
    return <div>Loading...</div>;
  };
  if (error) {
    return <div>Error! {error.message}</div>;
  };

  return (
    <ul>
      {data.posts.map(post => (
        <li key={post.id}>{post.title}</li>
      ))}
    </ul>
  );
};
```

this hook also accept limit as second parameter and many more you can add into it according to your requirements. as we provide some .

```
import { useState, useReducer, useEffect } from 'react';

function sleep(time) {
  return new Promise(resolve => setTimeout(resolve, time));
}

async function SuperFetch(
```



```
    url,
    method = 'GET',
    headers = {
      'Content-type': 'application/x-www-form-urlencoded; charset=
UTF-8',
    },
    body = {}
  ) {
    await sleep(2000); // demo purpose only
    let options = {
      method,
      headers,
    };
    if (method === 'POST' || method === 'PUT') options = { ...opti
ons, body };

    // authentication
    // we will had custom headers here.

    return fetch(url, options)
      .then(res => {
        return Promise.resolve(res.json());
      })
      .catch(error => Promise.reject(error));
  }

function dataFetchReducer(state, action) {
  switch (action.type) {
    case 'FETCH_INIT':
      return {
        ...state,
        loading: true,
        error: false,
      };
    case 'FETCH_SUCCESS':
      return {
        ...state,
        data: action.payload.slice(0, state.limit),
        total: action.payload,
        loading: false,
```

```
        error: false,
      };
    case 'FETCH_FAILURE':
      return {
        ...state,
        loading: false,
        error: true,
      };
    case 'LOAD_MORE':
      return {
        ...state,
        data: [
          ...state.data,
          ...state.total.slice(
            state.data.length,
            state.data.length + state.limit
          ),
        ],
        loading: false,
        error: false,
      };
    default:
      throw new Error();
  }
}

const useDataApi = (initialUrl, limit = 12, initialData = []) =>
{
  const [url, setUrl] = useState(initialUrl);

  const [state, dispatch] = useReducer(dataFetchReducer, {
    loading: false,
    error: false,
    data: initialData,
    total: initialData,
    limit: limit,
  });

  useEffect(() => {
    let didCancel = false;
```

```
const fetchData = async () => {
  dispatch({ type: 'FETCH_INIT' });

  try {
    const result = await SuperFetch(url);
    if (!didCancel) {
      dispatch({ type: 'FETCH_SUCCESS', payload: result });
    }
  } catch (error) {
    if (!didCancel) {
      dispatch({ type: 'FETCH_FAILURE' });
    }
  }
};

fetchData();

return () => {
  didCancel = true;
};
}, [url]);
const loadMoreData = () => {
  // dispatch({ type: 'FETCH_INIT' });
  dispatch({ type: 'LOAD_MORE' });
};
const doFetch = url => {
  setUrl(url);
};

return { ...state, doFetch, loadMoreData };
};

export default useDataApi;
```

# Authentication Provider

In this chapter the authentication process with pseudocode will be demoed here.

At first create a react context for the authentication and export it as we will use the AuthContext in multiple places inside the containers, components.

```
export const AuthContext = React.createContext();
```

Then we will build the AuthProvider component.

```
const fakeUserData = {
  id: 1,
  name: 'Jhon Doe',
  avatar: '',
  roles: ['USER', 'ADMIN'],
};

/**
 * We have used Fake JWT token from "jwt.io"
 * This is a sample token to show auth is working
 * Your token will come from your server when user successfully
  loggedIn.
 */

const fakeToken = 'FAKE-JWT-TOKEN-CODE';

const addItem = (key, value = '') => {};

const clearItem = key => {};

const isValidToken = () => {};

const AuthProvider = props => {
  const [loggedIn, setLoggedIn] = useState(isValidToken());
  const [user, setUser] = useState(null);
  const [token, setToken] = useState(null);
```

```
const signIn = params => {}
const signUp = params => {}
/**
 * For 3rd-party Authentication [e.g. Auth0, firebase, AWS etc
]
 *
 */
const tokenAuth = (token, user = {}) => {}
const forgetPass = params => {}
const changePass = params => {}
const logOut = () => {}

return (
  <AuthContext.Provider
    value={{
      loggedIn,
      logOut,
      signIn,
      signUp,
      forgetPass,
      changePass,
      tokenAuth,
    }}
  >
    <{props.children}</>
  </AuthContext.Provider>
);
};

export default AuthProvider;
```

In the code mentioned above, the AuthProvider is build up. All the authentication related functions *loggedIn*, *logOut*, *signIn* ,*signUp* , *forgetPass* , *changePass*, *tokenAuth* are passing through the AuthContext.

---

## \*\*\*\*\* Example Usages \*\*\*\*\*

***How to use AuthContext in Login component ??***

Suppose you need to create a Login page for the users. But how to use the AuthProvider??

1. At create a login form wrapper component using Formik via the initial values & validation schema.

```
// pseudocode

import { Formik } from 'formik';
import * as Yup from 'yup';

const initialValues = {
  email: '',
  password: '',
  rememberMe: false,
};

const getLoginValidationSchema = () => {
  return Yup.object().shape({
    email: Yup.string()
      .email('Invalid email')
      .required('Email is Required!'),
    password: Yup.string()
      .min(6, 'Password has to be longer than 6 characters!')
      .max(20, 'Too Long!')
      .required('Password is required!'),
  });
};

export default () => {
  return (
    <Formik
      initialValues={initialValues}
      onSubmit={handleSubmit}
      render={RenderSignInForm}
      validationSchema={getLoginValidationSchema}
    />
  );
};
```

## 1. Then Create the login form

```
// pseudocode
```

```
const RenderBasicInfoForm = props => {
  const { values, submitCount, handleSubmit } = props;
  return (
    <FormWrapper>
      <Form onSubmit={handleSubmit}>
        <Field
          component={AntInput}
          name="email"
          type="text"
          size="large"
          placeholder="Email"
          submitCount={submitCount}
          value={values.email}
          hasFeedback
        />
        <Field
          component={AntInput}
          name="password"
          type="password"
          size="large"
          placeholder="Password"
          value={values.password}
          submitCount={submitCount}
          hasFeedback
        />
        <FieldWrapper className="field-container">
          <SwitchWrapper>
            <Field
              component={AntSwitch}
              name="rememberMe"
              submitCount={submitCount}
              value={values.rememberMe}
            />
            <Label>Remember Me</Label>
          </SwitchWrapper>
          <Link to={FORGET_PASSWORD_PAGE}>Forget Password ?</Link>
        </FieldWrapper>
      </Form>
    </FormWrapper>
  );
}
```



```
        <Button
          className="signin-btn"
          type="primary"
          htmlType="submit"
          size="large"
          style={{ width: '100%' }}
        >
          <MdLockOpen />
          Login
        </Button>
      </Form>
    </FormWrapper>
  );
};

export default RenderBasicInfoForm;
```

1. Your login form is created, now connect the sign in form wrapper component [on step 1] with the *AuthProvider*!

```
// pseudocode
```

```
import { AuthContext } from '../..context/AuthProvider';
import { Formik } from 'formik';
import * as Yup from 'yup';
import { Redirect } from 'react-router-dom';

export default () => {
  const { signIn, loggedIn } = useContext(AuthContext);
  if (loggedIn) return <Redirect to={{ pathname: '/' }} />;
  const handleSubmit = formProps => {
    signIn(formProps);
  };
  return (
    <Formik
      initialValues={initialValues}
      onSubmit={handleSubmit}
      render={RenderSignInForm}
      validationSchema={getLoginValidationSchema}
    />
  );
};
```

You can see that we are using ***useContext*** *\_hook* here. Using that hook we manipulate the values/functions from the **AuthContext**.

---

### ***How to use AuthProvider in where some component need to use this ?***

Suppose you need to display a post grid on a page for the user, where it's for the authenticate users only. Then wrap that post grid using the AuthProvider.

```
// pseudocode

import AuthProvider from 'AUTH-PROVIDER-IMPORT-PATH';

export default function DisplayPosts(props) {
  return (
    <DisplayPostsPage>
      <AuthProvider>
        .
        .
        . Your components will be here...
        .
      </AuthProvider>
    </DisplayPostsPage>
  );
}
```

That's the overall process how you can use the ***AuthProvider*** and ***AuthContext*** in isomorphic-hotel projects.

## **Components [ written by Neamat Mim ]**

## Route

```
import React, { useContext } from 'react';
import { Route, Redirect, Switch } from 'react-router-dom';
import Loadable from 'react-loadable';
import { AuthContext } from '../context/AuthProvider';
import Layout from '../container/Layout/Layout';
import {
  LOGIN_PAGE,
  REGISTRATION_PAGE,
  FORGET_PASSWORD_PAGE,
  HOME_PAGE,
  LISTING_POSTS_PAGE,
  SINGLE_POST_PAGE,
  ADD_HOTEL_PAGE,
  AGENT_PROFILE_PAGE,
  AGENT_ACCOUNT_SETTINGS_PAGE,
  PRIVACY_PAGE,
  PRICING_PLAN_PAGE,
} from '../settings/constant';

/**
 *
 * Public Routes
 */
const Loading = () => null;

const routes = [
  {
    path: HOME_PAGE,
    component: Loadable({
      loader: () =>
        import(/* webpackChunkName: "Home" */ '../container/Home/
index'),
      loading: Loading,
      modules: ['Home'],
    }),
  },
];
```

```
    )),
    exact: true,
  },
  {
    path: LOGIN_PAGE,
    component: Loadable({
      loader: () =>
        import(/* webpackChunkName: "SignIn" */ './container/SignIn/SignIn'),
      loading: Loading,
      modules: ['SignIn'],
    }),
  },
  {
    path: REGISTRATION_PAGE,
    component: Loadable({
      loader: () =>
        import(/* webpackChunkName: "SignUp" */ './container/SignUp/SignUp'),
      loading: Loading,
      modules: ['SignUp'],
    }),
  },
  {
    path: FORGET_PASSWORD_PAGE,
    component: Loadable({
      loader: () =>
        import(
          /* webpackChunkName: "ForgetPassWord" */ './container/ForgetPassWord/ForgetPassWord'
        ),
      loading: Loading,
      modules: ['ForgetPassWord'],
    }),
  },
  {
    path: `${SINGLE_POST_PAGE}/:slug`,
    component: Loadable({
      loader: () =>
        import(
```

```
        /* webpackChunkName: "SinglePageView" */ './container/
SinglePage/SinglePageView'
    ),
    loading: Loading,
    modules: ['SinglePageView'],
  )),
},
{
  path: LISTING_POSTS_PAGE,
  component: Loadable({
    loader: () =>
      import(/* webpackChunkName: "Listing" */ './container/Li
sting/Listing'),
    loading: Loading,
    modules: ['Listing'],
  )),
},
{
  path: AGENT_PROFILE_PAGE,
  component: Loadable({
    loader: () =>
      import(
        /* webpackChunkName: "AgentDetailsViewPage" */ './cont
ainer/Agent/AccountDetails/AgentDetailsViewPage'
      ),
    loading: Loading,
    modules: ['AgentDetailsViewPage'],
  )),
},
{
  path: PRIVACY_PAGE,
  component: Loadable({
    loader: () =>
      import(/* webpackChunkName: "privacy" */ './container/Pr
ivacy/Privacy'),
    loading: Loading,
    modules: ['Privacy'],
  )),
},
{
```

```
    path: PRICING_PLAN_PAGE,
    component: Loadable({
      loader: () =>
        import(/* webpackChunkName: "Pricing" */ './container/Pr
icing/Pricing'),
      loading: Loading,
      modules: ['Pricing'],
    }),
  },
];

/**
 *
 * Protected Route Component
 *
 */

const AddHotel = Loadable({
  loader: () =>
    import(
      /* webpackChunkName: "RenderListingForm" */ './container/L
isting/RenderListingForm'
    ),
  loading: Loading,
  modules: ['RenderListingForm'],
});

const AgentAccountSettingsPage = Loadable({
  loader: () =>
    import(
      /* webpackChunkName: "AgentAccountSettingsPage" */ './cont
ainer/Agent/AccountSettings/AgentAccountSettingsPage'
    ),
  loading: Loading,
  modules: ['AgentAccountSettingsPage'],
});

/**
 *
 * Not Found Route Component
```



```
*
*/

const NotFound = Loadable({
  loader: () =>
    import(/* webpackChunkName: "NotFound" */ './container/404/4
04'),
  loading: Loading,
  modules: ['NotFound'],
});

const ProtectedRoute = ({ component: Component, ...rest }) => {
  const { loggedIn } = useContext(AuthContext);
  return (
    <Route
      render={props =>
        loggedIn ? <Component {...props} /> : <Redirect to={LOGI
N_PAGE} />
      }
      {...rest}
    />
  );
};

/**
 *
 * Overall Router Component
 *
 */

const Routes = () => {
  return (
    <Layout>
      <Switch>
        {routes.map(({ path, component, exact = false }) => (
          <Route key={path} path={path} exact={exact} component=
{component} />
        ))}
        <ProtectedRoute path={ADD_HOTEL_PAGE} component={AddHote
l} />
      </Switch>
    </Layout>
  );
};
```

```
        <ProtectedRoute
          path={AGENT_ACCOUNT_SETTINGS_PAGE}
          component={AgentAccountSettingsPage}
        />
        <Route component={NotFound} />
      </Switch>
    </Layout>
  );
};

export default Routes;
```

## **Post Grid Component**

**[will be written by Neamat Mim or Idris]**

# Google Map

Google map component is divided in 6 major portions.

- Display Google map with single marker or marker cluster
- Map AutoComplete Search
- Map Location search [with Map View]
- Map Location Box
- Map Data Helper Function
- Map InfoWindow

For More help here is the link of the npm package which is used here.

Link : <https://www.npmjs.com/package/react-google-maps>

## Display Google map with single marker or marker cluster

Displaying a Google map is divided into two sections. One with the single map marker and the other with the marker clusters.

**Map** is the wrapper component where it take other map components as a children.

```
import Map from 'MAP-IMPORT-FILE-LOCATION-PATH';

<Map>
  .
  .
  . {children component}
  .
  .
</MAP>
```

Now, for displaying the there need to be import **<MapDataProcessing />** component with some mandatory props.

```
import Map, { MapDataProcessing } from 'MAP-IMPORT-FILE-LOCATION-PATH';

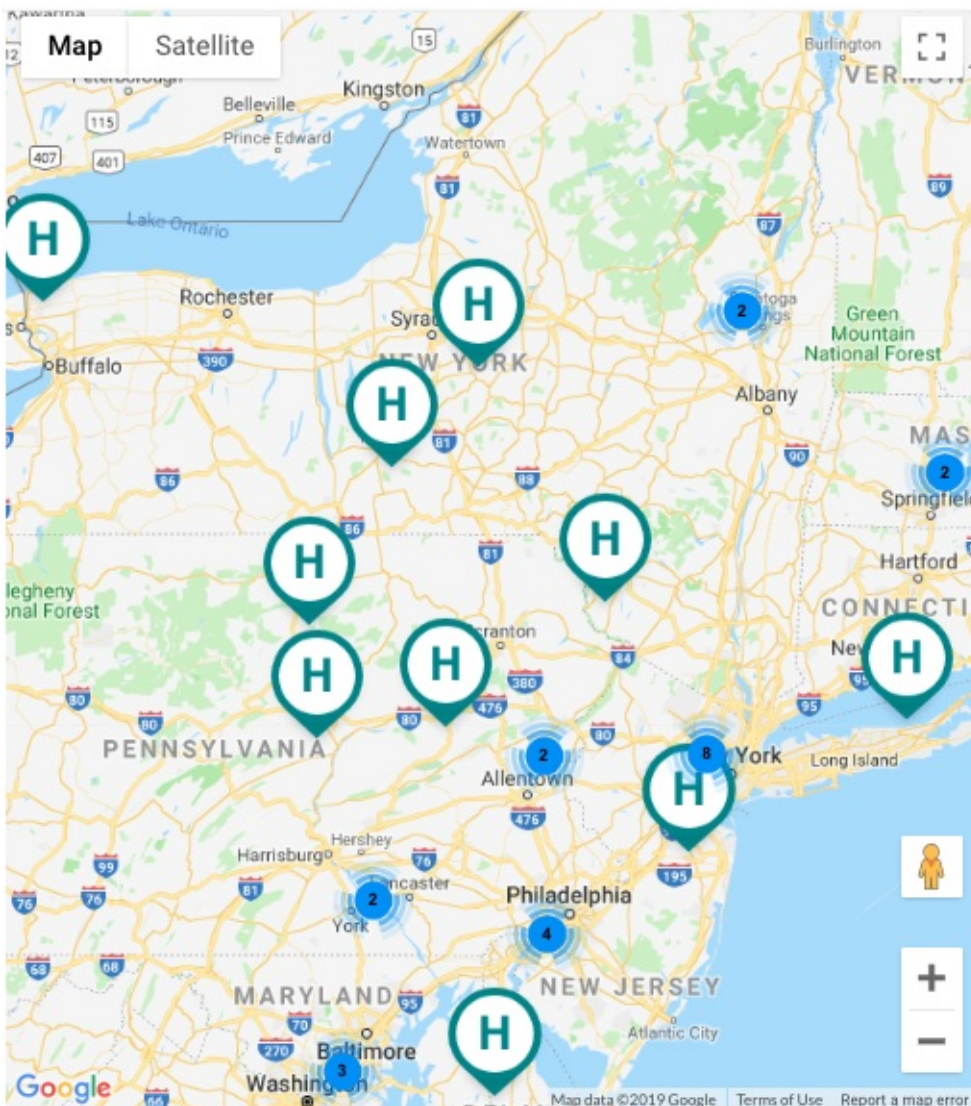
<Map>
  <MapDataProcessing location={MAP_DATA} multiple={false} />
</MAP>
```

| Props name | Data   |
|------------|--|
| location   | object   |
| multiple   | boolean { true for marker cluster, false for single marker } |

Display : For single map marker



Display : For marker cluster





## Map AutoComplete Search Component

Map auto complete component is designed for the locations choose options. This component receives one props for the data processing.


```
import MapAutoComplete from 'MAP_AUTO_COMPLETE_PATH'


const updatevalueFunc = (value) => {
  console.log(value)
}


<MapAutoComplete updatevalue={value => updatevalueFunc(value)} /
>
```


| Props       | data     |
|-------------|----------|
| updatevalue | function |


Display on view :


 New yd

 **New York** NY, USA

 **New York** USA

 **New York Penn Station** New York, NY,...

 **New York Hall Of Science** 111th Stre...

 **New York Public Library - Stephen ...**



## Map Location search

Map Location Search component is designed for the locations choose options but with the map view & marker drag n drop features.

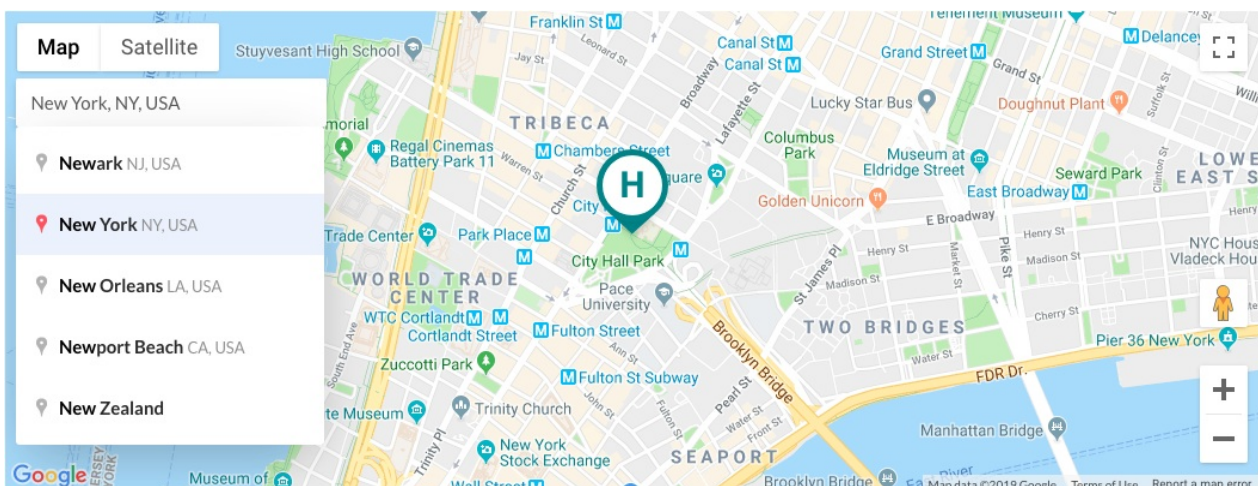
This component receives one props for the data processing.

```
import MapWithSearchBox from 'PATH';
import Map from 'PATH';

<Map>
  <MapWithSearchBox
    draggable={true}
    updatevalue={value => getUpdateValue(value)}
    {...props}
  />
</Map>
```

| Props       | Data     |
|-------------|----------|
| draggable   | boolean  |
| updatevalue | function |

for other props you can visit this link too. <https://tomchentw.github.io/react-google-maps/#searchbox>





## Map Info Window

Map info window component is designed for the displaying the map marker data on individual point.

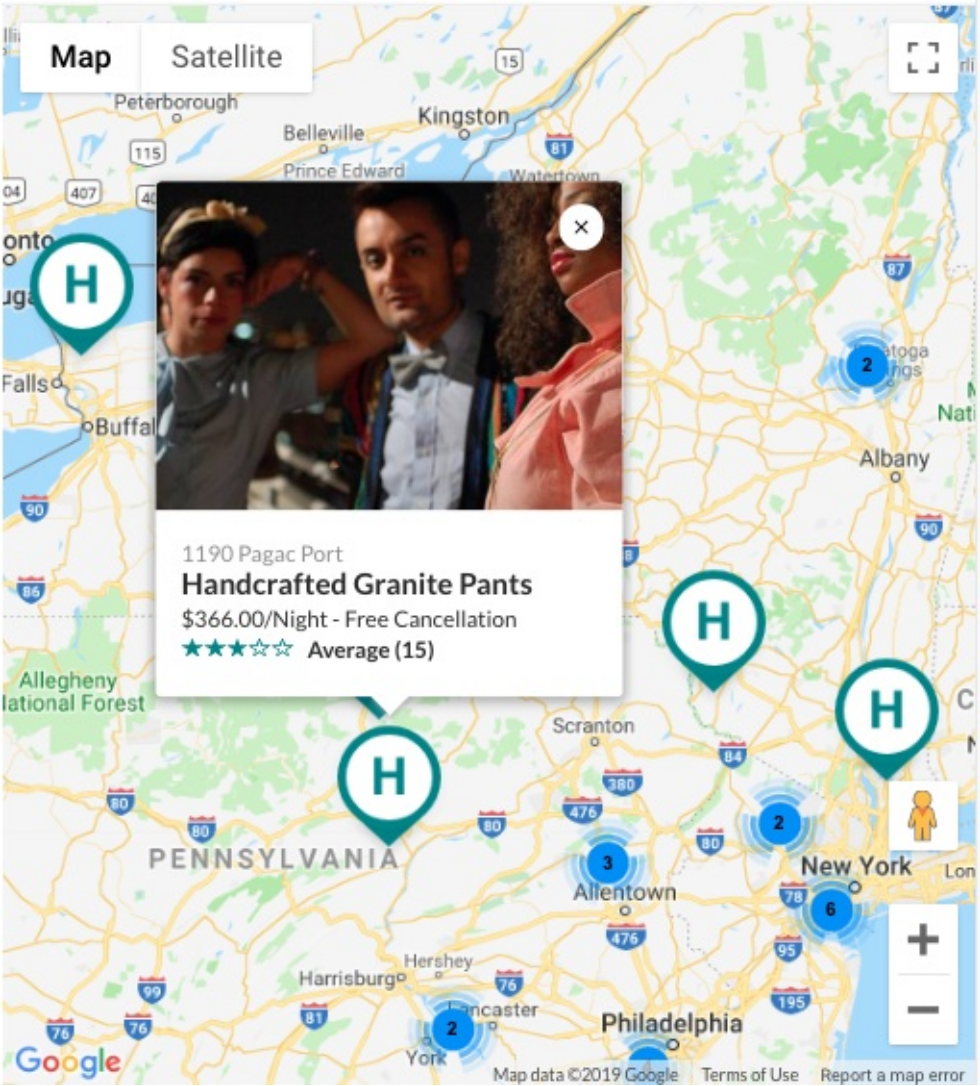
```
import HotelInfoWindow from 'PATH';

<HotelInfoWindow
  postData={POST_DATA}
  onCloseClick={() => {
    infoWindowToggle(SINGLE_POST.ID);
  }}
  {...props}
/>
```

| Props        | Date     |
|--------------|----------|
| postData     | object   |
| onCloseClick | function |

For other props you can check this link too <https://tomchentw.github.io/react-google-maps/#infowindow>

Display of hotel info window.



## Settings or Environment Params :

In this article we will show you the necessary settings or environment params which is used in the isomorphic [ hotel ] projects.

### Route Constants with pathname :

```
// General Page Constants
export const HOME_PAGE = '/';
export const AGENTS_PAGE = '/agents';

// Listing Single Page Constants
export const LISTING_POSTS_PAGE = '/listing';
export const SINGLE_POST_PAGE = '/post';

// Agent Profile Page Constants
export const AGENT_PROFILE_PAGE = '/profile';
export const AGENT_ACCOUNT_SETTINGS_PAGE = '/account-settings';
export const AGENT_PROFILE_EDIT_PAGE = '/edit';
export const AGENT_IMAGE_EDIT_PAGE = '/change-image';
export const AGENT_PASSWORD_CHANGE_PAGE = '/change-password';
export const AGENT_PROFILE_DELETE = '/delete';
export const AGENT_PROFILE_FAVOURITE = '/favourite-post';
export const AGENT_PROFILE_CONTACT = '/contact';
export const ADD_HOTEL_PAGE = '/add';

// Other Pages Constants
export const PRICING_PLAN_PAGE = '/pricing-plan';
export const PRIVACY_PAGE = '/privacy';

// Login & Registration Page Constants
export const LOGIN_PAGE = '/login';
export const REGISTRATION_PAGE = '/registration';
export const CHANGE_PASSWORD_PAGE = '/change-password';
export const FORGET_PASSWORD_PAGE = '/forget-password';
```

## **.env params :**

```
REACT_APP_GOOGLE_MAP_API_KEY = 'https://maps.googleapis.com/maps  
/api/js?v=3.exp&key=[YOUR_MAP_API_KEY]&libraries=geometry,drawin  
g,places',
```

## Data format for the Isomorphic [ Hotel ]

This chapter is very import. Here the focus is on the data format we have used for this Isomorphic [ Hotel ] project. Let's get started then.

### Agent Profile Data [ JSON format ]

```
{
  "id": INTEGER,
  "first_name": "",
  "last_name": "",
  "username": "",
  "password": "",
  "email": "",
  "cell_number": "",
  "profile_pic": {
    "id": INTEGER,
    "url": ""
  },
  "cover_pic": {
    "id": INTEGER,
    "url": ""
  },
  "date_of_birth": "",
  "gender": "",
  "content": "",
  "agent_location": {
    "lat": "",
    "lng": "",
    "formattedAddress": "",
    "zipcode": "",
    "city": "",
    "state_long": "",
    "state_short": "",
    "country_long": "",
    "country_short": ""
  },
}
```

```
"gallery": [  
  {  
    "id": INTEGER,  
    "url": ""  
  },  
  {  
    "id": INTEGER,  
    "url": ""  
  },  
  {  
    "id": INTEGER,  
    "url": ""  
  }  
],  
"social_profile": {  
  "facebook": "",  
  "twitter": "",  
  "linkedin": "",  
  "instagram": ""  
},  
"listed_post": [],  
"favourite_post": [],  
"createdAt": "",  
"updatedAt": ""  
}
```

---

## Hotel Post Data [ JSON format ] :

```
{  
  "id": INTEGER,  
  "agentId": INTEGER,  
  "title": "",  
  "slug": "",  
  "content": "",  
  "status": "",  
  "price": "",  
  "isNegotiable": BOOLEAN,  
  "propertyType": "",  
}
```



```
"condition": "",
"rating": FLOAT,
"ratingCount": INTEGER,
"contactNumber": "",
"termsAndCondition": "",
"amenities": [
  {
    "id": INTEGER,
    "guestRoom": INTEGER
  },
  {
    "id": INTEGER,
    "bedRoom": INTEGER
  },
  {
    "id": INTEGER,
    "wifiAvailability": BOOLEAN
  },
  {
    "id": INTEGER,
    "parkingAvailability": BOOLEAN
  },
  {
    "id": INTEGER,
    "poolAvailability": BOOLEAN
  },
  {
    "id": INTEGER,
    "airCondition": BOOLEAN
  },
  {
    "id": INTEGER,
    "extraBedFacility": BOOLEAN
  }
],
"image": {
  "id": INTEGER,
  "url": ""
},
"location": {
```

```
"id": INTEGER,
"lat": "",
"lng": "",
"formattedAddress": "",
"zipcode": "",
"city": "",
"state_long": "",
"state_short": "",
"country_long": "",
"country_short": ""
},
"gallery": [
  {
    "id": INTEGER,
    "url": ""
  },
  {
    "id": INTEGER,
    "url": ""
  },
  {
    "id": INTEGER,
    "url": ""
  }
],
"categories": [
  {
    "id": INTEGER,
    "slug": "",
    "name": "",
    "image": {
      "id": INTEGER,
      "url": ""
    }
  },
  {
    "id": INTEGER,
    "slug": "",
    "name": "",
    "image": {
```

```
        "id": INTEGER,  
        "url": ""  
    }  
}  
],  
"createdAt": "",  
"updatedAt": ""  
}
```

## FAQ

We have added several faq article in this section, please check this section before you open any support tickets.

## How to use this template for simple project structure:

This section is for those who don't want to use the monorepo structure or want to use it for a project where they have a very simple project structure.

I understand your concern. you don't want to use the monorepo structure because you want a very simplified to use it in your project and our current project structure may be little bit difficult for your to adjust, although we have informed it in our item description that we have added monorepo support.

so, here's what you can do it to make it work without using our monorepo. i guess your only concern is all that imports that needed to be changed. so to make it work at first remove all the packages/package that you don't need that is(isomorphic-next, hotel, hotel-next, servers etc) then there will be just one package that is isomorphic itself or just copy the package/isomorphic directory into your new create-react-app project. after that you need to move all the files and folder from shared/isomorphic(copy the files and folder from this directory) into packages/isonorphic/src/shared-isomorphic directory(you need to create shared-isomorphic directory here) or same thing apply if you are moving it to to your new create-react-app project. define absolute path following this link <https://create-react-app.dev/docs/importing-a-component/#absolute-imports> it will help you to replace all of the below path "@isomorphic/shared/isomorphic" with your new path that should be "isomorphic-shared/" and for the rest of the shared/package for example shared/Library and shared/UIView should follow the same process.

so to summarize it i will assume you will take everything into your new project (make sure you have all the package dependency available in your package.json file you can copy the packages/isomorphic/package.json file as well)

1. copy packages/isomorphic or just copy the files and folder from packages/isomorphic into your project.
2. copy shared/isomorphic contents in your new project inside src/folder-name(ex: isomorphic-shared)/
3. copy shared/UIView and shared/Library into your new project
4. define absolute path <https://create-react-app.dev/docs/importing-a-component/#absolute-imports>

5. now do a find and replace in your code editor(ex: vscode) the import name "`@isomorphic/shared/isomorphic`" with "`your-folder-name(ex:isomorphic-shared)/`" as for `UIView` and `Library` "`@isomorphic/shared/UIView`" with "`UIView/`" and "`Library`"

thats it, now you should now have a project without monorepo.

## How to run boilerplate and remove code package that I don't need?

if you have already followed this section <https://redq.gitbooks.io/isomorphic-reloaded/content/chapter1.html> then proceed.

you just run this command from the root,

```
yarn start:iso-boilerplate
```

if you don't want any other part of the packages you just delete it from `packages/{the-package-you-don't-want-to-use}`

but just fyi it will be better if you get a little knowledge about lerna and yarn workspace from our introduction page of the documentation

<https://redq.gitbooks.io/isomorphic-reloaded/content/>

also to know about the package and what you need you will get some information from here,

<https://redq.gitbooks.io/isomorphic-reloaded/content/folder-structure.html>

and from `shared/{some-common-component-library-packages}`

you can remove the part that you don't need but for now i will suggest you to keep it that way, once you know how does this work you can remove them easily.

also if you want you can read this part as well,

<https://redq.gitbooks.io/isomorphic-reloaded/content/how-to-use-this-template-for-simple-project-structure.html>