**LAPORAN PEMROGRAMAN BERBASIS FRAMEWORK**

**JOBSHEET XI**



Dibuat oleh:

Nama : Lili Nur Indah Sari

NIM : 1841720037

**PROGRAM STUDI TEKNIK INFORMATIKA JURUSAN TEKNOLOGI INFORMASI POLITEKNIK NEGERI MALANG**

**APRIL 2021**

**FIREBASE LOGIN**

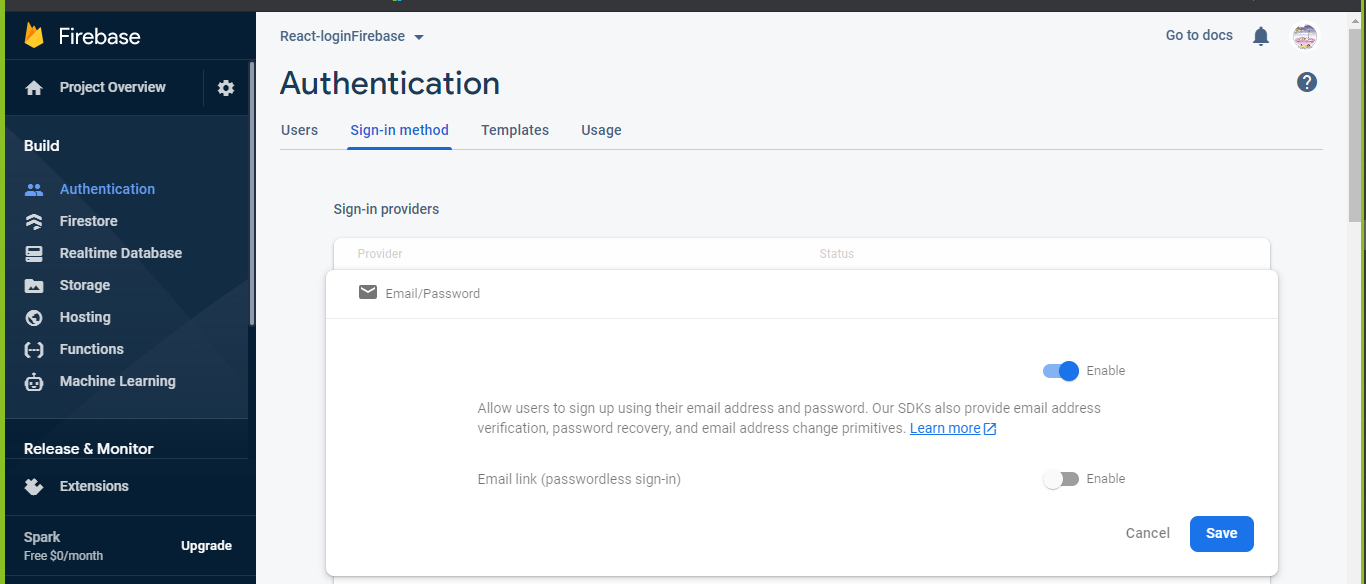
Firebase adalah serangkaian alat (bantu) untuk developer aplikasi, namun bukan hanya untuk developer aplikasi Android. Ini untuk developer aplikasi iOS dan juga developer aplikasi web. Akan tetapi, karena kursus ini adalah tentang development Android, pelajaran ini hanya membahas tentang cara menggunakan Firebase dengan aplikasi Android.

Sebagai developer Android, Anda menggunakan Android Studio untuk membangun aplikasi, namun Anda bisa menggunakan Firebase untuk menambahkan fitur ke aplikasi, mendapatkan pengguna aplikasi yang lebih luas, menguji aplikasi, menghasilkan pendapatan dari aplikasi, dan mendapatkan analisis tentang penggunaan aplikasi tersebut.

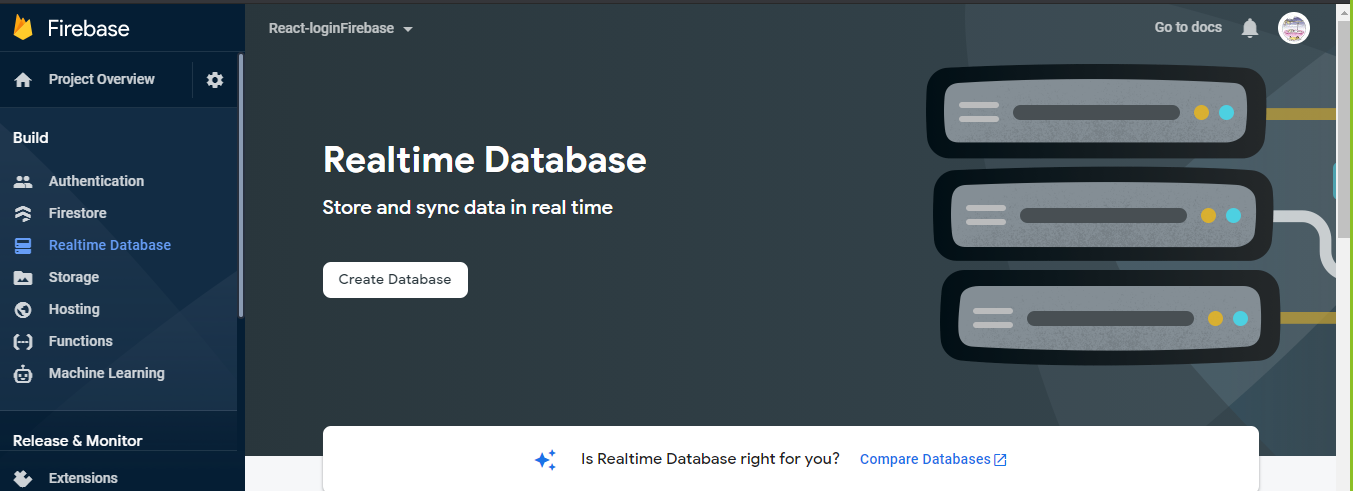
Langkah :

1. Setting Firebase Console

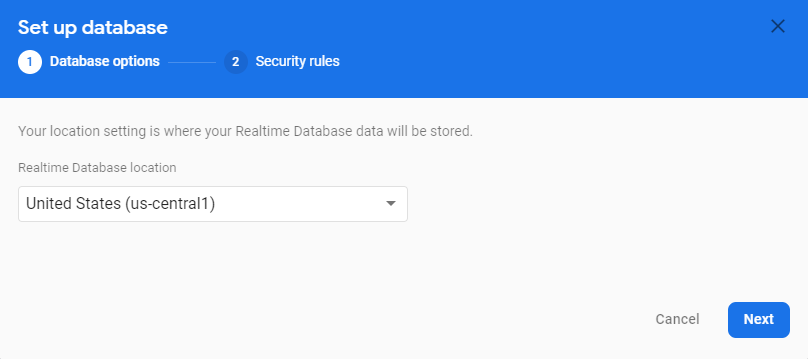
Login ke firebase console dan add a new project. Pilih authentification dan pilih tab menu Set up sign in method



# 2. Setelah enabling athentification pilih menu Database dan pilih create database



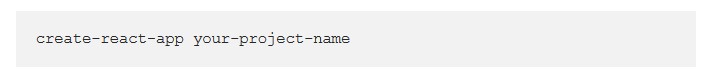
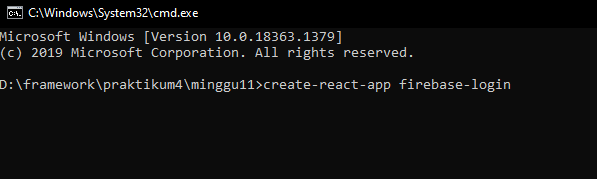
1. Select **Start in test mode**.



1. PIlih Done

# Setting react app dan firebase

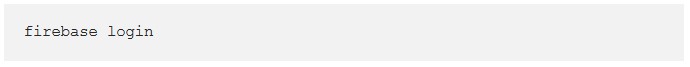
## 1. Buat aplikasi baru dengan perintah

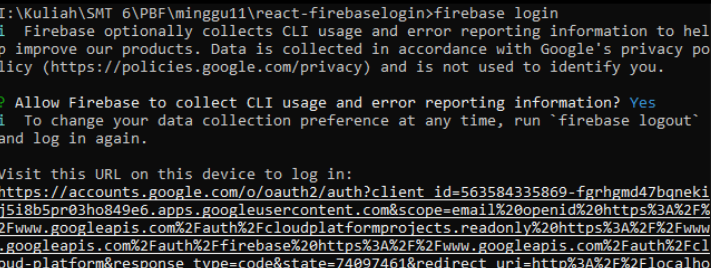


## 2. Install project baru dengan perintah

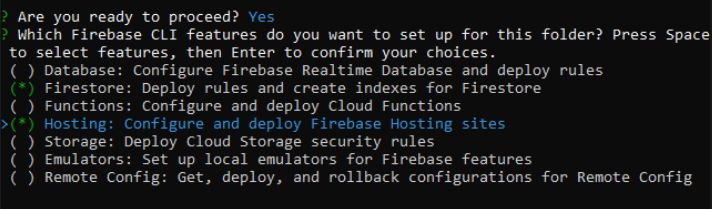


## 3. Install firebase



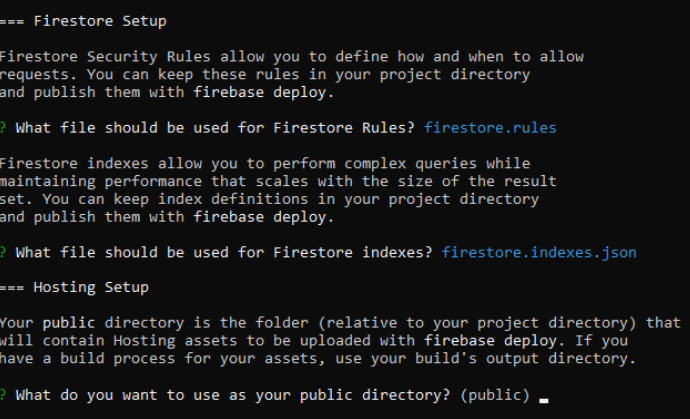


## 4. Pilih setting untuk firebase nya firebase dan Hosting



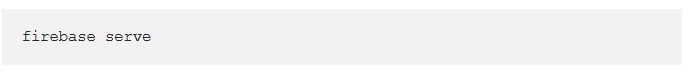
## 5. Pilih build jangan pilih Do not choose public

## 6. Pilih



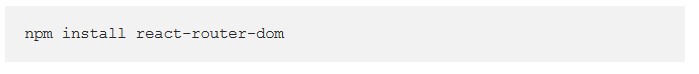
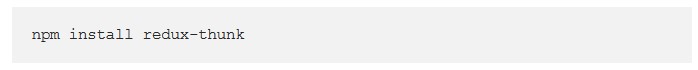
## Pilih Y lalu N untuk overwriting index.html

### 7. Untuk membangun firebase

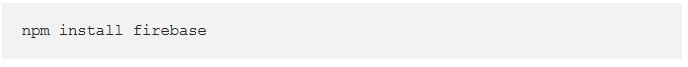


# Instalasi Redux

## 1. Install Redux dan react redux, redux thunk



## 2. Install firebase



## 3. Pilih UI library



## 4. Paste kan file public/index.html

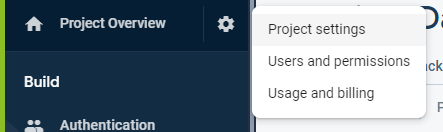


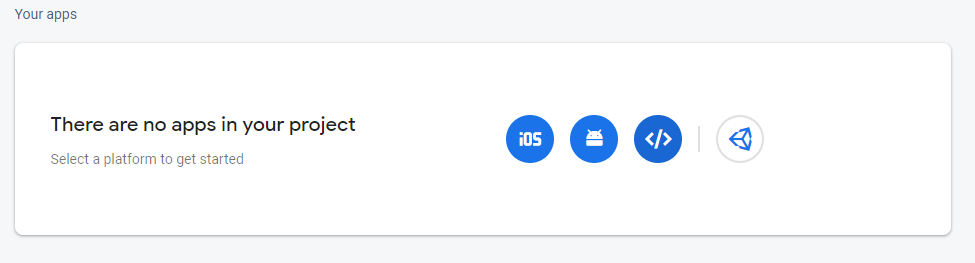
## 5. Install material icon

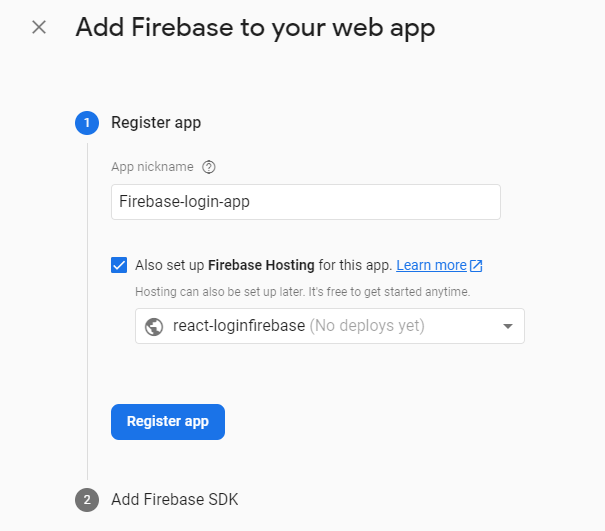


# Setting firebase config

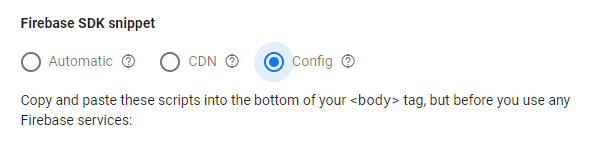
## 1. Pilih project setting







## 2. Pilih config



## 3. Edit **firebase/firebase.js**

import firebase from "firebase/app";

import "firebase/auth";

import "firebase/firestore";

const firebaseConfig = {

    // Your config values

    apiKey: "AIzaSyC\_VgdayZnH8MJEdkVaJZcDYecvNGmPICo",

    authDomain: "react-loginfirebase.firebaseapp.com",

    databaseURL: "https://react-loginfirebase-default-rtdb.firebaseio.com",

    projectId: "react-loginfirebase",

    storageBucket: "react-loginfirebase.appspot.com",

    messagingSenderId: "1033562874424",

    appId: "1:1033562874424:web:34bbecd46e9dda0354bccd",

    measurementId: "G-R6TEYEFHBL"

};

export const myFirebase = firebase.initializeApp(firebaseConfig);

const baseDb = myFirebase.firestore();

export const db = baseDb;

## 4. **Edit actions/auth.js**

import { myFirebase } from "../firebase/firebase";

export const LOGIN\_REQUEST = "LOGIN\_REQUEST";

export const LOGIN\_SUCCESS = "LOGIN\_SUCCESS";

export const LOGIN\_FAILURE = "LOGIN\_FAILURE";

export const LOGOUT\_REQUEST = "LOGOUT\_REQUEST";

export const LOGOUT\_SUCCESS = "LOGOUT\_SUCCESS";

export const LOGOUT\_FAILURE = "LOGOUT\_FAILURE";

export const VERIFY\_REQUEST = "VERIFY\_REQUEST";

export const VERIFY\_SUCCESS = "VERIFY\_SUCCESS";

## 5. user requesting to login masih dalam **actions/auth.js**

const requestLogin = () => {

    return {

        type: LOGIN\_REQUEST

    };

};

const receiveLogin = user => {

    return {

        type: LOGIN\_SUCCESS,

        user

    };

};

const loginError = () => {

    return {

        type: LOGIN\_FAILURE

    };

};

## 6. edit loginUser()

import { myFirebase } from "../firebase/firebase";

export const LOGIN\_REQUEST = "LOGIN\_REQUEST";

export const LOGIN\_SUCCESS = "LOGIN\_SUCCESS";

export const LOGIN\_FAILURE = "LOGIN\_FAILURE";

export const LOGOUT\_REQUEST = "LOGOUT\_REQUEST";

export const LOGOUT\_SUCCESS = "LOGOUT\_SUCCESS";

export const LOGOUT\_FAILURE = "LOGOUT\_FAILURE";

export const VERIFY\_REQUEST = "VERIFY\_REQUEST";

export const VERIFY\_SUCCESS = "VERIFY\_SUCCESS";

const requestLogin = () => {

    return {

        type: LOGIN\_REQUEST

    };

};

const receiveLogin = user => {

    return {

        type: LOGIN\_SUCCESS,

        user

    };

};

const loginError = () => {

    return {

        type: LOGIN\_FAILURE

    };

};

const requestLogout = () => {

    return {

        type: LOGOUT\_REQUEST

    };

};

const receiveLogout = () => {

    return {

        type: LOGOUT\_SUCCESS

    };

};

const logoutError = () => {

    return {

        type: LOGOUT\_FAILURE

    };

};

const verifyRequest = () => {

    return {

        type: VERIFY\_REQUEST

    };

};

const verifySuccess = () => {

    return {

        type: VERIFY\_SUCCESS

    };

};

export const loginUser = (email, password) => dispatch => {

    dispatch(requestLogin());

    myFirebase

        .auth()

        .signInWithEmailAndPassword(email, password)

        .then(user => {

            dispatch(receiveLogin(user));

        })

        .catch(error => {

            //Do something with the error if you want!

            dispatch(loginError());

        });

};

export const logoutUser = () => dispatch => {

    dispatch(requestLogout());

    myFirebase

        .auth()

        .signOut()

        .then(() => {

            dispatch(receiveLogout());

        })

        .catch(error => {

            //Do something with the error if you want!

            dispatch(logoutError());

        });

};

export const verifyAuth = () => dispatch => {

    dispatch(verifyRequest());

    myFirebase.auth().onAuthStateChanged(user => {

        if (user !== null) {

            dispatch(receiveLogin(user));

        }

        dispatch(verifySuccess());

    });

};

# Setting reducer

## Buka reducers/auth.js

1. import {
2. LOGIN\_REQUEST,
3. LOGIN\_SUCCESS,
4. LOGIN\_FAILURE,
5. LOGOUT\_REQUEST,
6. LOGOUT\_SUCCESS,
7. LOGOUT\_FAILURE,
8. VERIFY\_REQUEST,
9. VERIFY\_SUCCESS
10. } from "../actions/auth";
11. export default (
12. state = {
13. isLoggingIn: false,
14. isLoggingOut: false,
15. isVerifying: false,
16. loginError: false,
17. logoutError: false,
18. isAuthenticated: false,
19. user: {}
20. },
21. action
22. ) => {
23. switch (action.type) {
24. case LOGIN\_REQUEST:
25. return {
26. ...state,
27. isLoggingIn: true,
28. loginError: false
29. };
30. case LOGIN\_SUCCESS:
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,
60. isLoggingOut: false,
61. logoutError: true
62. };
63. case VERIFY\_REQUEST:
64. return {
65. ...state,
66. isVerifying: true,
67. verifyingError: false
68. };
69. case VERIFY\_SUCCESS:
70. return {
71. ...state,
72. isVerifying: false
73. };
74. default:
75. return state;
76. }
77. };

## 2. Edit reducers/index.js

import { combineReducers } from "redux";

import auth from "./auth";

export default combineReducers({ auth });

## 3. Setting Redux plumbing

## 4. **Pilih src** folder called **configureStore.js dan edit**

import { applyMiddleware, createStore } from "redux";

import thunkMiddleware from "redux-thunk";

import { verifyAuth } from "./actions/auth";

import rootReducer from "./reducers";

export default function configureStore(persistedState) {

    const store = createStore(

        rootReducer,

        persistedState,

        applyMiddleware(thunkMiddleware)

    );

    store.dispatch(verifyAuth());

    return store;

}

## 5. Pilih dan edit folder src dan root.js

import React from "react";

import { Provider } from "react-redux";

import { BrowserRouter as Router } from "react-router-dom";

import App from "./App";

import configureStore from "./configureStore";

const store = configureStore();

function Root() {

    return (

        <Provider store={store}>

            <Router>

                <App />

            </Router>

        </Provider>

    );

}

export default Root;

## 

6. Pilih dan edit folder src/index.js

import React from 'react';

import ReactDOM from 'react-dom';

import Root from "./Root";

import \* as serviceWorker from './serviceWorker';

ReactDOM.render(<Root />, document.getElementById("root"));

serviceWorker.unregister();

## 7. Buat lah coding App.js

import React from "react";

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

import { connect } from "react-redux";

import ProtectedRoute from "./components/protectedRoute";

import Home from "./components/Home";

import Login from "./components/Login";

function App(props) {

  const { isAuthenticated, isVerifying } = props;

  return (

    <Switch>

      <ProtectedRoute

        exact

        path="/"

        component={Home}

        isAuthenticated={isAuthenticated}

        isVerifying={isVerifying}

      />

      <Route path="/login" component={Login} />

    </Switch>

  );

}

function mapStateToProps(state) {

  return {

    isAuthenticated: state.auth.isAuthenticated,

    isVerifying: state.auth.isVerifying

  };

}

export default connect(mapStateToProps)(App);

## 8. Buatlah file dan simpan **components/Login.js**

import React, { Component } from "react";

import { connect } from "react-redux";

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

import { loginUser } from "../actions/auth";

import { withStyles } from "@material-ui/styles";

import Avatar from "@material-ui/core/Avatar";

import Button from "@material-ui/core/Button";

import TextField from "@material-ui/core/TextField";

import LockOutlinedIcon from "@material-ui/icons/LockOutlined";

import Typography from "@material-ui/core/Typography";

import Paper from "@material-ui/core/Paper";

import Container from "@material-ui/core/Container";

const styles = () => ({

    "@global": {

        body: {

            backgroundColor: "#fff"

        }

    },

    paper: {

        marginTop: 100,

        display: "flex",

        padding: 20,

        flexDirection: "column",

        alignItems: "center"

    },

    avatar: {

        marginLeft: "auto",

        marginRight: "auto",

        backgroundColor: "#f50057"

    },

    form: {

        marginTop: 1

    },

    errorText: {

        color: "#f50057",

        marginBottom: 5,

        textAlign: "center"

    }

});

class Login extends Component {

    state = { email: "", password: "" };

    handleEmailChange = ({ target }) => {

        this.setState({ email: target.value });

    };

    handlePasswordChange = ({ target }) => {

        this.setState({ password: target.value });

    };

    handleSubmit = () => {

        const { dispatch } = this.props;

        const { email, password } = this.state;

        dispatch(loginUser(email, password));

    };

    render() {

        const { classes, loginError, isAuthenticated } = this.props;

        if (isAuthenticated) {

            return <Redirect to="/" />;

        } else {

            return (

                <Container component="main" maxWidth="xs">

                    <Paper className={classes.paper}>

                        <Avatar className={classes.avatar}>

                            <LockOutlinedIcon />

                        </Avatar>

                        <Typography component="h1" variant="h5">

                            Sign in

                        </Typography>

                        <TextField

                            variant="outlined"

                            margin="normal"

                            fullWidth

                            id="email"

                            label="Email Address"

                            name="email"

                            onChange={this.handleEmailChange}/>

                        <TextField

                            variant="outlined"

                            margin="normal"

                            fullWidth

                            name="password"

                            label="Password"

                            type="password"

                            id="password"

                            onChange={this.handlePasswordChange}/>

                        {loginError && (

                            <Typography component="p" className={classes.errorText}>

                                Incorrect email or password.

                            </Typography>

                        )}

                        <Button

                            type="button"

                            fullWidth

                            variant="contained"

                            color="primary"

                            className={classes.submit}

                            onClick={this.handleSubmit}>

                            Sign In

                        </Button>

                    </Paper>

                </Container>

            );

        }

    }

}

function mapStateToProps(state) {

    return {

        isLoggingIn: state.auth.isLoggingIn,

        loginError: state.auth.loginError,

        isAuthenticated: state.auth.isAuthenticated

    };

}

export default withStyles(styles)(connect(mapStateToProps)(Login));

## 9. Buka **component/Home.js**

import React, { Component } from "react";

import { connect } from "react-redux";

import { logoutUser } from "../actions/auth";

class Home extends Component {

    handleLogout = () => {

        const { dispatch } = this.props;

        dispatch(logoutUser());

    };

    render() {

        const { isLoggingOut, logoutError } = this.props; return (

            <div>

                <h1>This is your app's protected area.</h1>

                <p>Any routes here will also be protected</p>

                <button onClick={this.handleLogout}>Logout</button>

                {isLoggingOut && <p>Logging Out....</p>}

                {logoutError && <p>Error logging out</p>}

            </div>

        );

    }

}

function mapStateToProps(state) {

    return {

        isLoggingOut: state.auth.isLoggingOut,

        logoutError: state.auth.logoutError

    };

}

export default connect(mapStateToProps)(Home);

## 10. Buatfile dalam folder /src/component/ dengan nama protectedRoute.js

import React from "react";

import { Route, Redirect } from "react-router-dom";

const ProtectedRoute = ({

    component: Component,

    isAuthenticated,

    isVerifying,

    ...rest

}) => (

    <Route

        {...rest}

        render={props =>

            isVerifying ? (

                <div />

            ) : isAuthenticated ? (

                <Component {...props} />

            ) : (

                <Redirect

                    to={{

                        pathname: "/login",

                        state: { from: props.location }

                    }}

                />

            )

        }

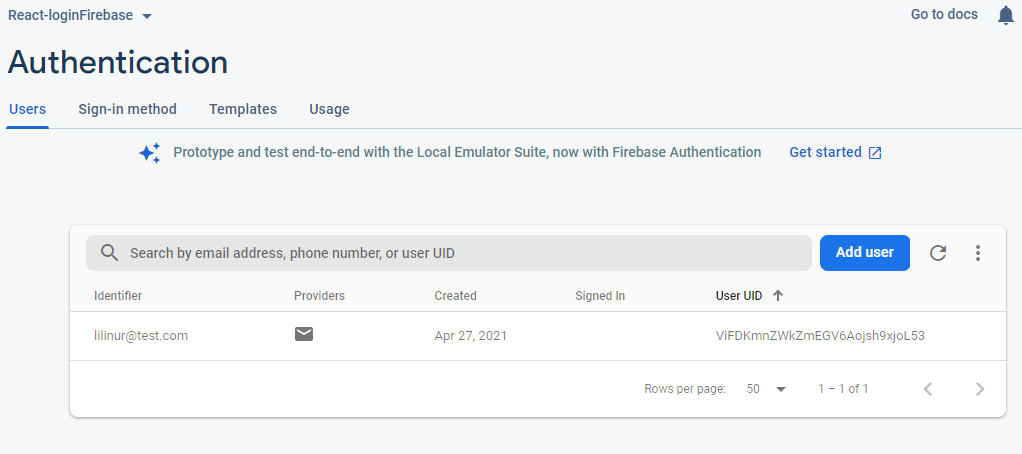
    />

);

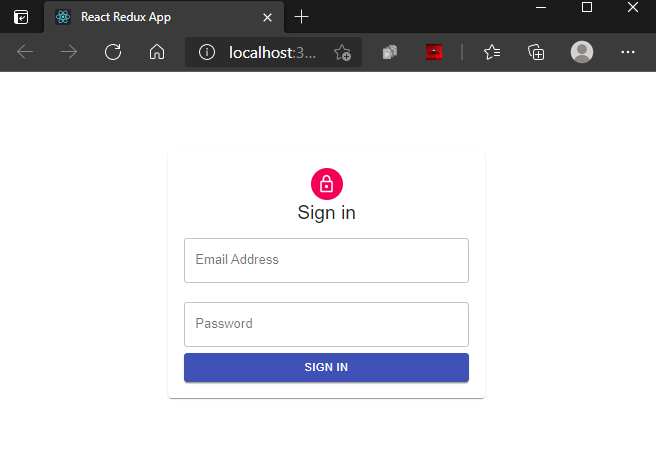
export default ProtectedRoute;

## 11. Running start

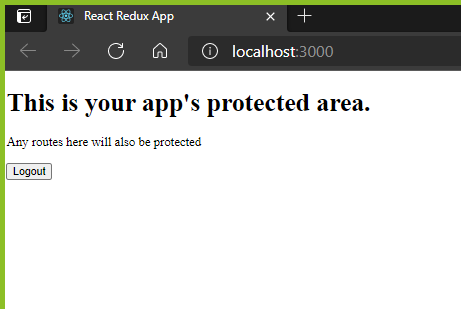
## 12. MAsuk ke firebase console kemudian add user



## 13. Pilih **npm start** kemudian masukkan username dan password



## 14. Kemudian masuk ke dalam halaman beranda



Link Youtube: <https://www.youtube.com/playlist?list=PL51COvB1Q4df_GcG9Yhs7pI_J847EsjIa>

Link Github: <https://github.com/lilipolinema/PBFReact.git>