

TUGAS INDIVIDU

Jobsheet 11

Firestore Login

Pemrograman Berbasis Framework



Oleh :

Dzikri Alif Abdillah (1741720052)

TI – 3A

PROGRAM STUDI – D4 TEKNIK INFORMATIKA

JURUSAN – TEKNOLOGI INFORMASI

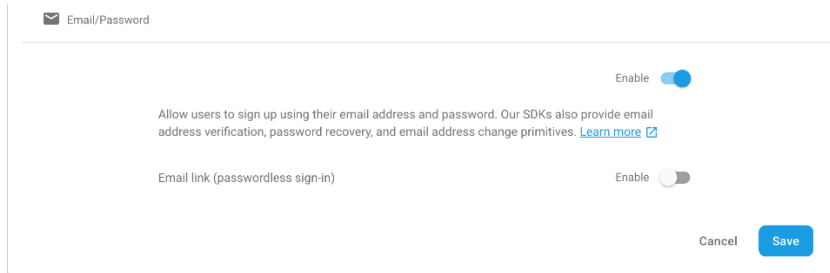
POLITEKNIK NEGERI MALANG

2020/2021

Langkah :

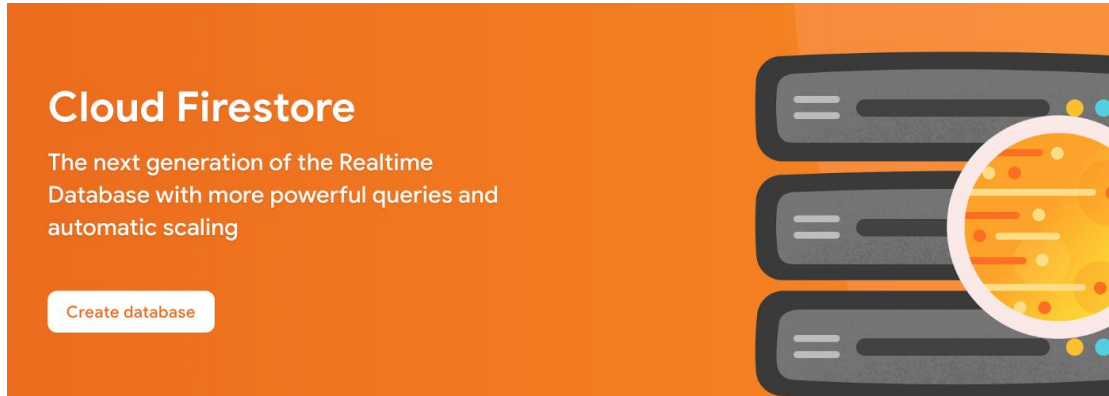
1. Setting Firebase Console

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



The screenshot shows the 'Email/Password' tab in the Firebase Authentication console. It features two toggle switches: 'Email/Password' (which is turned on) and 'Email link (passwordless sign-in)' (which is turned off). Below the toggles, there is a 'Cancel' button and a blue 'Save' button.

2. Setelah enabling authentication pilih menu Database dan pilih create database



3. Select Start in test mode.

Create database ×

✓ Secure rules for Cloud Firestore — 2 Set Cloud Firestore location

Your location setting is where your Cloud Firestore data will be stored.

⚠ After you set this location, you cannot change it later. Also, this location setting will be the location for your default Cloud Storage bucket. [Learn more](#)

Cloud Firestore location

nam5 (us-central) ▼

Enabling Cloud Firestore will prevent you from using Cloud Datastore with this project, notably from the associated App Engine app

Cancel **Done**

4. Pilih Done

Setting React App dan Firebase :

1. Buat Aplikasi baru dengan perintah **npx create-react-app namaApp** jika sudah langsung jalankan dengan perintah **npm start**
2. Install firebase dengan CLI **npm install -g firebase-tools** dan setelah selesai ketik **firebase login** , setelah selesai firebase login maka ketik **firebase init**
3. Pilih setting untuk firebasenya firebase dan hosting

```
( ) Database: Deploy Firebase Realtime Database rules
(*) Firestore: Deploy rules and create indexes for Firestore
( ) Functions: Configure and deploy Cloud Functions
>(*) Hosting: Configure and deploy Firebase Hosting sites
( ) Storage: Deploy Cloud Storage security rules
```

- Pilih build jangan pilih Do not choose public

```
Your public directory is the folder (relative to your project directory) that will contain Hosting assets to be uploaded with firebase deploy. If you have a build process for your assets, use your build's output directory.
```

```
? What do you want to use as your public directory? no
```

- Pilih

```
? Configure as a single-page app (rewrite all urls to /index.html)? No
+ Wrote no/404.html
+ Wrote no/index.html

i Writing configuration info to firebase.json...
i Writing project information to .firebaserc...
```

- Untuk membangun firebase

```
C:\MINGW64\d\DZ\KULIAH\2019-2020\Semester 6\Pemrograman Framework\loginfirebase
HP@DESKTOP-DZ MINGW64 /d/DZ/KULIAH/2019-2020/Semester 6/Pemrograman Framework/loginfirebase
$ firebase serve
```

Instalasi Redux :

- Install Redux dan react redux, redux thunk dengan command
npm install redux ,react-redux, redux-thunk, react-router-dom

- Install firebase

```
C:\MINGW64\d\DZ\KULIAH\2019-2020\Semester 6\Pemrograman Framework\loginfirebase
HP@DESKTOP-DZ MINGW64 /d/DZ/KULIAH/2019-2020/Semester 6/Pemrograman Framework/loginfirebase
$ npm install firebase
```

- Pilih UI library

```
C:\MINGW64\d\DZ\KULIAH\2019-2020\Semester 6\Pemrograman Framework\loginfirebase
HP@DESKTOP-DZ MINGW64 /d/DZ/KULIAH/2019-2020/Semester 6/Pemrograman Framework/loginfirebase
$ npm install @material-ui/core
```

4. Paste kan file public/index.html

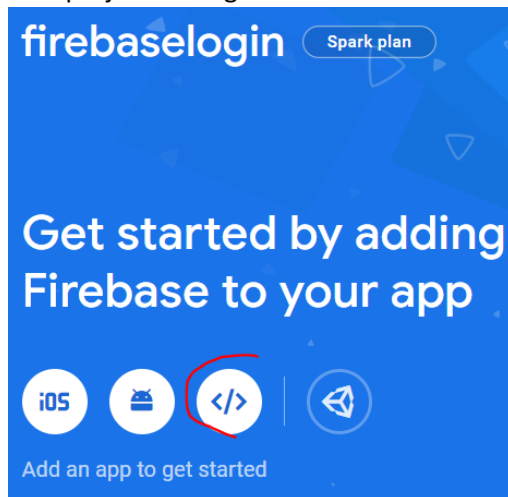
```
<link rel="manifest" href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700&display=swap"/>
```

5. Install material icon

```
MINGW64:/d/DZ/KULIAH/2019-2020/Semester 6/Pemrograman Framework/loginfirebase
HP@DESKTOP-DZ MINGW64 /d/DZ/KULIAH/2019-2020/Semester 6/Pemrograman Framework/loginfirebase
$ npm install @material-ui/icons
```

Setting firebase config :

1. Pilih project setting



1 Register app

App nickname ⓘ

firebaselogin

☒ Also set up **Firebase Hosting** for this app. [Learn more](#) ⓘ

Hosting can also be set up later. It's free to get started anytime.

fir-loginapp-532a4 .web.app

Register app

2 Add Firebase SDK

2. Pilih config

Firebase SDK snippet

☐ Automatic [?](#) ☐ CDN [?](#) ☒ Config [?](#)

3. Edit firebase/firebase.js

```
firebase > JS firebase.js > [?] firebaseConfig
import "firebase/auth";
import "firebase/firestore";

const firebaseConfig = {
};

export const myFirebase = firebase.initializeApp(firebaseConfig);
const baseDb = myFirebase.firestore();
export const db = baseDb;
```

4. Edit actions/auth.js

```
actions > JS 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()

```
actions > JS auth.js > ...
import {myFirebase} from "../firebase/firebase"

export const LOGIN_REQUEST = "LOGIN_REQUEST";
export const LOGIN_SUCCEST = "LOGIN_SUCCESS";
export const LOGIN_FAILURE = "LOGIN_FAILURE";

export const LOGOUT_REQUEST = "LOGOUT_REQUEST";
export const LOGOUT_SUCCEST = "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_SUCCEST,
    user
  };
};

const loginError = () => {
  return {
    type: LOGIN_FAILURE
  };
};

const requestLogout = () => {
  return {
    type: LOGOUT_REQUEST
  };
};

const receiveLogout = () => {
  return {
    type: LOGOUT_SUCCEST
  };
};
```

```

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 => {
      dispatch(loginError());
    });
};

export const logoutUser = () => dispatch => {
  dispatch(requestLogout());
  myFirebase.auth().signOut()
    .then(() => {
      dispatch(receiveLogout());
    })
    .catch(error => {
      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

```
reducers > JS auth.js > default
import {
  LOGIN_REQUEST,
  LOGIN_SUCCESS,
  LOGIN_FAILURE,
  LOGOUT_REQUEST,
  LOGOUT_SUCCESS,
  LOGOUT_FAILURE,
  VERIFY_REQUEST,
  VERIFY_SUCCESS
} from "../actions/";

export default (
  state = {},
  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;
  }
};

```

2. Edit reducers/index.js

```

reducers > JS 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

```
JS configureStore.js > ...
import { applyMiddleware, createStore } from "redux";
import thunkMiddleware from "redux-thunk";

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

```
JS 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

```
JS 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
// unregister() to register() below. Note this comes with so
// Learn more about service workers: https://bit.ly/CRA-PWA
serviceWorker.unregister();
```

7. Buat lah coding App.js

```
JS 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

```
components > JS Login.js > styles > form
import React, { Component } from "react";
import { connect } from "react-redux";
import { Redirect } from "react-router-dom";
import { loginUser } from "../actions";
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}>

```

```

    {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

```

components > JS Home.js > default
import React, { Component } from "react";
import { connect } from "react-redux";
import { logoutUser } from "../actions";

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. Buatlah file dalam folder src/component/ dengan nama protectedRoute.js

```
components > JS 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