**TUGAS INDIVIDU**

**FIREBASE LOGIN**

**Disusun sebagai**

**Pemerograman Berbasis Framework**

**Oleh :**

**Ahmaid Syifaur Romli**

**TI 3A**

**1741720156/01**



**PROGRAM STUDI D-IV T.INFORMATIKA**

**JURUSAN TEKNOLOGI INFORMASI POLITEKNIK NEGERI MALANG**

**2020**

**PRAKTIKUM**

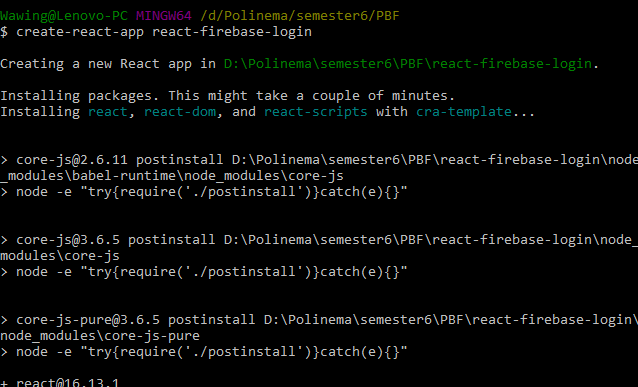
Langkah-langkah praktikum :

Setting firebase console.

1. Setting database pada project firebase **Database** dan pilih create database.
2. 
3. Select **Start in test mode**
4. Pilih done

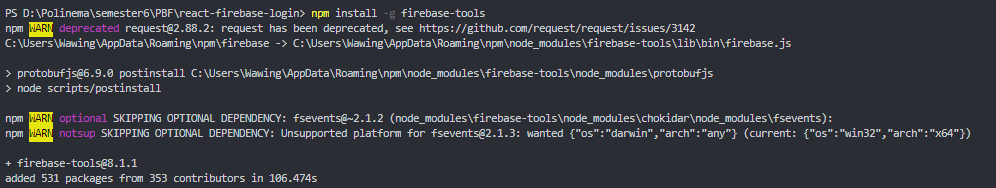
Setting react app dan firebase

1. Buat aplikasi baru dengan perintah create-react-app your-project-name

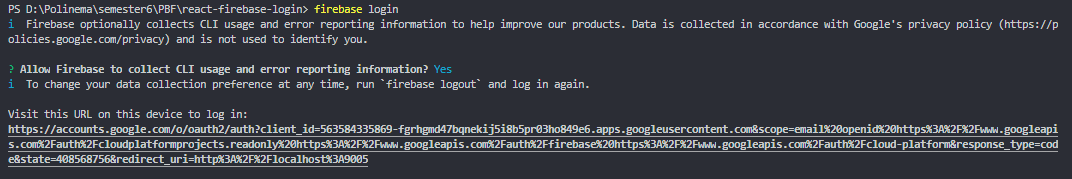


1. Install project baru dengan perintah npm run start
2. Install firebase npm install -g firebase-tools firebase login | firebase init

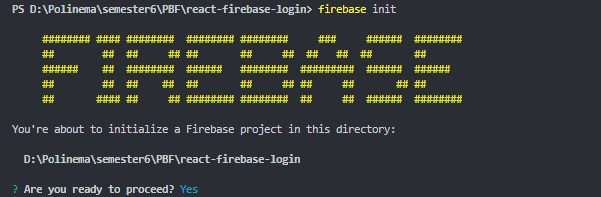
* Firebase install



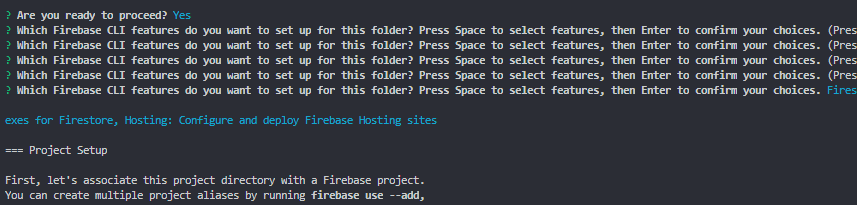
* Firebase login



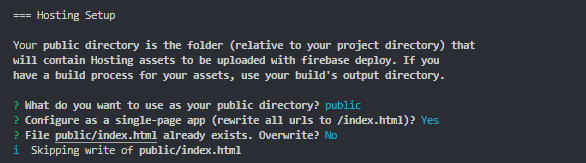
* Firebase init



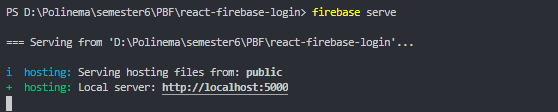
1. Pilih setting untuk firebase nya firebase dan Hosting



1. Pilih build (jangan pilih do not choose public)



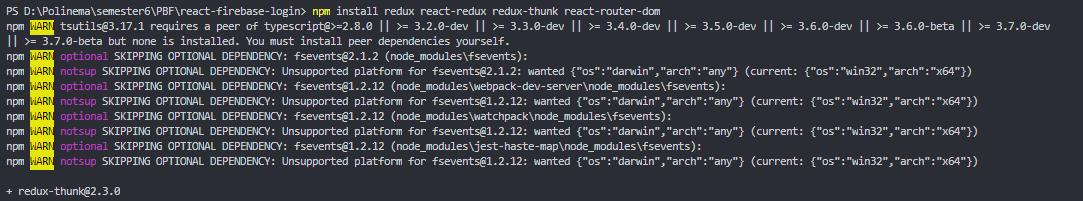
1. Pilih Y lalu N untuk overwriting index.html
2. Untuk membangun firebase npm run build | firebase serve



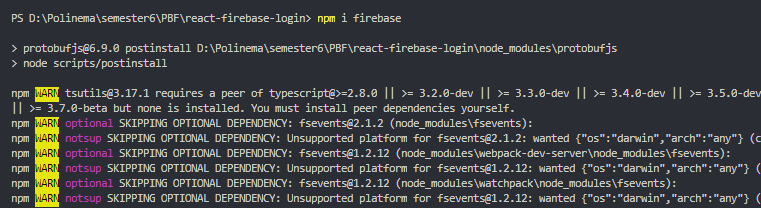
Instalasi Redux

1. Install Redux dan react redux, redux thunk.

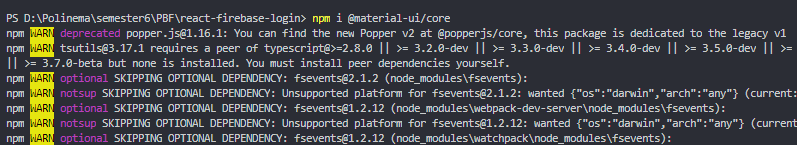
* Npm install redux
* Npm install react-redux
* Npm install redux-thunk
* Npm install react-router-dom



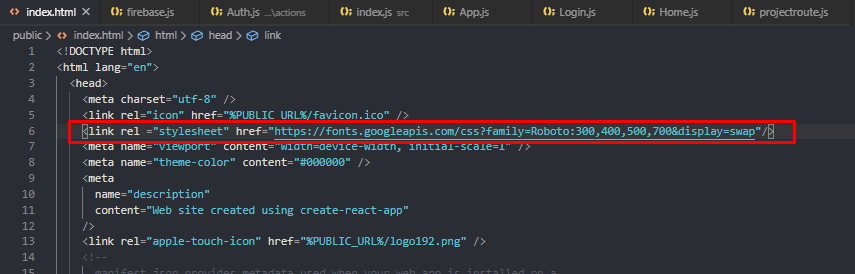
1. Install firebase npm install firebase



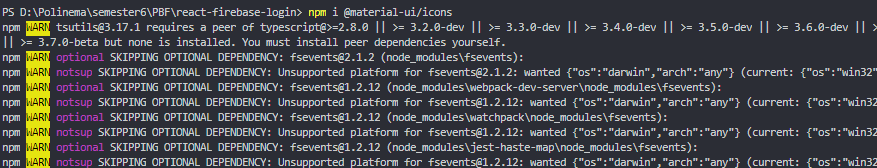
1. Pilih ui library npm install @material-ui/core



1. Paste kan file public/index.html

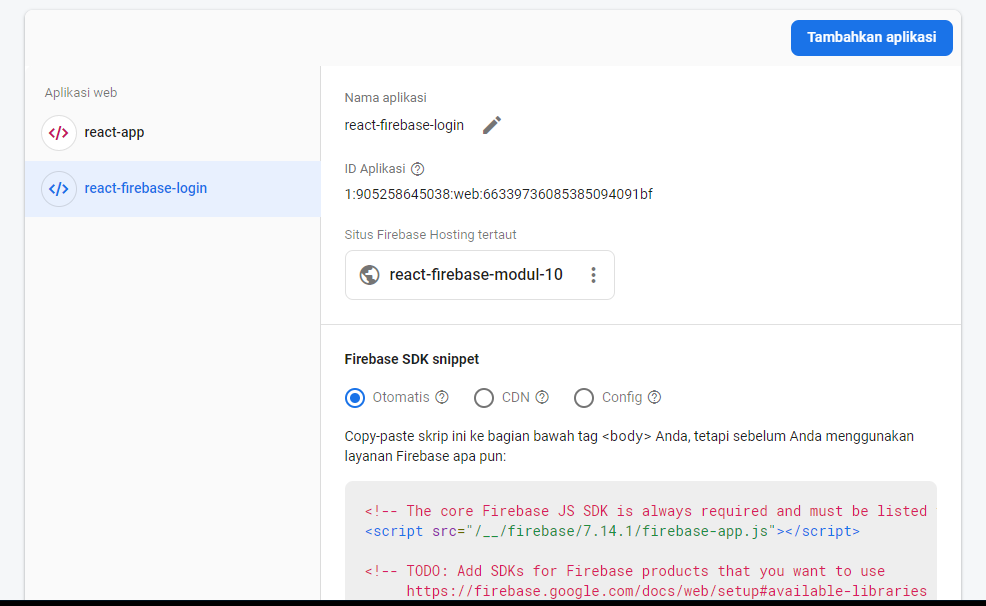


1. Install material icon npm install @material-ui/icons



Setting firebase config

1. Pilih project setting
2. Pilih Add new Project



1. Edit firebase/firebase.js\



1. 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";

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());

    });

};

1. 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

    };

};

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

    };

};

1. Edit loginUser()

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

1. Buka reducers/auth.js

import {

    LOGIN\_REQUEST,

    LOGIN\_SUCCESS,

    LOGIN\_FAILURE,

    LOGOUT\_REQUEST,

    LOGOUT\_SUCCESS,

    LOGOUT\_FAILURE,

    VERIFY\_REQUEST,

    VERIFY\_SUCCESS

} from "../actions/Auth";

export default (

    state = {

        isLoggingIn: false,

        isLoggingOut: false,

        isVerifying: false,

        loginError: false,

        logoutError: false,

        isAuthenticated: false,

        user: {}

    },

    action

) => {

    switch (action.type) {

        case LOGIN\_REQUEST:

            return {

                ...state,

                isLoggingIn: true,

                loginError: false

            };

        case LOGIN\_SUCCESS:

            return {

                ...state,

                isLoggingIn: false,

                isAuthenticated: true,

                user: action.user

            };

        case LOGIN\_FAILURE:

            return {

                ...state,

                isLoggingIn: false,

                isAuthenticated: false,

                loginError: true

            };

        case LOGOUT\_REQUEST:

            return {

                ...state,

                isLoggingOut: true,

                logoutError: false

            };

        case LOGOUT\_SUCCESS:

            return {

                ...state,

                isLoggingOut: false,

                isAuthenticated: false,

                user: {}

            };

        case LOGOUT\_FAILURE:

            return {

                ...state,

                isLoggingOut: false,

                logoutError: true

            };

        case VERIFY\_REQUEST:

            return {

                ...state,

                isVerifying: true,

                verifyingError: false

            };

        case VERIFY\_SUCCESS:

            return {

                ...state,

                isVerifying: false

            };

        default:

            return state;

    }

};

1. Edit reducers/index.js

import { combineReducers } from "redux";

import auth from "./Auth";

export default combineReducers({ auth });

1. Setting redux plumbing
2. 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 "./reducer";

export default function configureStore(persistedState) {

  const store = createStore(

    rootReducer,

    persistedState,

    applyMiddleware(thunkMiddleware)

  );

  store.dispatch(verifyAuth());

  return store;

}

1. 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;

1. Pilih 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"));

// If you want your app to work offline and load faster, you can change

// unregister() to register() below. Note this comes with some pitfalls.

// Learn more about service workers: https://bit.ly/CRA-PWA

serviceWorker.unregister();

1. Buatlah coding App.js

import React from "react";

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

import { connect } from "react-redux";

import ProtectedRoute from "./components/projectroute";

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);

1. 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));

1. 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 {

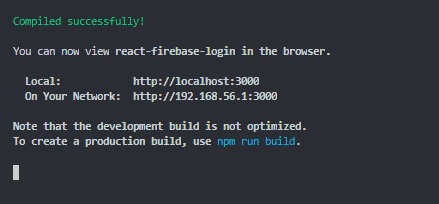
        isLoggingOut: state.auth.isLoggingOut,

        logoutError: state.auth.logoutError

    };

} export default connect(mapStateToProps)(Home);

1. Running start



1. Membuat protectedRouter.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 }

                                }}

                            />

                        )

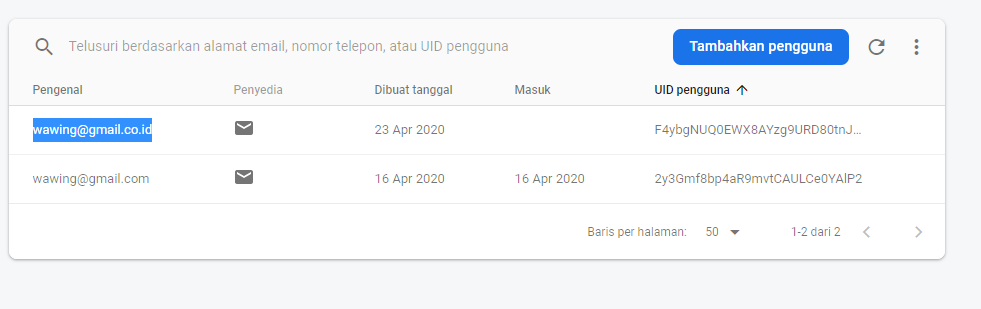
            }

        />

    );

export default ProtectedRoute;

1. Masuk ke firebase console kemudian add user



1. Pilih npm start kemudian masukkan username dan password
2. Kemudian masuk ke dalam halaman beranda

