Laporan

“Login Firebase”

Disusun sebagai

MATA KULIAH : PBF

Oleh :

Daffa Tahta Akbar 1741720152

Kelas :

3B

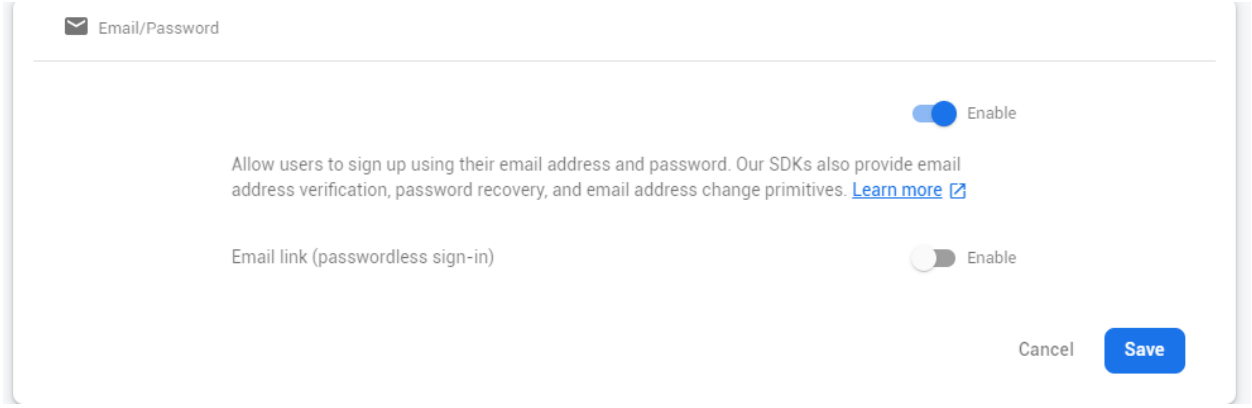
PROGRAM STUDI D-IV TEKNIK INFORMATIKA

JURUSAN TEKNIK INFORMASI

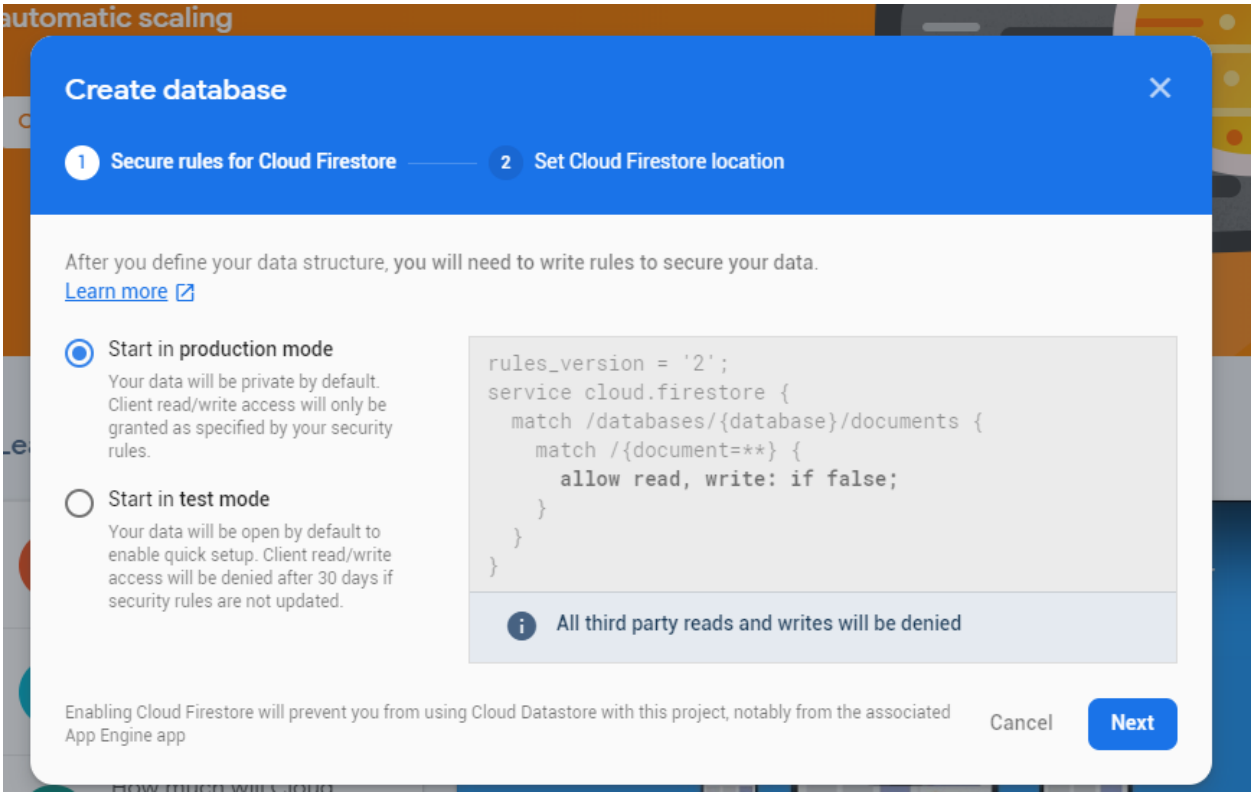
POLITEKNIK NEGERI MALANG

2020

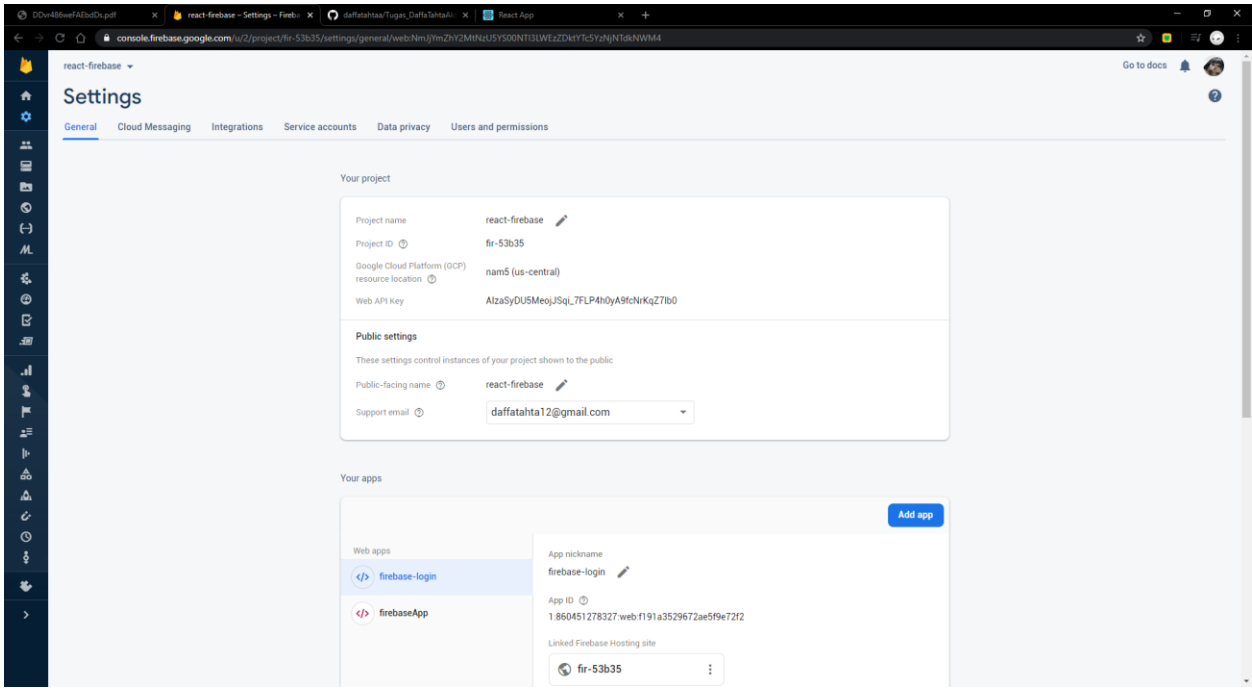
1. Enable login email



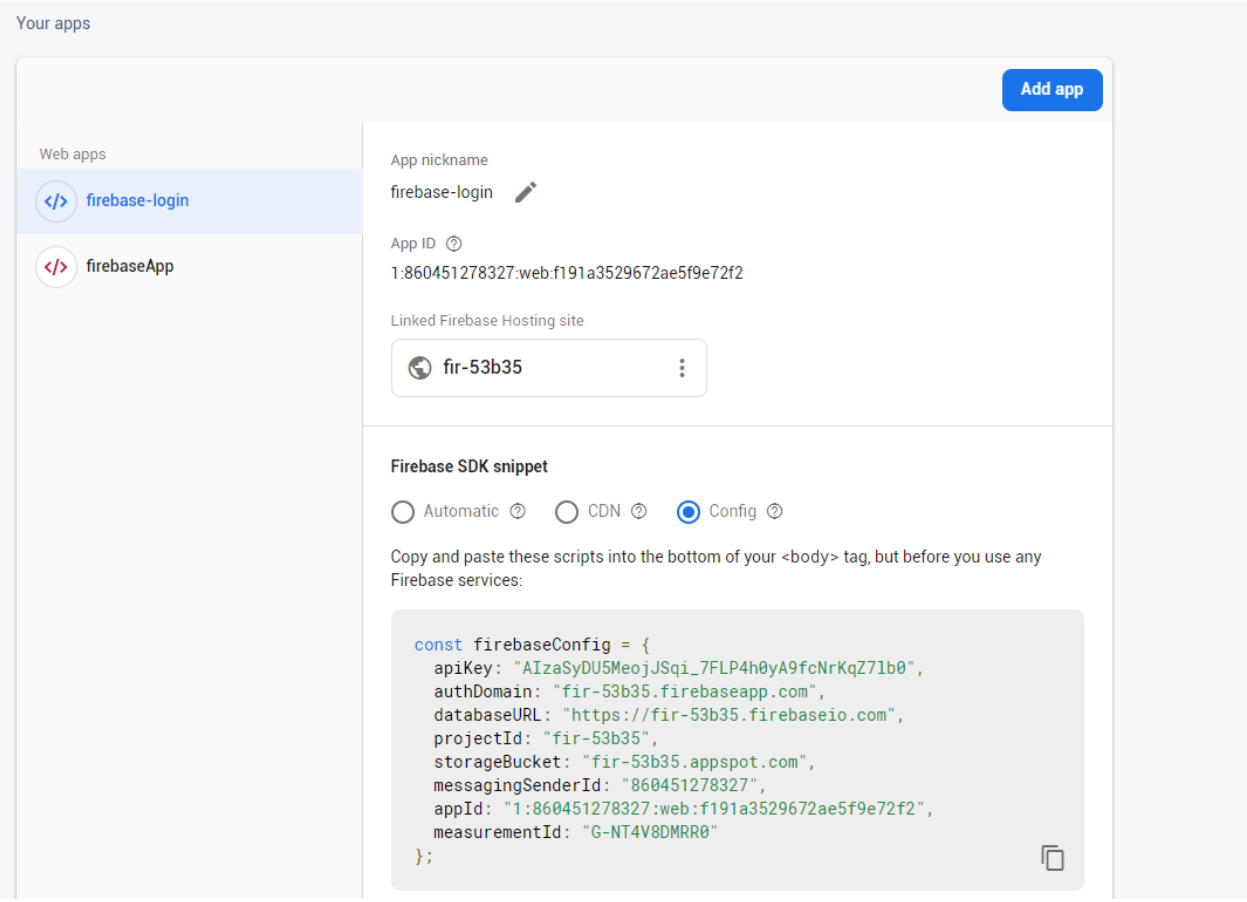
2. Buat database



3. Create new react app
4. Install
    a. Npm install -g firebase-tools
    b. Firebase login
    c. Firebase init
5. Setting untuk firebase dan hosting
    a. Pilih build
    b. Lalu pilih overwrite
6. Install redux
    a. React-router-dom
    b. React-redux
    c. Redux-thunk
    d. Npm I firebase
    e. Npm I @material-ui/core
7. Install material icon
   Npm I @material-ui-icons
8. Setting firebase config
    a. Pilih project setting
    b. Ke bagian </>
    c. Masukkan app nickname
    d. Centang hosting
    e. Click next lalu masuk ke project setting



f. Masuk ke bagian your apps



g. Ke bagian config

h. Copy dan paste di firebase.js

```
import firebase from "firebase/app";
import "firebase/auth";
import "firebase/firestore";

const firebaseConfig = {
  apiKey: "AIzaSyDU5MeojJSqi_7FLP4h0yA9fcNrKqZ7lb0",
  authDomain: "fir-53b35.firebaseapp.com",
  databaseURL: "https://fir-53b35.firebaseio.com",
  projectId: "fir-53b35",
  storageBucket: "fir-53b35.appspot.com",
  messagingSenderId: "860451278327",
  appId: "1:860451278327:web:f191a3529672ae5f9e72f2",
  measurementId: "G-NT4V8DMRR0",
};

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

9. Edit file 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());
  });
};
```

10. Edit reducers/index.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;
  }
};
```

11. Setting redux plumbing
12. Pilih src folder called configureStore.js

```
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;
}
```

13. Pilih dan edit foldersrc dan root

```
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;
```

14. Edit folder src/index.js

```
import React from "react";
import ReactDOM from "react-dom";
import Root from "./root";
import "./index.css";

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();
```

15. Buat 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);
```

16. Buatlah file dan simpan component/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">
```

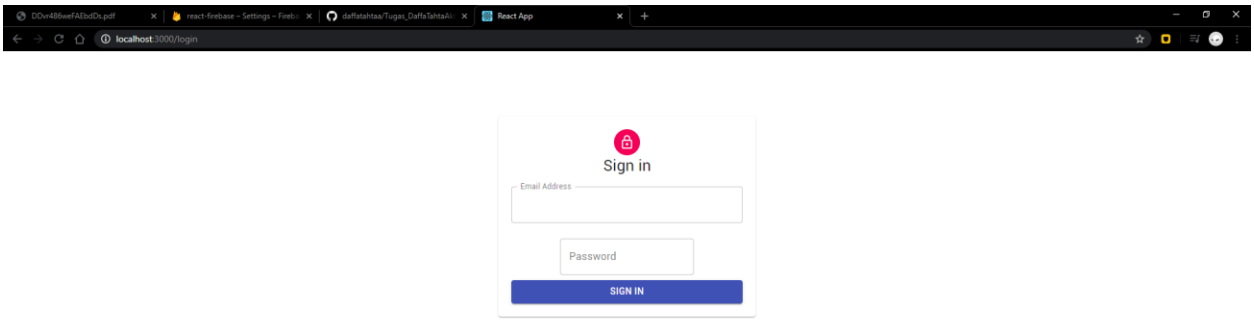
```
              Sign in
            </Typography>
            <TextField
              variant="outlined"
              margin="normal"
              fullWidth
              id="email"
              label="Email Address"
              name="email"
              onChange={this.handleEmailChange}
            />
            <TextField
              variant="outlined"
              margin="normal"
              fullWidthname="password"
              label="Password"
              type="password"
              id="password"
              onChange={this.handlePasswordChange}
            />
            {loginError && (
              <Typography component="p"
classNameclasses.errorText}>
                Incorrect email or password.
              </Typography>
            )}
            <Button
              type="button"
              fullWidth
              variant="contained"
              color="primary"
              classNameclasses.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));
```

17. 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);
```

18. Run



19. Link github: https://github.com/daffatahtaa/Tugas_DaffaTahtaAkbar_1741720152_PBF/tree/master/LoginFirebase/login-firebase