

TUGAS PERORANGAN

LAPORAN MINGGU 12: Firebase Login

Disusun sebagai

MATA KULIAH: PEMROGRAMAN BERBASIS FRAMEWORK

Oleh:

Rois Dwi Admaja/1741720193

TI – 3B / 23



PROGRAM STUDI: D-IV TEKNIK INFORMATIKA

JURUSAN: TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2020

1. Menambahkan konfigurasi firebase.js

```
src > firebase > JS firebase.js > ...
1  import firebase from "firebase/app";
2  import "firebase/auth";
3  import "firebase/firestore";
4
5  const firebaseConfig = {
6    apiKey: "AIzaSyCsFkorNs1_GJOIYRr0zzpAx8GZowxtYH8",
7    authDomain: "react-login-fab26.firebaseio.com",
8    databaseURL: "https://react-login-fab26.firebaseio.com",
9    projectId: "react-login-fab26",
10   storageBucket: "react-login-fab26.appspot.com",
11   messagingSenderId: "506362191682",
12   appId: "1:506362191682:web:1a4e23451f9ca5bf09f2db",
13   measurementId: "G-MVQPVS6K1"
14 }
15
16 export const myFirebase = firebase.initializeApp(firebaseConfig);
17 const baseDb = myFirebase.firestore();
18 export const db = baseDb;
```

2. Membuat file auth pada folder src/actions/

```
src > actions > JS auth.js > ...
1  import {myFirebase} from "../firebase/firebase";
2
3  export const LOGIN_REQUEST = "LOGIN_REQUEST";
4  export const LOGIN_SUCCESS = "LOGIN_SUCCESS";
5  export const LOGIN_FAILURE = "LOGIN_FAILURE";
6
7  export const LOGOUT_REQUEST = "LOGOUT_REQUEST";
8  export const LOGOUT_SUCCESS = "LOGOUT_SUCCESS";
9  export const LOGOUT_FAILURE = "LOGOUT_FAILURE";
10
11 export const VERIFY_REQUEST = "VERIFY_REQUEST";
12 export const VERIFY_SUCCESS = "VERIFY_SUCCESS";
13
```

```

15 const requestLogin = _=>{
16   return {
17     type: LOGIN_REQUEST
18   };
19 };
20
21 const receiveLogin = user =>{
22   return {
23     type: LOGIN_SUCCESS,
24     user
25   };
26 };
27
28 const loginError = _ =>{
29   return {
30     type: LOGIN_FAILURE
31   };
32 };
33
34 const requestLogout = _=>{
35   return {
36     type: LOGOUT_REQUEST
37   };
38 };
39
40 const receiveLogout = _=>{
41   return {
42     type: LOGOUT_SUCCESS
43   };

```

```

44   };
45
46 const logoutError = _=>{
47   return{
48     type: LOGOUT_FAILURE
49   };
50 };
51
52 const verifyRequest = _=>{
53   return {
54     type: VERIFY_REQUEST
55   };
56 };
57
58 const verifySuccess = _=>{
59   return {
60     type: VERIFY_SUCCESS
61   };
62 };
63
64 export const loginUser = (email, password) => dispatch =>{
65   dispatch(requestLogin());
66   myFirebase
67     .auth()
68     .signInWithEmailAndPassword(email, password)
69     .then(user=>{
70       dispatch(receiveLogin(user));
71     }).catch(error=>{
72       dispatch(loginError());

```

```

src > actions > auth.js > ...
73   });
74 };
75
76 export const logoutUser = _=>dispatch=>{
77   dispatch(requestLogout());
78   myFirebase
79     .auth()
80     .signOut()
81     .then(()=>{
82       dispatch(receiveLogout());
83     })
84     .catch(error=>{
85       dispatch(logoutError());
86     });
87 };
88
89
90 export const verifyAuth = ()=>dispatch=>{
91   dispatch(verifyRequest());
92   myFirebase.auth().onAuthStateChanged(user=>{
93     if (user !== null) {
94       dispatch(receiveLogin(user));
95     }
96     dispatch(verifySuccess());
97   })
98 }

```

Pada file auth melakukan import pada firebase , dan membuat variabel untuk autektikasi login menggunakan promise

3. Setelah itu membuat reducer untuk autektikasi login

```

src > reducers > auth.js > default
1  import { LOGIN_REQUEST,
2    LOGIN_SUCCESS,
3    LOGIN_FAILURE,
4    LOGOUT_REQUEST,
5    LOGOUT_SUCCESS,
6    LOGOUT_FAILURE,
7    VERIFY_REQUEST,
8    VERIFY_SUCCESS } from "../actions/auth";
9
10 export default {
11   state = {
12     isLoggingIn:false,
13     isLoggingOut:false,
14     isVerifying:false,
15     loginError:false,
16     logoutError:false,
17     isAuthenticated:false,
18     user:{}
19   },
20   action
21 }=>{
22   switch (action.type) {
23     case LOGIN_REQUEST:
24       return {
25         ...state,
26         isLoggingIn:true,
27         loginError:false
28       };
29     case LOGIN_SUCCESS:
30
31       isLoggingOut:false,
32       logoutError:true
33     };
34     case VERIFY_REQUEST:
35       return{
36         ...state,
37         isVerifying:true,
38         verifyingError:false
39       };
40     case VERIFY_SUCCESS:
41       return{
42         ...state,
43         isVerifying:false
44       }
45     default:
46       return state;
47   }
48 }

```

```

30
31   return{
32     ...state,
33     isLoggingIn:false,
34     isAuthenticated:true,
35     user:action.user
36   };
37   case LOGIN_FAILURE:
38     return{
39       ...state,
40       isLoggingIn:false,
41       isAuthenticated:false,
42       loginError:true
43     };
44   case LOGOUT_REQUEST:
45     return{
46       ...state,
47       isLoggingOut:true,
48       logoutError:false
49     };
50   case LOGOUT_SUCCESS:
51     return{
52       ...state,
53       isLoggingOut:false,
54       isAuthenticated:false,
55       user:{}
56     };
57   case LOGOUT_FAILURE:
58     return{
59       ...state,

```

4. Lalu membuat index.js pada folder reducer

```

src > reducers > index.js > default
1  import {combineReducers} from "redux";
2  import auth from "./auth";
3  export default combineReducers({auth});

```


5. Setelah membuat index.js pada reducer lalu membuat configureStore.js

```


src > configureStore.js > ...
1  import {applyMiddleware, createStore} from "redux";
2  import thunkMiddleware from "redux-thunk";
3
4  import {verifyAuth} from "../actions/auth";
5  import rootReducer from "../reducers";
6
7
8
9  export default function configureStore(persistedState){
10    const store = createStore(
11      rootReducer,
12      persistedState,
13      applyMiddleware(thunkMiddleware)
14    );
15    store.dispatch(verifyAuth());
16    return store;
17  }

```

6. Lalu membuat file Root.js yang berfungsi sebagai root pada web ini

```
src >  Root.js >  default
1  import React from "react";
2  import {Provider} from "react-redux";
3  import {BrowserRouter as Router} from "react-router-dom";
4  import App from "./App";
5  import configureStore from "./configureStore";
6
7  const store = configureStore();
8  function Root(){
9      return(
10         <Provider store={store}>
11             <Router>
12                 <App/>
13             </Router>
14         </Provider>
15     );
16 }
17
18 export default Root;
```

7. Lalu mengedit index.js pada folder src digunakan untuk menampung Root

```
src >  index.js
1  import React from 'react';
2  import ReactDOM from 'react-dom';
3  import './index.css';
4  import Root from './Root'
5  import * as serviceWorker from './serviceWorker';
6
7  ReactDOM.render(<Root />, document.getElementById('root'));
8
9  // If you want your app to work offline and load faster, you can change
10 // unregister() to register() below. Note this comes with some pitfalls.
11 // Learn more about service workers: https://bit.ly/CRA-PWA
12 serviceWorker.unregister();
13
```

8. Merubah App.js pada folder src

```
src >  App.js >  App
1  import React from 'react';
2  import {Route, Switch} from "react-router-dom";
3  import {connect} from "react-redux";
4  import ProtectedRoute from './components/ProtectedRoute'
5  import Home from './components/Home';
6  import Login from './components/Login';
7
8  function App(props) {
9      const {isAuthenticated, isVerifying} = props;
10     return (
11         <Switch>
12             <ProtectedRoute exact path="/" component={Home}
13                 isAuthenticated={isAuthenticated}
14                 isVerifying={isVerifying}/>
15             <Route path="/login" component={Login}/>
16         </Switch>
17     );
18 }
19
20 function mapState(state) {
21     return{
22         isAuthenticated:state.auth.isAuthenticated,
23         isVerifying:state.auth.isVerifying
24     };
25 }
26
27 export default connect(mapState) (App);
```

9. Membuat halaman login

```

41 class Login extends Component {
42   state = { email: "", password: "" };
43   handleEmailChange = ({ target }) => {
44     this.setState({ email: target.value });
45   };
46   handlePasswordChange = ({ target }) => {
47     this.setState({ password: target.value });
48   };
49   handleSubmit = () => {
50     const { dispatch } = this.props;
51     const { email, password } = this.state;
52     dispatch(loginUser(email, password));
53   };
54   render() {
55     const { classes, loginError, isAuthenticated } = this.props;
56     if (isAuthenticated) {
57       return <Redirect to="/" />;
58     } else {
59       return (
60         <Container component="main" maxWidth="xs">
61           <Paper className={classes.paper}>
62             <Avatar className={classes.avatar}>
63               <LockOutlinedIcon />
64             </Avatar>
65             <Typography component="h1" variant="h5">Sign in</Typography>
66             <TextField variant="outlined"
67               margin="normal"
68               fullWidth
69               id="email"
70               label="Email Address"
71               name="email"
72               onChange={this.handleEmailChange}/>
73             <TextField variant="outlined"
74               margin="normal"
75               fullWidth
76               name="password"
77               label="Password"
78               type="password"
79               id="password"
80               onChange={this.handlePasswordChange}/>
81             {loginError && (
82               <Typography component="p"
83                 className={classes.errorText}>
84                 Incorrect email or password.
85               </Typography>
86             )}
87             <Button type="button"
88               fullWidth
89               variant="contained"
90               color="primary"
91               className={classes.submit}
92               onClick={this.handleSubmit}>
93               Sign In
94             </Button>
95           </Paper>
96         </Container>
97       );
98     }
99   }
100 }
101
102 function mapStateToProps(state) {
103   return {
104     isLoggingIn: state.auth.isLoggingIn,
105     loginError: state (parameter) state: any
106     isAuthenticated: state.auth.isAuthenticated
107   };
108 }
109 export default withStyles(styles)(connect(mapStateToProps)(Login));

```

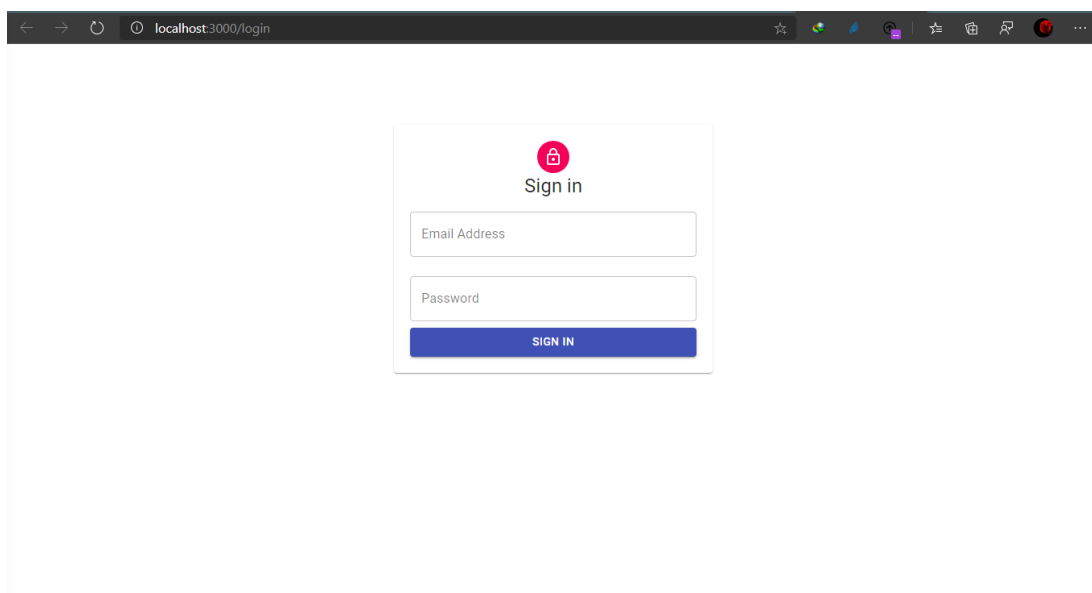
10. Lalu membuat halaman home , halaman home ini adalah protected koponen dimana untuk mengaksesnya harus login dulu

```

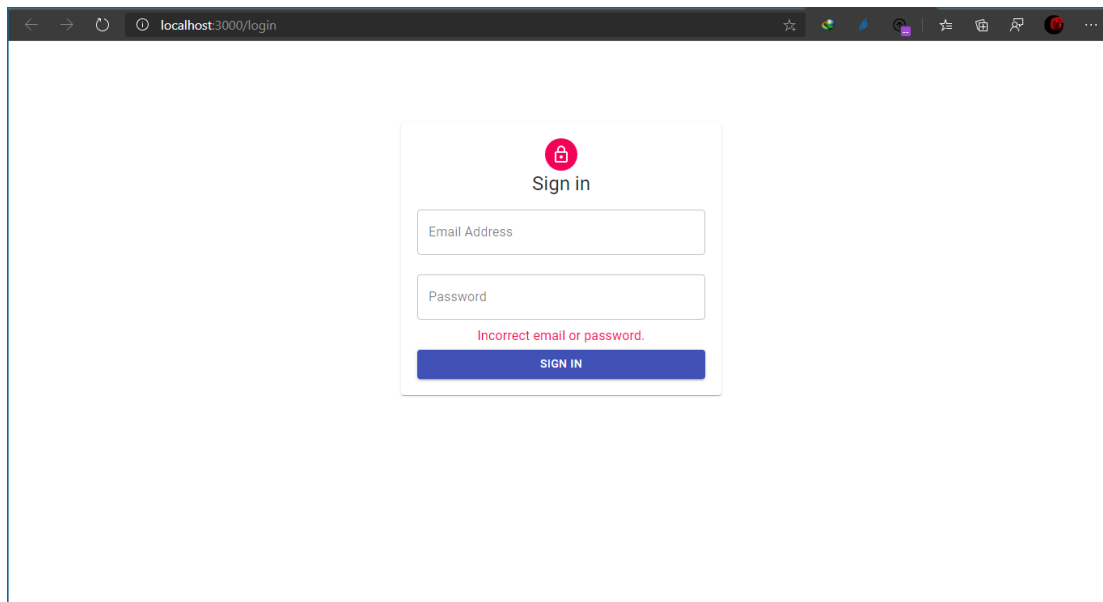
src > components > Home.js > ...
1  import React, { Component } from "react";
2  import { connect } from "react-redux";
3  import { logoutUser } from "../actions/auth";
4
5  class Home extends Component {
6    handleLogout = _ =>{
7      const {dispatch} = this.props;
8      dispatch(logoutUser());
9    };
10
11    render(){
12      const {isLoggedIn, logoutError} = this.props;
13      return (
14        <div>
15          <h1>This is your app's protected area.</h1>
16          <p>Any routes here will also be protected</p>
17          <button onClick={this.handleLogout}>Logout</button>
18          {isLoggedIn && <p>Logging Out...</p>}
19          {logoutError && <p>Error logging out</p>}
20        </div>
21      );
22    }
23  }
24
25  function mapStateToProps(state) {
26    return {
27      isLoggedIn: state.auth.isLoggedIn,
28      logoutError: state.auth.logoutError
29    };
30  }
31
32  export default connect(mapStateToProps)(Home);

```

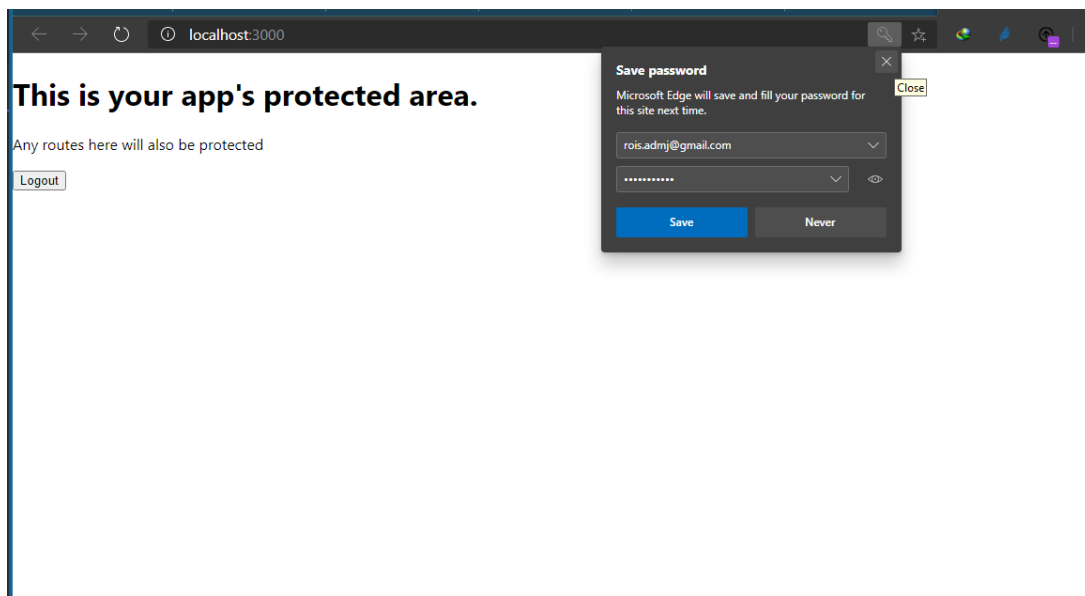
Berikut hasil dari program diatas



Jika belum diisikan email dan password maka akan terjadi incorrect password



Jika user dan password sudah terdaftar maka user dapat mengakses halaman home



Lalu jika logout diklik akan keluar ke halaman login lagi