**FRONT END**

**I. Instalasi**

- Visual studio code

*http://www.duniacara.com/2018/08/cara-menginstall-visual-studio-code-di-windows-10*

- Node.js dan npm

*https://www.taniarascia.com/how-to-install-and-use-node-js-and-npm-mac-and-windows/*

- Postman

*http://toolsqa.com/postman/download-and-install-postman/*

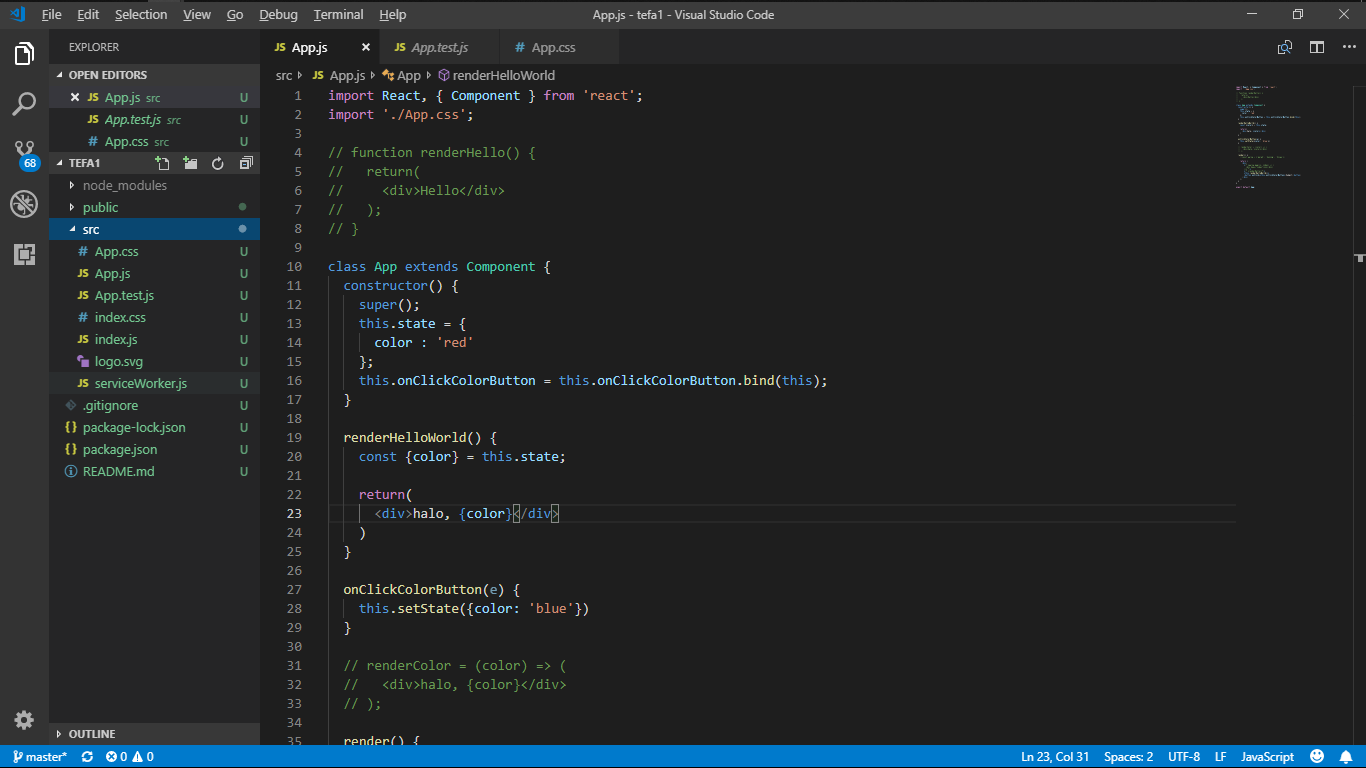
- React

*https://www.robinwieruch.de/react-js-windows-setup/*

- Git

*https://www.petanikode.com/git-install/*

**II. Tampilan Awal**



Setelah menginstall react.js dan membuatnya melalui cmd, maka akan secara default meng-create folder node modules, src, dan lainnya.

III. Function

1. Bukalah file App.js

2. Buatlah constructor tepat dibawah class App

constructor() {

super();

this.state = {

color : 'red'

};

this.onClickColorButton = this.onClickColorButton.bind(this);

}

3. Setelah itu buatlah function renderHelloWorld dan onClickColorButton

renderHelloWorld() {

const {color} = this.state;

return(

<div>halo, {color}</div>

)

}

onClickColorButton(e) {

this.setState({color: 'blue'})

}

4. Di bagian return, paggil renderHelloWorld dan juga buatlah button serta panggil function onClickColorButton

<div>

{this.renderHelloWorld()}

<button onClick={this.onClickColorButton}>Submit</button>

</div>

5. Berikut ini adalah program lengkapnya

import React, { Component } from 'react';

import './App.css';

class App extends Component {

constructor() {

super();

this.state = {

color : 'red'

};

this.onClickColorButton = this.onClickColorButton.bind(this);

}

renderHelloWorld() {

const {color} = this.state;

return(

<div>halo, {color}</div>

)

}

onClickColorButton(e) {

this.setState({color: 'blue'})

}

render() {

return (

<div>

{this.renderHelloWorld()}

<button onClick={this.onClickColorButton}>Submit</button>

</div>

);

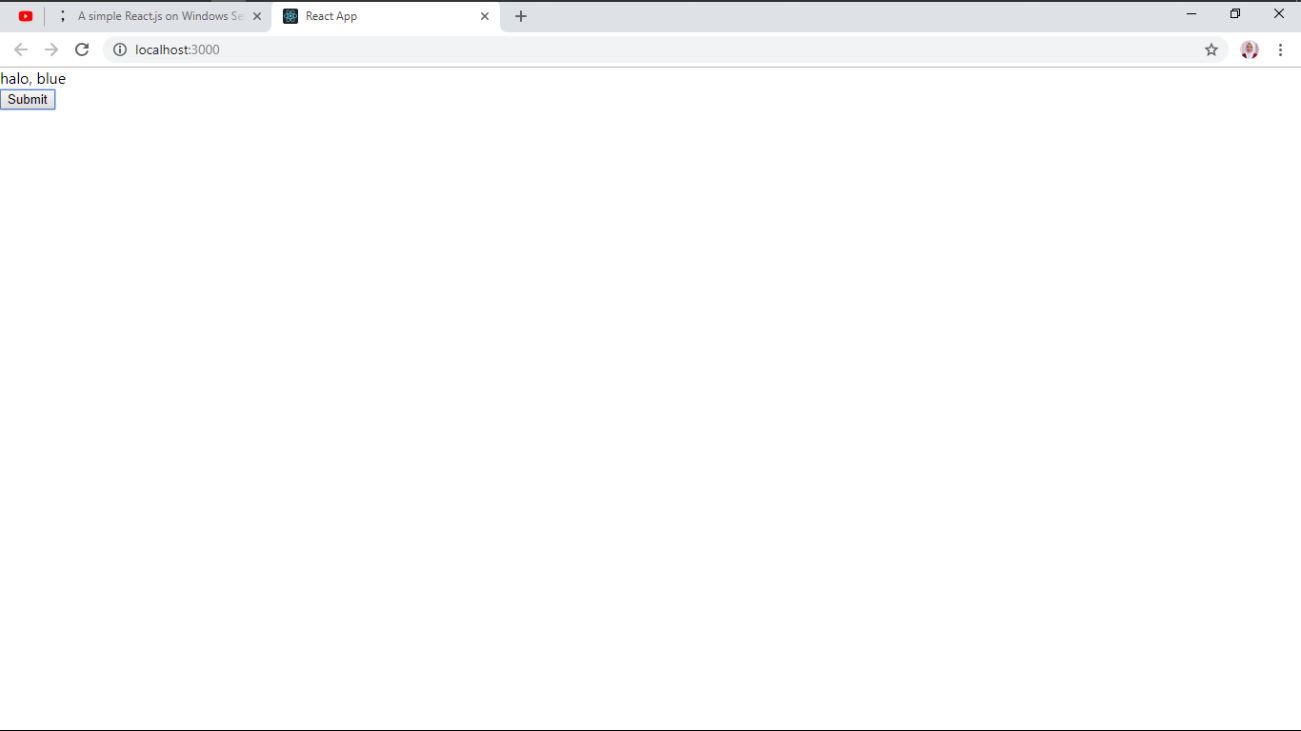
}

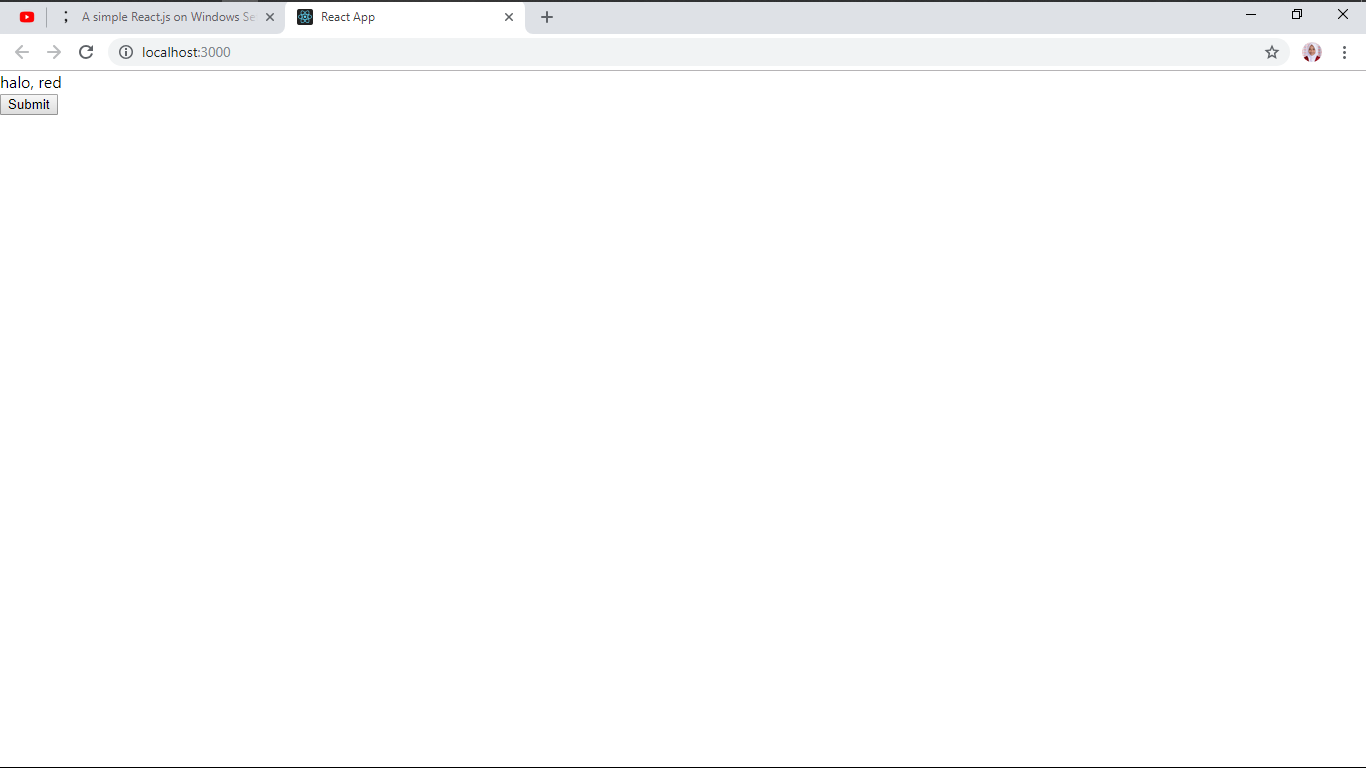
}

export default App;

6. Jalankan melalui terminal dengan mengetikkan “npm start”

7. Hasil kerja





**IV. Array**

1. Untuk mengambil data dari API, kita menggunakan AXIOS sebagai pihak ketiga. Sebelumnya install lah terlebih dahulu axios dengan mengetikkan perintah

npm install axios di terminal.

2. Berikut ini adalah contoh program array

import React from 'react';

import axios from 'axios';

import './GetDataCSS.css';

export default class GetData extends React.Component {

constructor() {

super();

this.state = {

data: [],

//data: "",

message: "",

isloading: true

}

this.componentDidMount = this.componentDidMount.bind(this);

}

componentDidMount() {

const self = this;

axios({

method: 'get',

url: 'http://ec2-13-251-45-154.ap-southeast-1.compute.amazonaws.com:8080/api/v1/insert-order-management?page=1&size=20',

headers: {

'Authorization': 'Basic dGVsa29tOmRhMWMyNWQ4LTM3YzgtNDFiMS1hZmUyLTQyZGQ0ODI1YmZlYQ=='

}

})

.then(function (response) {

self.setState({

data: response.data.data.result,

message: response.message

})

})

.catch(function (error) {

self.setState({ message: error })

console.log(error);

});

}

render() {

const { data } = self.state;

console.log(data);

return (

<div className="container\_get">

<div className="div\_table\_get">

<table border="1" className="table\_get">

<thead>

<tr>

<th>Email</th>

<th>Nama</th>

<th>Nama Project</th>

<th>Nama Unit</th>

</tr>

</thead>

<tbody>{this.state.data.map(function (data, key) {

return (

<tr key={key}>

<td>{data.first\_name}</td>

<td>{data.email}</td>

<td>{data.project\_name}</td>

<td>{data.unit\_name}</td>

</tr>

)

})}

</tbody>

</table>

</div>

</div>

);

}

}

**V. State Management**

Ini adalah contoh dari state

this.state = {

data: [],

message: "",

isloading: true

}

**VI. Props Management**

Perbedaan yang paling menonjol antara state dan props ialah: state hanya bisa dipanggil di file dimana ia dideklarasikan, sedangkan props, kita bisa memanggilnya di file lain. Berikut ini contoh memanggil props :

<h1>Hello, {this.props.username}</h1>

**VII. Testing Application**

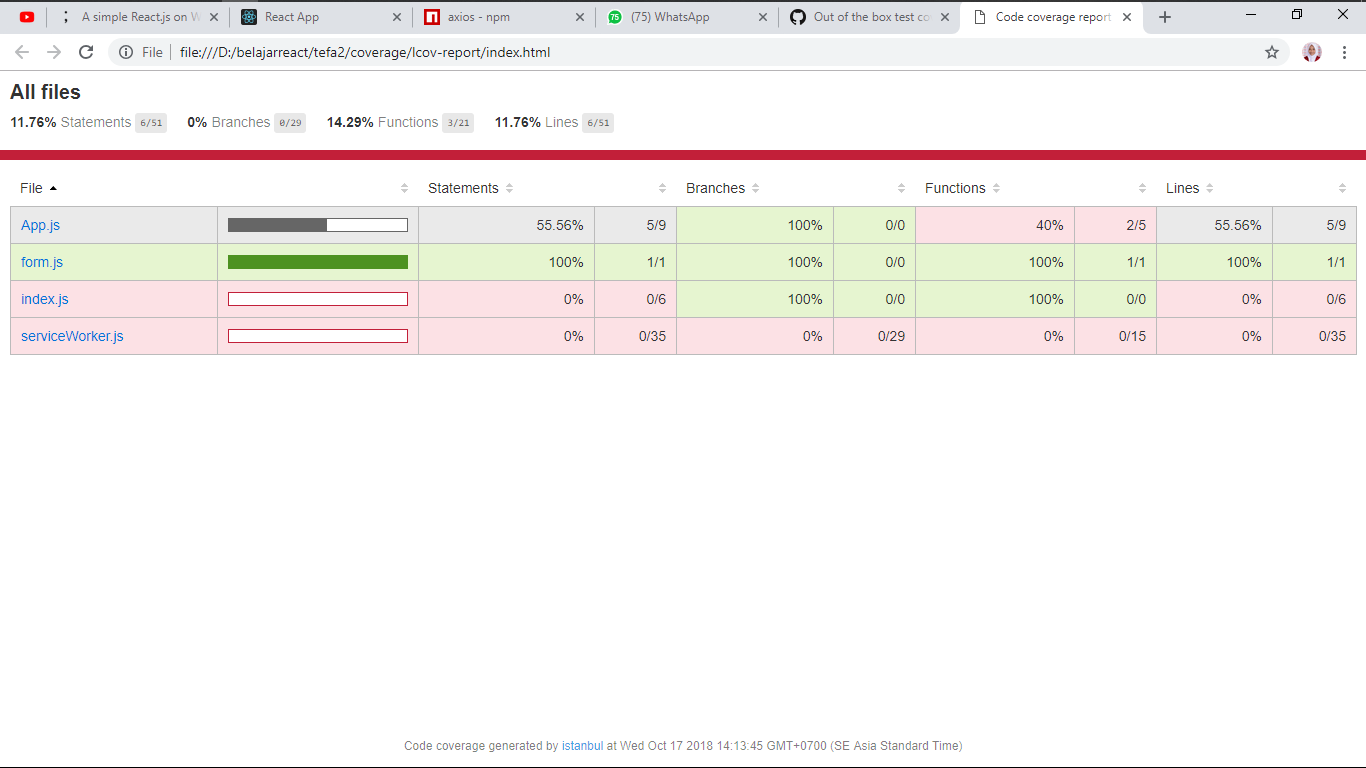
Untuk mengetahui seberapa maksimal file kita bekerja, maka dilakukan testing yang bisa kita lakukan dengan menggunakan bantuan Eslint ataupun Jest. Untuk mengetest program kita, ketikkan

Npm run test

Jika muncul tampilan seperti berikut / PASS, maka program kita dinyatakan lulus test atau bisa bekerja sebagaimana mestinya / tidak error. Atau kita juga bisa menggunakan perintah

npm test -- --coverage

Untuk membuat folder coverage. Folder ini bisa memudahkan kita dalam melihat hasil testing, kita dapat melihat seberapa persen program kita bekerja. Buka folder coverage yg auto tertambahkan, buka bagian index.html di browser



**VIII. Code Base**

Pada bagian ini, kita membagi tempat untuk membuat sesuatu. Kita membuat segala component di folder components yang disediakan

*src/components*. Setiap component harus memiliki 3 file, yaitu component.js, index.js, dan style.js. Index.js adalah file yang pertama kali dibaca pada setiap component. Index.js akan memanggil component.js dan style.js. Program berjalan pertama kali melalui index.js aplikasi

document.getElementById('app')

yang akan memanggil file App.js. Disini, program akan menjalankan pages home.

<Route exact path={ROUTES.HOME()} component={pages.Home} />

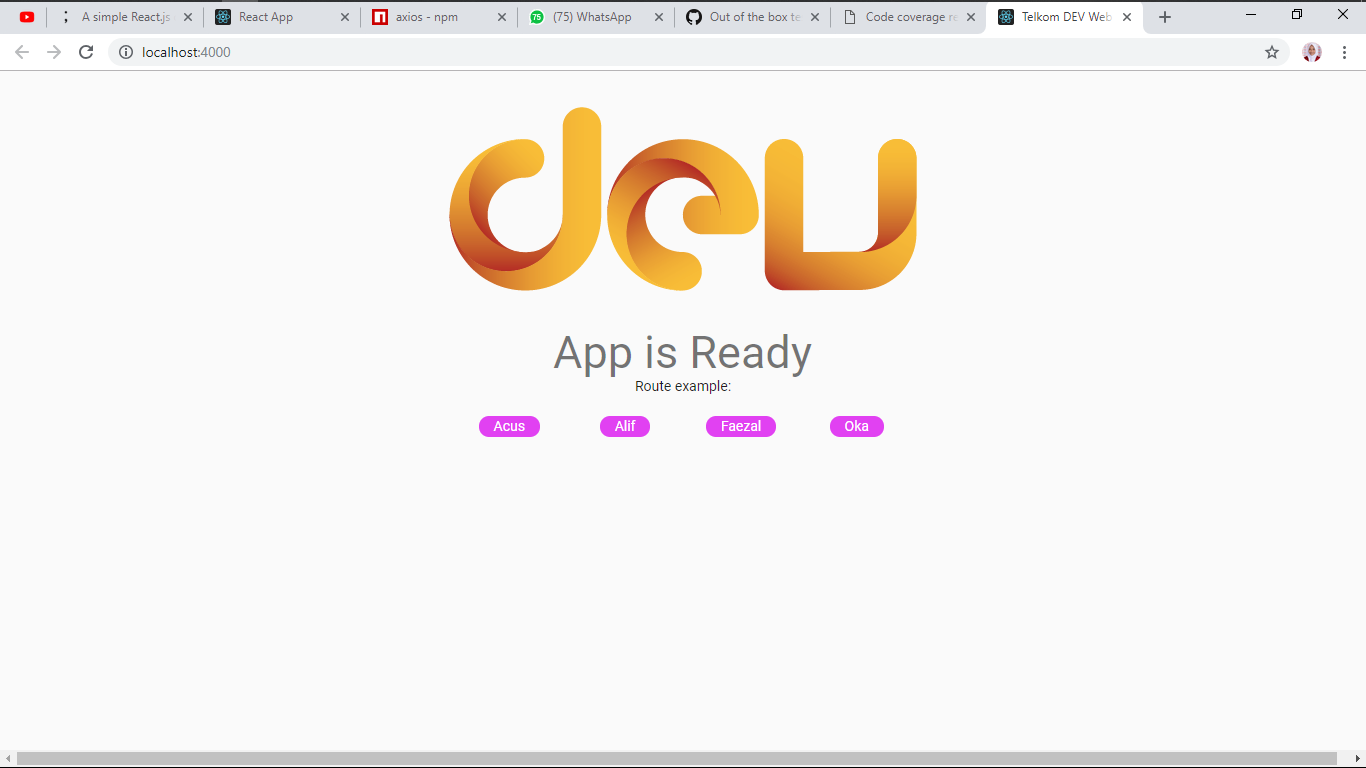
Perintah ini akan mengarahkan aplikasi untuk menuju folder pages/Home dan yang dijalankan pertama kali adalah file index.js. Lalu, akan memanggil component.js

import Home from './component';

dan akan memanggil component-component yang dibutuhkan melalui import.

import ListItem from '../../components/ListItem';

jika di run melalui terminal dengan mengetikkan npm run dev, maka secara otomatis, program akan dijalankan dibrowser.



Bagian yang ditandai adalah ListItem yang telah kita import tadi.

src/configs/images.js

Berfungsi untuk memanggil gambar

src/configs/index.js

Mengimport semua file yang ada pada folder configs

src/configs/routes.js

Mengatur route/alamat page/file

src/configs/services.js

File untuk menampung url API

src/constants/actions.js

Konstanta kondisi

src/configs/index.js

Mengimport semua file yang ada pada folder constants

src/reducers/index.js

Untuk menampung / memanggil semua reducer di setiap pages. Fungsi reducer adalah sebagai kondisi fetching data (mengurai data json).

src/styles/index.js

Untuk menyetting warna, font, dan lainnya agar tidak hardcode

src/utils/validateinput.js

Untuk memvalidasi form yang kita buat.

IX. Routes

Untuk membuat routes / untuk berpindah ke file lain, maka kita membutuhkan route. Buka file routes.js

const routes = {

HOME() { return `/`; },

LOGIN() { return `/login`; },

LIST\_ORDER\_MANAGEMENT() { return `/list-order-management` },

DEVELOPER(param) { return `/developer/${param}`; }

};

export default routes;

Berikut adalah path yang akan diakses untuk membuka page di browser. Selanjutnya, tambahkan nama pages di index.js pada folder pages

import Developer from './Developer';

import Error404 from './Error404';

import Home from './Home';

import ListOrderManagement from './ListOrderManagement';

import Login from './Login';

const pages = {

Developer,

Error404,

Home,

ListOrderManagement,

Login,

};

export default pages;

Setelah itu, panggil routes melalui App.js

<Route exact path={ROUTES.LIST\_ORDER\_MANAGEMENT()} component={pages.ListOrderManagement} />

**X. Async with Axios on CodeBase**

Pertama-tama, tambahkan link / url dari API pada configs/services.js, seperti berikut

const BASE\_URL = 'http://ec2-13-251-45-154.ap-southeast-1.compute.amazonaws.com:8080/api/v1';

const LOGIN\_URL = 'http://private-anon-c885589ea7-crmindigo.apiary-mock.com/login';

const services = {

GET\_ORDER\_MANAGEMENT: `${BASE\_URL}/insert-order-management`,

LOGIN: `${LOGIN\_URL}`,

};

export default services;

Untuk login, async ada di src/pages/login/actions.js

import { ACTIONS } from '../../constants';

import fetch from '../../utils/fetch';

import payload from './component';

import { SERVICES } from '../../configs';

export function fetchLogin() {

return dispatch => {

const options = {

method: 'POST',

url: SERVICES.LOGIN,

params: payload,

headers: {

'Content-Type': 'application/json'

}

};

dispatch(loadingAction());

fetch(options)

.then(res => {

dispatch(doneLoadingAction());

if (res.success) {

localStorage.setItem('data', JSON.stringify(res.data));

location.href = '/list-order-management';

} else {

dispatch(loginFailedAction('You are not allowed to access'));

}

})

.catch(err => {

dispatch(loginFailedAction(err.error\_description));

dispatch(doneLoadingAction());

});

};

}

function loadingAction() {

return { type: ACTIONS.LOADING };

}

function doneLoadingAction() {

return { type: ACTIONS.DONE\_LOADING };

}

function loginFailedAction(message) {

return {

type: ACTIONS.LOGIN\_FAILED,

message

};

}

Untuk ListOrderManagement, di src/pages/ListOrderManagement/actions.js

import { ACTIONS } from '../../constants';

import { AUTHORIZATION, SERVICES } from '../../configs';

import fetch from '../../utils/fetch';

export function fetchDataOrderManagement() {

return (dispatch) => {

const options = {

method: 'get',

url: `${SERVICES.GET\_ORDER\_MANAGEMENT}?page=2&size=5`,

headers: {

Authorization: AUTHORIZATION

}

};

dispatch(loadingAction());

fetch(options)

.then( response => {

dispatch(dataFetched(response));

})

.catch(() => {

dispatch(dataFetched({}));

});

};

}

function loadingAction() {

return { type: ACTIONS.LOADING };

}

function dataFetched(data) {

return {

type: ACTIONS.LIST\_OF\_ORDER\_MANAGEMENT\_FETCHED,

data: data

};

}