**Introduction**

**HOMMIAC APP** is an React admin t, which is based on the React 16.9+, Bootstrap 4+, Redux, build using create-react-app and ready to use REST API.

**HOMMIAC APP** use unique UI components, created special for React application, which allows it to be easily customize and build your own app quickly.

HOMMIAC APP built on **Redux** - it is state management for React applications. Store is a controlled state container designed to help write performant, consistent applications on top of React.

Also HOMMIAC APP uses React with typescript to help you maintain your project and provide static type analysis on compilation level.

All styles are fully develop with **SCSS** and easy to understand and customize.

Technologies

* React
* React redux
* React router dom
* React create app
* Bootstrap
* Npm
* Sass
* CSS3 Animations
* Typescript
* Google fonts

Core features

* Unlimited navbar colors
* 3 different layouts
* Data tables
* Vertical/horizontal navbar
* Fully Responsive Layout
* Light/dark sidebar
* Light/dark topbar
* Easy to customize
* 3 icon packs
* And many more…

All pages

* UI Kit
  + Components
    - Alerts
    - Autocompletes
    - Badges
    - Buttons
    - Cards
    - Checkboxes
    - Contacts
    - Inputs
    - Modal windows
    - Radio buttons
    - Ratings
    - Selects
    - Switchers
    - Textareas
    - Timelines
  + Icons
    - Icons options
    - Icofont
    - Antd icons
  + Typography
  + Tables
  + Forms
    - Form elements
    - Form layouts
    - Form validation
  + Charts
    - Recharts
    - Chart.js
    - React echarts
  + Maps
    - Google map
    - Vector map
    - World map
* Apps
  + Service pages
    - Invoices
    - Invoice
    - Pricing
    - Edit account
    - User profile
    - Events timeline
    - Events calendar
  + Sessions
    - Sign in
    - Sign up
    - 404
    - 500

**Getting started**

This template is built with *React* and requires *Node(v10.16.0+)* and *NPM(v6.4.1+)* to be installed.

Installing prerequisites

**Node.js and NPM:** You can download Node.js from <https://nodejs.org>. NPM comes bundled with Node.js

Installing npm packages

Before start you'll need to install required packages and dependencies. Open folder with theme in terminal and run npm install.

Then run npm start to start dev server. By default it will run at <http://localhost:3000/>.

Deployment

Before deploying application, code need to be minified and optimized. For these purposes you can use npm run build or if you using yarn yarn build. When completed code will be stored to build folder, and will be ready for deployment. For more information about deployment see [cra deployment guide](https://create-react-app.dev/docs/deployment)

**Customization**

Main app's settings

You can change main app's settings in src/redux/settings/DefaultSettings.ts.

Sass settings

All style files can be found at src/assets/sass. For changing styles of the app you can go to \_variables.scss,it contains all color schemas and main parameters of the app. Main scss file is located in src/assets/sass/style.scss.

Ant design theme

Also this template contain theme for ant design components. You can find it at config-override.js at the root of the project. There within modifyVars field you can change existing variables, or and new. For more info about customization check [official docs](https://ant.design/docs/react/customize-theme).

**Routing and main menu**

Routing

All routes located at routing folder, placed at root of the project. It already contain separate routes for different layouts: default routes for vertical and horizontal layout, errorRoutes for error layout and public routes for public layout.  
If you want add new route, select one of the defiened arrays, and add route there. Example:  
...  
{ path: '/dashboard', component: PageDashboard }  
...  
For adding new layout, just import layout component into and add it to routes array, and specify path on which it must be used, also specify children routes array that can be reachout within layout component. Example provided below:

Firstly create routers array, that you want to be bound to your layout:

const YOUR\_ROUTES: Routes = [

{ path: '404', component: PageNotFoundComponent},

{ path: '500', component: PageInternalErrorComponent}

];

And then add this array of routes into App.tsx Switch by passing this array into Routes helper component. Additionaly you can specify layout for this routes

import { Switch, Route } from 'react-router-dom';

<Switch>

*// Using routes withount layout*

<Routes routes={YOUR\_ROUTES} />

...

*// Adding layout to routes group*

<Route path='/public'>

<PublicLayout>

<Routes routes={YOUR\_ROUTES} layout='public'/>

</PublicLayout>

</Route>

...

*// Using multiple layouts with same routes*

<Route path='/horizontal'>

<HorizontalLayout>

<Routes routes={YOUR\_ROUTES} layout='horizontal'/>

</HorizontalLayout>

</Route>

<Route path='/vertical'>

<VerticalLayout>

<Routes routes={YOUR\_ROUTES} layout='vertical'/>

</VerticalLayout>

</Route>

</Switch>

Routes component is defined in App.tsx and not provided by react-router-dom

You can find more about routing and using react-router-dom at [official docs](https://reactrouter.com/web/guides/quick-start)

**Main menu and navigations**

Main menu configuration file located in public/data/menu.json. You can customize main menu from here.

Menu item interface can be found at src/app/interfaces/main-menu.ts

Main menu item contains next parameters:

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| **Title** | Title of menu item (parameter is required) |
| **Routing** | Routing path of item |
| **Icon** | Class name of items icon |
| **Active** | State of menu item |
| **Sub** | Array of subitems of the item |

Menu item with 3 sub items

{

"title": "Dashboard",

"icon" : "icofont-dashboard-web",

"sub" :

{

"title": "Dashboard 1",

"routing" : "dashboard1"

},

{

"title": "Dashboard 2",

"routing" : "dashboard2"

},

{

"title": "Dashboard 3",

"routing" : "dashboard3"

}

Menu item withount sub items

{

"title" : "Users",

"icon" : "icofont-ui-user",

"routing" : "users"

}

Also you can create your own link for naviagattion via NavLink component. It allows you to define an active class and styles according to router and works in a harmony with router.

import { NavLink } from "react-router-dom";

<NavLink to="/app/dashboards">Dashboards</NavLink>

**Redux**

Information shared across the app via Redux.

As an example, app settings, layout or page state.

This is shared via Redux. All the Redux related code is located under redux folder categorized by types: actions, reducers and types.

To adding new data to redux state you need to create new actions and reducers.

Also if you want use complex data typex, you will need to create related types to this reducer.

To use component with redux you need to wrap it with redux HOC connect that will pass redux state and dispatchers to component, or use redux hooks to dispatch actions and select data from store.

More info about redux hooks can be found at [Redux official docs](https://react-redux.js.org/api/hooks)

*// Sample action type*

export interface ResetSettingsAction {

type: typeof RESET\_SETTINGS;

}

*// Sample action*

export const resetSettings = (): SettingsTypes.ResetSettingsAction => ({

type: SettingsTypes.RESET\_SETTINGS

});

*// Sample reducer*

export const settingsReducer = (

state: IAppSettings = initialState,

action: SettingsActionTypes

): IAppSettings => {

switch (action.type) {

...other actions

case RESET\_SETTINGS:

return { ...initialState };

default:

return { ...state };

}

};

*// Sample mapStateToProps function*

const mapStateToProps = (state: AppState) => ({

settings: state.settings

});

*// Sample mapDispatchToProps function*

const mapDispatchToProps = dispatch => ({

onSettingsReset: () => dispatch('RESET\_SETTINGS')

});

export default connect(

mapStateToProps,

mapActionsToProps

)(MyComponent);

If you want pass additional props to redux component. You can pass them as normal props. Components will receive them as second argument of mapDispatchToProps or mapStateToProps functions. Example:

*// Pass props to redux component*

<Component data={someData} />

*// In the component*

const mapStateToProps = (state, ownProps) => {

return {

prop: ownProps.data

};

};

More can be found at react-redux [official documentation](https://react-redux.js.org/introduction/quick-start)

Pages

Pages state used to define breadcrumbs, title and menage loading neaded data for the page.

For adding page functionality to component you can use usePageData or useFetchPageData custom hooks

**usePageData**

usePageData hook accept object the implements IPageData interface and set passed data as current page state.  
The interface provided below

export interface IPageData {

title?: string;

loaded?: boolean;

breadcrumbs?: IBreadcrumb[];

fullFilled?: boolean;

}

export interface IBreadcrumb {

title: string;

route?: string;

}

**useFetchPageData**

useFetchPageData Used to obtain the necessary information and set the page status according to the retrieval process. Also this hoo will automatically set fullFilled state to true after date will be obtained.

useFetchPageData signature looks like this

function useFetchPageData(

url: string,

initialState: T = null,

callback?: (T) => any

): [T, (data: T) => void] {

...

}

url is url to perform GET request to

initialState is initialData that will be provided

callback is optional parametr that can be passed to perform function on succesfull data retrieval, retrieved data will be provided as argument to this function

Bellow provided demo of using this hooks for page with data fetching

const pageData: IPageData = {

title: 'Demo page',

fullFilled: false,

breadcrumbs: [

{

title: 'Home',

route: 'home'

},

{

title: 'Demo Page'

}

]

};

const dataUrl = '/api/data';

const initialState = [];

const DemopPage = () => {

usePageData(pageData);

const [data, setData] = useFetchPageData(

dataUrl,

initialState

);

return (

<>

{data}

</>

);

};

**Ui components**

This theme uses Ant Design components.

Before using ant components you need to import it to file where you want to use it from ant. Example:

import { Alert, Avatar, Button } from 'antd';

To find out more and for components documentation got to [official Ant Design documentation](https://ant.design/docs/react/getting-started)

**Icons**

This template uses antd icons v4, as this is recomended and reduces bundle size. More about icons you can find at [offical doc](https://ant.design/components/icon/" \l "List-of-icons)

But Antd Icons demo page uses compatible version of icons for demo purposes.  
If you do not need it you can run npm uninstall @ant-design/compatible or yarn remove @ant-design/compatible to remove compatible version of icons