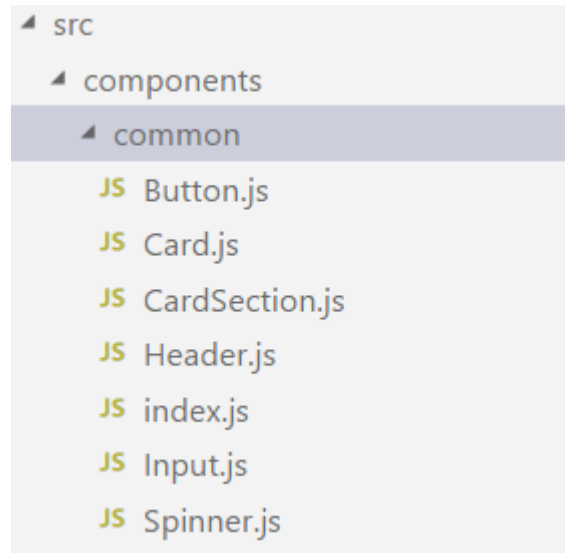


React Native Development

Manager Apps Walkthrough

Step 1

- Copy semua common component yang kita punya di project lalu ke project yang sekarang

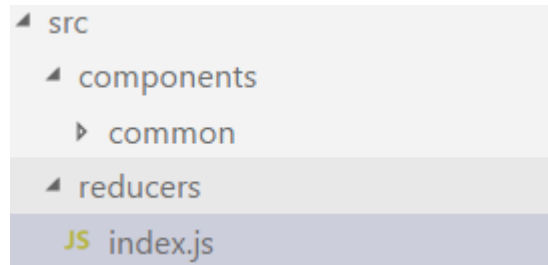


Step 2

- Install Redux dan React Redux dengan mengetik ini di Integrated Terminal
- `npm install --save redux react-redux`

Step 3

- Kemudian buat folder reducers didalam folder src dan buat file index.js didalam folder reducers



- Isi index.js nya

```
import { combineReducers } from 'redux';

export default combineReducers({
  banana: () => []
});
```

Step 4

- Hapus semua code yang ada di App.js, dan diganti dengan ini :

```
import React, { Component } from 'react';
import { Provider } from 'react-redux';
import { View } from 'react-native';
import { createStore } from 'redux';
import { Header } from '../src/components/common';
import reducers from '../src/reducers';

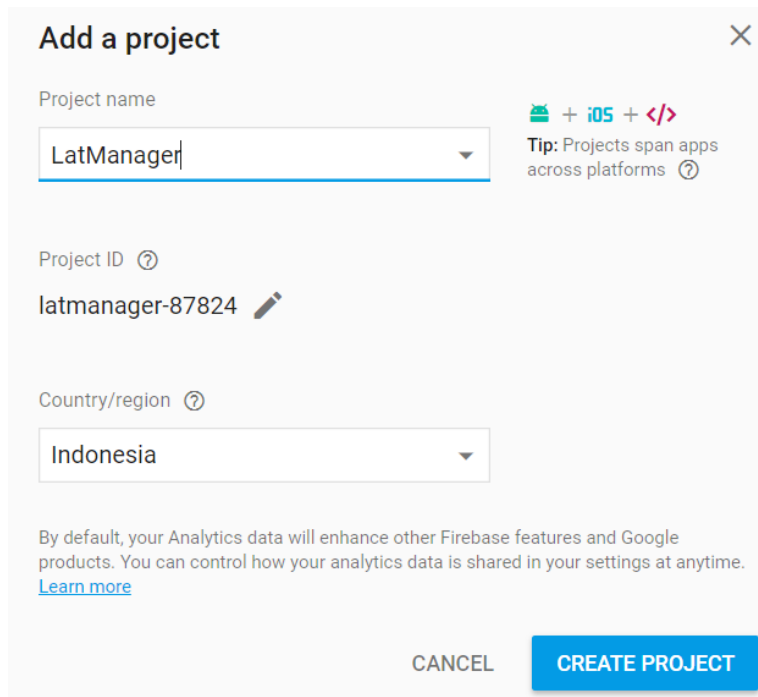
class App extends Component {
  render() {
    const store = createStore(reducers);
    return (
      <Provider store={store}>
        <View>
          <Header headerText="Please Login" />
        </View>
      </Provider>
    );
  }
}

export default App;
```

Please Login

Step 5

- Kemudian install firebase dengan mengetik di Terminal:
`npm install --save firebase`
- Kemudian buka <https://console.firebase.google.com/>
- Lalu Klik Add Project



The screenshot shows the 'Add a project' dialog in the Firebase console. It has a title bar 'Add a project' with a close button. The form contains three main sections: 'Project name' with a dropdown menu showing 'LatManager', 'Project ID' with a text field showing 'latmanager-87824' and a help icon, and 'Country/region' with a dropdown menu showing 'Indonesia'. To the right of the 'Project name' field, there is a small icon showing a mobile phone, '+ iOS', and a code icon, followed by a tip: 'Tip: Projects span apps across platforms' with a help icon. At the bottom, there is a paragraph of text explaining that analytics data will enhance other Firebase features and Google products, with a link to 'Learn more'. At the very bottom, there are two buttons: 'CANCEL' and 'CREATE PROJECT'.

Add a project ✕

Project name

LatManager

+ iOS + </>

Tip: Projects span apps across platforms ?

Project ID ?

latmanager-87824

Country/region ?

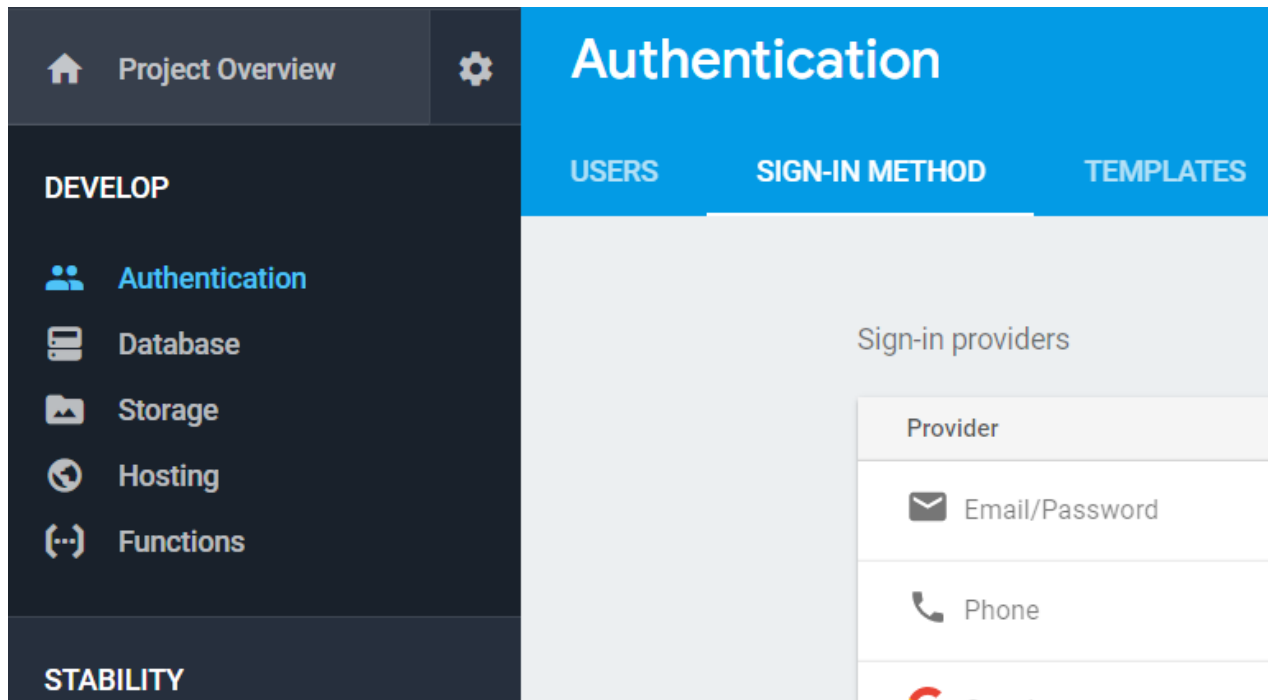
Indonesia

By default, your Analytics data will enhance other Firebase features and Google products. You can control how your analytics data is shared in your settings at anytime. [Learn more](#)

CANCEL CREATE PROJECT

Step 6

- Kemudian buka menu authentication, dan buka tab sign-in method




Step 7

- Kemudian enable email provider

 Email/Password

Enable ☒

Allow users to sign up using their email address and password. Our SDKs also provide email address verification, password recovery, and email address change primitives. [Learn more](#) 

Email link (passwordless sign-in)

Enable ☐

CANCEL

SAVE

Step 8

- Kemudian import firebase di App.js

`import firebase from 'firebase';`

- Kemudian tambahkan lifecycle method `componentWillMount` di App.js

Step 9

- Kemudian balik ke console Firebase dibagian menu authentication, dan pencet button **WEB SETUP** dikanan atas

Add Firebase to your web app

Copy and paste the snippet below at the bottom of your HTML, before other `script` tags.

```
<script src="https://www.gstatic.com/firebasejs/4.12.1/firebase.js"></script>
<script>
  // Initialize Firebase
  var config = {
    apiKey: "AIzaSyB6EcZTf_4ESRePdR6KJ7xQHM_cxiSlfFQ",
    authDomain: "latmanager-87824.firebaseio.com",
    databaseURL: "https://latmanager-87824.firebaseio.com",
    projectId: "latmanager-87824",
    storageBucket: "latmanager-87824.appspot.com",
    messagingSenderId: "41841854914"
  };
  firebase.initializeApp(config);
</script>
```

COPY

Step 10

- Copy dari var config sampai initializeApp, kemudian taru didalam `componentWillMount`(ubah var jadi const, dan kutip 2 jadi kutip 1)

```
componentWillMount() {  
  const config = {  
    apiKey: 'AIzaSyB6EcZTf_4ESRePdR6KJ7xQHM_cxiSlfFQ',  
    authDomain: 'latmanager-87824.firebaseio.com',  
    databaseURL: 'https://latmanager-87824.firebaseio.com',  
    projectId: 'latmanager-87824',  
    storageBucket: 'latmanager-87824.appspot.com',  
    messagingSenderId: '41841854914'  
  };  
  firebase.initializeApp(config);  
}
```

Step 11

- Kemudian buat file LoginForm.js didalam folder components

```
import React, { Component } from 'react';
import { Card, CardSection, Input, Button } from './common';

class LoginForm extends Component {
  render() {
    return (
      <Card>
        <CardSection>
          <Input
            label="Email"
            placeholder="email@gmail.com"
          />
        </CardSection>
      </Card>
    );
  }
}

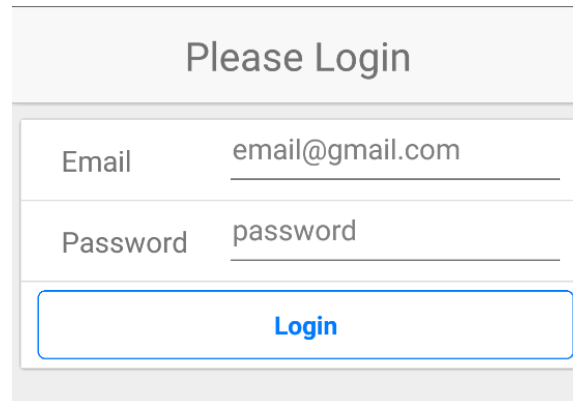
export default LoginForm;
```

Step 11 (continue)

- Kemudian tambahkan 2 CardSection lagi dibawah CardSection input Email. Lalu import dan pasang LoginForm tersebut di App.js

```
<CardSection>
  <Input
    secureTextEntry
    label="Password"
    placeholder="password"
  />
</CardSection>
<CardSection>
  <Button>
    Login
  </Button>
</CardSection>
```

```
<Provider store={store}>
  <View>
    <Header headerText="Please Login" />
    <LoginForm />
  </View>
</Provider>
```



Please Login	
Email	email@gmail.com
Password	password
Login	

Step 12

- Tambahkan function onChange diatas function render

```
class LoginForm extends Component {  
  onChange = (text) => {  
    |  
  }  
  
  render() {
```

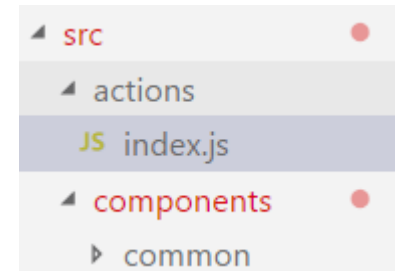
- Kemudian pasang function tersebut ke input Email

```
<Input  
  label="Email"  
  placeholder="email@gmail.com"  
  onChangeText={this.onChange}  
>
```

Step 13

- Buat folder actions didalam folder src
- Kemudian buat file index.js didalam folder actions
- Isi index.js nya

```
export const emailChanged = (text) => {  
  return {  
    type: 'email_changed',  
    payload: text  
  };  
};
```



Step 14

- Import connect dari react-redux dan emailChanged dari actions di LoginForm.js

```
import React, { Component } from 'react';  
import { connect } from 'react-redux';  
import { emailChanged } from '../actions';  
import { Card, CardSection, Input, Button } from '../common';
```

- Kemudian ubah exportnya di LoginForm

```
export default connect(null, { emailChanged })(LoginForm);
```


Step 15

- Kemudian isi function onEmailChange

```
onEmailChange = (text) => {  
  this.props.emailChanged(text);  
}
```

- Kemudian buat file AuthReducer.js didalam folder reducers

```
└─ reducers  
   ├── JS AuthReducer.js  
   └── JS index.js
```

- Lalu import dan pasang di index.js di folder reducers

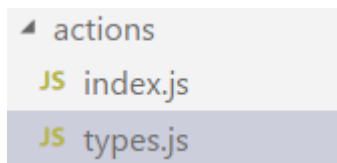
```
import { combineReducers } from 'redux';  
import AuthReducer from './AuthReducer';  
  
export default combineReducers({  
  auth: AuthReducer  
});
```

Step 15 (continue)

- Kemudian isi AuthReducer.js dengan ini

```
const INITIAL_STATE = {  
  email: ''  
};  
  
export default (state = INITIAL_STATE, action) => {  
  switch (action.type) {  
    default:  
      return state;  
  }  
};
```

- Lalu buat file types.js didalam folder actions dan isinya



```
export const EMAIL_CHANGED = 'email_changed';
```

Step 16

- Kemudian import dan pasang types tersebut di index.js actions

```
import { EMAIL_CHANGED } from './types';

export const emailChanged = (text) => {
  return {
    type: EMAIL_CHANGED,
    payload: text
  };
};
```

Step 16 (continue)

- Lalu import juga di AuthReducer.js dan pasang

```
import { EMAIL_CHANGED } from '../actions/types';

const INITIAL_STATE = {
  email: ''
};

export default (state = INITIAL_STATE, action) => {
  switch (action.type) {
    case EMAIL_CHANGED:
      return { ...state, email: action.payload };
    default:
      return state;
  }
};
```

Step 17

- Kemudian tambahkan `mapStateToProps` di `LoginForm.js`

```
const mapStateToProps = (state) => {  
  const { email } = state.auth;  
  
  return { email };  
};  
  
export default connect(mapStateToProps, { emailChanged })(LoginForm);
```

- Lalu pasang props email tersebut ke value di Input Email

```
<Input  
  label="Email"  
  placeholder="email@gmail.com"  
  onChangeText={this.onEmailChange}  
  value={this.props.email}  
>
```

Step 18

- Lalu tambahkan function onPasswordChange dibawah onEmailChange

```
onEmailChange = (text) => {  
  |   this.props.emailChanged(text);  
  |  
  |}
```

```
onPasswordChange = (text) => {  
  |   this.props.passwordChanged(text);  
  |  
  |}
```

- Lalu pasang ke input Passwordnya

```
<Input  
  |   secureTextEntry  
  |   label="Password"  
  |   placeholder="password"  
  |   onChangeText={this.onPasswordChange}  
  |  
  |>
```

Step 19

- Lalu tambahkan di index.js di folder actions (export func passwordChanged)

```
export const emailChanged = (text) => {  
  return {  
    type: EMAIL_CHANGED,  
    payload: text  
  };  
};
```

```
export const passwordChanged = (text) => {  
  return {  
    type: PASSWORD_CHANGED,  
    payload: text  
  };  
};
```

Step 20

- Kemudian tambahkan typesnya di types.js

```
export const EMAIL_CHANGED = 'email_changed';  
export const PASSWORD_CHANGED = 'password_changed';
```

- Lalu import di index.js actions lagi

```
import { EMAIL_CHANGED, PASSWORD_CHANGED } from './types';
```

```
export const emailChanged = (text) => {  
  return {
```


Step 21

- Kemudian tambahkan di AuthReducer.js, import PASSWORD_CHANGED, password di initialState, dan case untuk PASSWORD_CHANGED

```
import { EMAIL_CHANGED, PASSWORD_CHANGED } from '../actions/types';
```

```
const INITIAL_STATE = {  
  email: '',  
  password: ''  
};
```

```
export default (state = INITIAL_STATE, action) => {  
  switch (action.type) {  
    case EMAIL_CHANGED:  
      return { ...state, email: action.payload };  
    case PASSWORD_CHANGED:  
      return { ...state, password: action.payload };  
    default:  
      return state;  
  }  
}
```

Step 22

- Kemudian tambahkan di LoginForm.js import action creator passwordChanged yang sudah kita buat

```
import { emailChanged, passwordChanged } from '../actions';
```

- Lalu pasang di function connect

```
export default connect(mapStateToProps, { emailChanged, passwordChanged })(LoginForm);
```

- Kemudian tambahkan di function mapStateToProps

```
const mapStateToProps = (state) => {  
  const { email, password } = state.auth;  
  
  return { email, password };  
};
```

- Lalu pasang ke input Password kita ke props valuenya

```
onChangeText={this.onPasswordChange}  
value={this.props.password}
```

Step 23

- Tambahkan action creator baru di index.js di folder actions

```
export const loginUser = ({ email, password }) => {  
  
};
```

- Kemudian import firebase di file itu juga

```
import firebase from 'firebase';  
import { EMAIL_CHANGED, PASSWORD_CHANGED } from './types';
```

- Kemudian install redux-thunk dengan mengetik di terminal

```
npm install --save redux-thunk
```

Step 23 (continue)

- Kemudian import ReduxThunk di file App.js kita, dan function applyMiddleware dari Redux

```
import { createStore, applyMiddleware } from 'redux';  
import firebase from 'firebase';  
import ReduxThunk from 'redux-thunk';
```

- Kemudian ubah isi function render() di App.js untuk apply ReduxThunk sebagai middleware kita

```
const store = createStore(reducers, {}, applyMiddleware(ReduxThunk));  
return (  
  <Provider store={store}>  
    <View>  
      <Header headerText="Please Login" />  
    </View>  
  </Provider>  
)
```

Step 24

- Kemudian sekarang isi function action creator loginUser kita

```
export const loginUser = ({ email, password }) => {  
  return (dispatch) => {  
    firebase.auth().signInWithEmailAndPassword(email, password)  
      .then(user => {  
        dispatch({  
          type: LOGIN_USER_SUCCESS,  
          payload: user  
        });  
      });  
  });  
};
```

- Kemudian tambahkan types LOGIN_USER_SUCCESS, lalu import

```
export const EMAIL_CHANGED = 'email_changed';  
export const PASSWORD_CHANGED = 'password_changed';  
export const LOGIN_USER_SUCCESS = 'login_user_success';
```

```
import { EMAIL_CHANGED, PASSWORD_CHANGED, LOGIN_USER_SUCCESS } from './types';
```

Step 25

- Di file LoginForm.js import actions loginUser yang baru kita buat

```
import { connect } from 'react-redux';  
import { emailChanged, passwordChanged, loginUser } from '../actions';
```

- Kemudian sambungkan di function connect()

```
export default connect(mapStateToProps, {  
  emailChanged, passwordChanged, loginUser  
})(LoginForm);
```

- Kemudian tambahkan function ini dibawah onPasswordChange

```
onButtonPress = () => {  
  const { email, password } = this.props;  
  
  this.props.loginUser({ email, password });  
}
```

- Kemudian pasang ke Button Login kita function tersebut

```
<Button onPress={this.onButtonPress}>  
  Login  
</Button>
```

Step 26

- Kemudian import types LOGIN_USER_SUCCESS di AuthReducer.js

```
import {  
  EMAIL_CHANGED,  
  PASSWORD_CHANGED,  
  LOGIN_USER_SUCCESS  
} from '../actions/types';
```

- Lalu tambahkan Case baru LOGIN_USER_SUCCESS

```
case PASSWORD_CHANGED:  
  return { ...state, password: action.payload };  
case LOGIN_USER_SUCCESS:  
  return { ...state, user: action.payload };  

```

- Kemudian tambahkan di initial_state

```
const INITIAL_STATE = {  
  email: '',  
  password: '',  
  user: null  
};
```

Step 27

- Kemudian di index.js di folder actions tambahkan .catch() di function loginUser

```
firebase.auth().signInWithEmailAndPassword(email, password)
  .then(user => {
    dispatch({
      type: LOGIN_USER_SUCCESS,
      payload: user
    });
  })
  .catch((error) => {
    console.log(error);

    firebase.auth().createUserWithEmailAndPassword(email, password)
      .then(user => {
        dispatch({
          type: LOGIN_USER_SUCCESS,
          payload: user
        });
      });
  });
```


Step 28

- Kemudian tambahkan function ini dibawah function loginUser

```
const loginUserSuccess = (dispatch, user) => {  
  dispatch({  
    type: LOGIN_USER_SUCCESS,  
    payload: user  
  });  
};
```

- Lalu edit function loginUser lagi untuk memakai function baru itu

```
firebase.auth().signInWithEmailAndPassword(email, password)  
  .then(user => {  
    loginUserSuccess(dispatch, user);  
  })  
  .catch((error) => {  
    console.log(error);  
  
    firebase.auth().createUserWithEmailAndPassword(email, password)  
      .then(user => {  
        loginUserSuccess(dispatch, user);  
      });  
  });
```

Step 29

- Kemudian tambahkan types baru LOGIN_USER_FAIL di types.js

```
export const LOGIN_USER_SUCCESS = 'login_user_success';  
export const LOGIN_USER_FAIL = 'login_user_fail';  
,
```

- Lalu import ke index.js actions

```
import {  
  EMAIL_CHANGED,  
  PASSWORD_CHANGED,  
  LOGIN_USER_SUCCESS,  
  LOGIN_USER_FAIL  
} from './types';
```

- Baru buat function loginUserFail di index.js actions

```
const loginUserFail = (dispatch) => {  
  dispatch({ type: LOGIN_USER_FAIL });  
};
```

Step 29 (continue)

- Kemudian tambahkan difunction loginUser pasang loginUserFail

```
.catch((error) => {  
  console.log(error);  
  
  firebase.auth().createUserWithEmailAndPassword(email, password)  
    .then(user => {  
      loginUserSuccess(dispatch, user);  
    })  
    .catch(() => loginUserFail(dispatch));  
});
```

Step 30

- Kemudian di AuthReducer.js tambahkan import LOGIN_USER_FAIL

```
import {  
  EMAIL_CHANGED,  
  PASSWORD_CHANGED,  
  LOGIN_USER_SUCCESS,  
  LOGIN_USER_FAIL  
} from '../actions/types';
```

- Lalu tambahkan prop error di INITIAL_STATE

```
const INITIAL_STATE = {  
  email: '',  
  password: '',  
  user: null,  
  error: ''  
};
```

- Kemudian tambah Case baru untuk LOGIN_USER_FAIL

```
case LOGIN_USER_FAIL:  
  return { ...state, error: 'Authentication Failed.' };  
  ...
```

Step 31

- Kemudian di LoginForm.js tambahkan di mapStateToProps ambil state error yang tadi kita buat

```
const mapStateToProps = (state) => {  
  const { email, password, error } = state.auth;  
  
  return { email, password, error };  
};
```

- Lalu tambahkan function renderError() diatas function render(), dan jangan lupa import View dan Text dari library react-native bila belum

```
renderError() {  
  if (this.props.error) {  
    return (  
      <View style={{ backgroundColor: 'white' }}>  
        <Text style={styles.errorTextStyle}>  
          {this.props.error}  
        </Text>  
      </View>  
    );  
  }  
}
```

Step 31 (continue)

- Lalu tambahkan const styles diatas function mapStateToProps

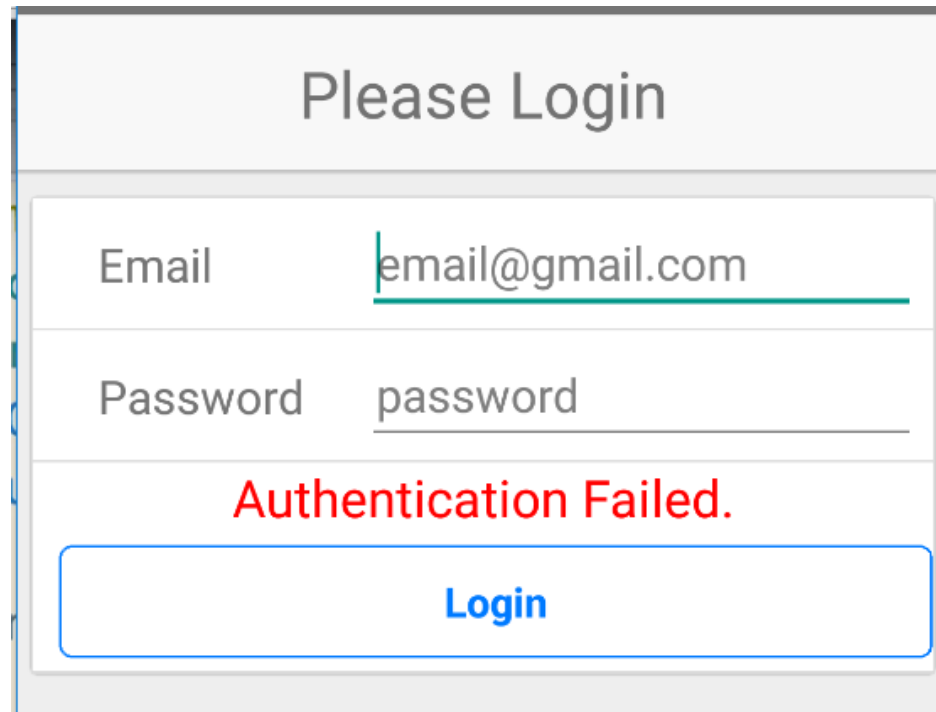
```
const styles = {  
  errorTextStyle: {  
    fontSize: 20,  
    alignSelf: 'center',  
    color: 'red'  
  }  
};
```

- Kemudian pasang function renderError() tersebut

```
</CardSection>  
{this.renderError()}  
<CardSection>  
  <Button onPress={this.onButtonPress}>  
    Login  
  </Button>  
</CardSection>
```

Step 31 (continue)

- Bila kita tes pencet button login sekarang bakal muncul error message tersebut bila terjadi error (masuk ke catch paling dalam)



Please Login

Email

Password

Authentication Failed.

Login

Step 32

- Kemudian tambahkan type LOGIN_USER baru di types.js

```
export const LOGIN_USER_SUCCESS = 'login_user_success';  
export const LOGIN_USER_FAIL = 'login_user_fail';  
export const LOGIN_USER = 'login_user';
```

- Lalu tambahkan import type tersebut di index.js actions

```
LOGIN_USER_FAIL,  
LOGIN_USER  
} from './types';
```

- Dan tambahkan manggil function dispatch tersebut sebelum proses sign in firebase

```
export const loginUser = ({ email, password }) => {  
  return (dispatch) => {  
    dispatch({ type: LOGIN_USER });  
  
    firebase.auth().signInWithEmailAndPassword(email,  
    password).then(() => {  
      dispatch({ type: LOGIN_USER_SUCCESS });  
    }, (error) => {  
      dispatch({ type: LOGIN_USER_FAIL, message: error.message });  
    });  
  };  
};
```


Step 32 (continue)

- Lalu tambahkan import type tersebut juga di AuthReducer

```
    LOGIN_USER_FAIL,  
    LOGIN_USER  
} from '../actions/types';
```

- Kemudian tambahkan bool loading props di INITIAL_STATE

```
const INITIAL_STATE = {  
  email: '',  
  password: '',  
  user: null,  
  error: '',  
  loading: false  
};
```

Step 32 (continue)

- Dan lalu tambahkan case baru untuk LOGIN_USER dan di LOGIN_USER_SUCCESS tambahkan kirim ...INITIAL_STATE, dan tambahkan loading: false di LOGIN_USER_FAIL

```
case LOGIN_USER_SUCCESS:
  return { ...state, ...INITIAL_STATE, user: action.payload };
case LOGIN_USER_FAIL:
  return { ...state, error: 'Authentication Failed.', loading: false };
case LOGIN_USER:
  return { ...state, loading: true, error: '' };
```

Step 33

- Kemudian tambahkan di mapStateToProps di LoginForm loading props

```
const mapStateToProps = (state) => {  
  const { email, password, error, loading } = state.auth;  
  
  return { email, password, error, loading };  
};
```

- Lalu import spinner juga di LoginForm

```
import { Card, CardSection, Input, Button, Spinner } from './common';
```

- Kemudian tambahkan function renderButton() dibawah renderError()

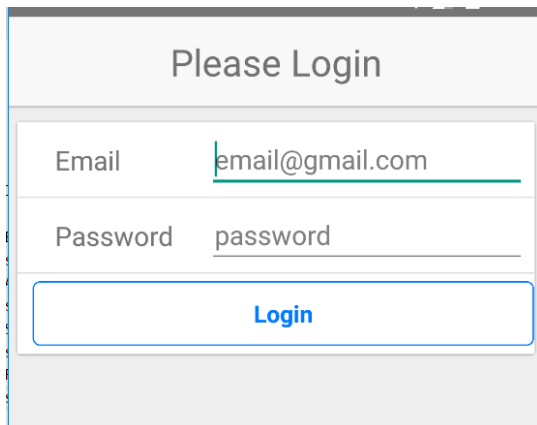
```
renderButton() {  
  if (this.props.loading) {  
    return <Spinner size="large" />;  
  }  
  
  return (  
    <Button onPress={this.onButtonPress}>  
      Login  
    </Button>  
  );  
}
```

Step 33 (continue)

- Lalu pasang function tsb kedalam CardSection menggantikan Button

```
{this.renderError()}  
<CardSection>  
|   {this.renderButton()}  
</CardSection>
```

- Kemudian sekarang saat kita memencet button login dan menunggu respon firebase, button akan berubah menjadi spinner, dan bila berhasil login, error message akan hilang bila ada, isi input text email dan password akan kosong, dan spinner balik ke button login lagi



The image shows a login form with a light gray header containing the text 'Please Login'. Below the header, there are two input fields: 'Email' with the value 'email@gmail.com' and 'Password' with the value 'password'. At the bottom of the form is a blue button labeled 'Login'.

Step 34

- Kemudian install react-navigation library untuk pindah2 page

`npm install --save react-navigation`

- Kemudian install juga react-native-elements dan react-native-vector-icons

`npm install react-native-elements --save`

`npm install react-native-vector-icons --save`

- Kemudian Link React Native Vector Icons package supaya bisa bekerja dengan React Native Elements'

`react-native link react-native-vector-icons`

Step 35

- Kemudian kita akan membuat beberapa component baru (page-page yang diperlukan) yaitu EmployeeList, EmployeeCreate, EmployeeEdit, dan Profile (semuanya file .js didalam folder components).

```
└─ components
  └─ common
    JS EmployeeCreate.js
    JS EmployeeEdit.js
    JS EmployeeList.js
    JS LoginForm.js
    JS Profile.js
```

Step 36

- Kemudian isi masing2 componentnya seperti ini dulu
- Ini EmployeeList component :

```
import React, { Component } from 'react';
import { View, Text } from 'react-native';

class EmployeeList extends Component {
  render() {
    return (
      <View>
        <Text>Ini Page Employee List</Text>
      </View>
    );
  }
}

export default EmployeeList;
```

Step 36 (continue)

- Ini EmployeeCreate Component :

```
import React, { Component } from 'react';
import { View, Text } from 'react-native';

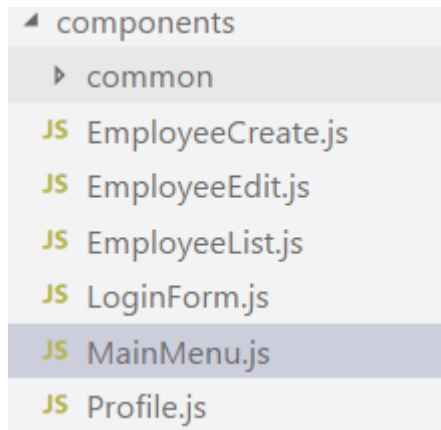
class EmployeeCreate extends Component {
  render() {
    return (
      <View>
        <Text>Ini Page Employee Create</Text>
      </View>
    );
  }
}

export default EmployeeCreate;
```

- Dan component EmployeeEdit dan Profile juga memiliki isi yang sama cuma beda nama classnya dan isi Text yang munculnya.

Step 37

- Kemudian buat file MainMenu.js di folder Components juga (berfungsi untuk sebagai drawer navigation apps ini)



Step 37 (continue)

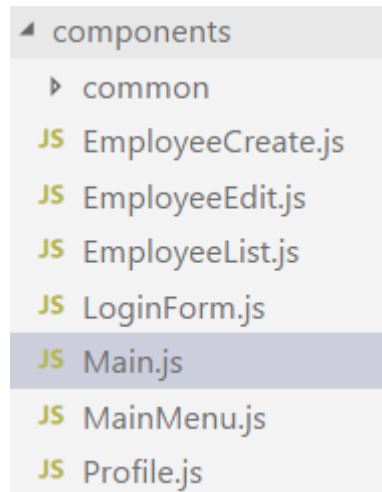
- Kemudian isi file MainMenu.js nya adalah ini

```
import { DrawerNavigator } from 'react-navigation';
import EmployeeList from './EmployeeList';
import EmployeeCreate from './EmployeeCreate';
import EmployeeEdit from './EmployeeEdit';
import Profile from './Profile';
```

```
export default DrawerNavigator(
  {
    EmployeeList: {
      screen: EmployeeList
    },
    AddNewEmployee: {
      screen: EmployeeCreate
    },
    EditEmployee: {
      screen: EmployeeEdit
    },
    Profile: {
      screen: Profile
    }
  },
  {
    initialRouteName: 'EmployeeList',
    headerMode: 'none'
  }
);
```

Step 38

- Kemudian buat file Main.js didalam folder Components juga, (berfungsi untuk sebagai stack navigation apps ini dari LoginForm ke MainMenu).



Step 38 (continue)

- Isi Main.js nya seperti ini

```
import React from 'react';
import { createStackNavigator } from 'react-navigation';
import LoginForm from './LoginForm';
import MainMenu from './MainMenu';

export default createStackNavigator(
  {
    Login: {
      screen: LoginForm
    },
    MainMenu: {
      screen: ({ navigation }) => <MainMenu screenProps={{ rootNavigation: navigation }} />
    }
  },
  {
    initialRouteName: 'Login',
    headerMode: 'none'
  }
);
```

Step 39

- Kemudian import Main component tersebut ke App.js

```
import Main from './src/components/Main';
```

- Lalu pasang didalam Provider (sebagai childrennya)

```
const store = createStore(reducers, {}, applyMiddleware(ReduxThunk));  
return (  
  <Provider store={store}>  
    <Main />  
  </Provider>  
);
```

- Kemudian hapus import2 yang tidak diperlukan

Step 40

- Kemudian import StackActions dan NavigationActions di LoginForm.js
- Kemudian tambahkan get user global state menjadi props di LoginForm dengan menambahkan di mapStateToProps

```
import { StackActions, NavigationActions } from 'react-navigation';

const mapStateToProps = (state) => {
  const { email, password, error, loading, user } = state.auth;

  return { email, password, error, loading, user };
};
```

Step 40 (continue)

- Lalu tambahkan lifecycleMethod `componentWillReceiveProps` juga di `LoginForm` dan isinya ini untuk navigasi pindah page ke `MainMenu`

```
componentWillReceiveProps(newProps) {  
  if (newProps.user) {  
    const resetAction = StackActions.reset({  
      index: 0,  
      actions: [NavigationActions.navigate({ routeName: 'MainMenu' })],  
    });  
    this.props.navigation.dispatch(resetAction);  
  }  
}
```

Step 40 (continue)

- Kemudian import Header dari react-native-elements di LoginForm.js

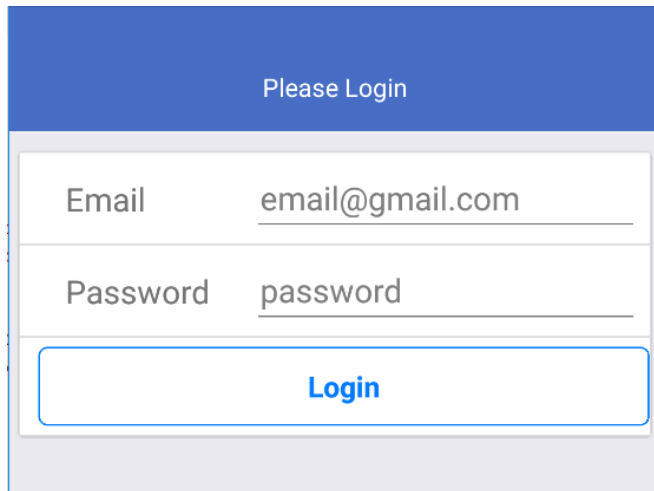
```
import { Header } from 'react-native-elements';
```

- Lalu pasang ke function render() Header tersebut

```
return (  
  <View>  
    <Header  
      placement="left"  
      centerComponent={{ text: 'Please Login', style: { color: '#fff' } }}  
    />  
    <Card>  
      <CardSection>  
        <Input  
          label="Email"  
          placeholder="email@gmail.com"  
        />  
      </CardSection>  
    </Card>  
  </View>  
)
```

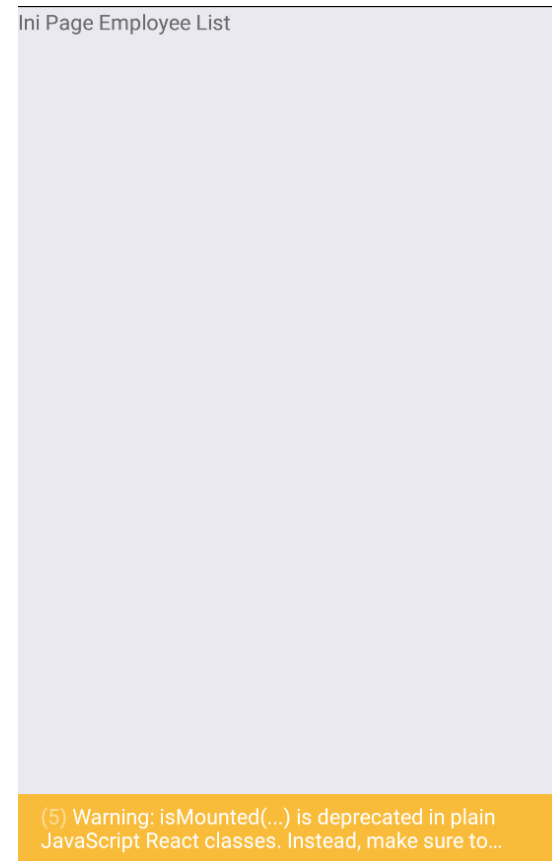

Step 40 (continue)

- Bentuk LoginForm sekarang akan seperti ini



A login form with a blue header bar containing the text "Please Login". Below the header, there are two input fields: "Email" with the value "email@gmail.com" and "Password" with the value "password". At the bottom of the form is a blue "Login" button.

- Dan bila sudah login akan pindah page
Dan muncul page seperti dikanan ini =>



Step 41

- Kemudian tambahkan di App.js render function untuk menghide yellowbox warning

```
render() {  
  const store = createStore(reducers, {}, applyMiddleware(ReduxThunk));  
  console.disableYellowBox = true;  
  return (  
    <Provider store={store}>  
      <Main />  
    </Provider>  
  );  
}
```

Step 42

- Kemudian di EmployeeList.js tambahkan import header dari react-native-elements

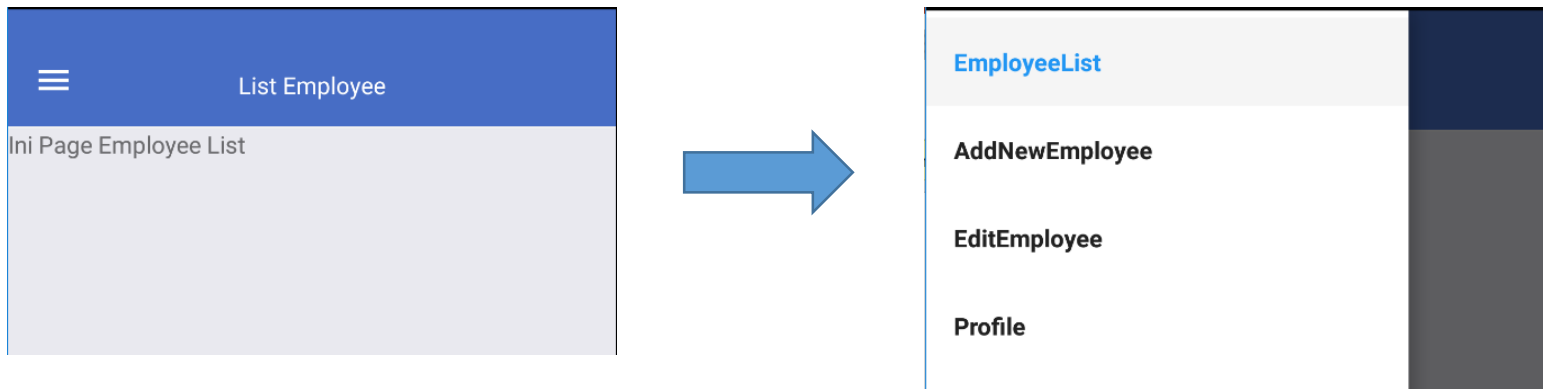
```
import { View, Text } from 'react-native';  
import { Header } from 'react-native-elements';
```

- Lalu kemudian pasang didalam view

```
<View>  
  <Header  
    placement="left"  
    leftComponent={{  
      icon: 'menu',  
      color: '#fff',  
      onPress: () => this.props.navigation.toggleDrawer()  
    }}  
    centerComponent={{ text: 'List Employee', style: { color: '#fff' } }}  
  />  
  <Text>Ini Page Employee List</Text>
```

Step 42 (continue)

- Setelah itu tampilannya akan seperti ini dan bisa navigasi ke page lain



Step 43

- Kemudian tambahkan Header tersebut ke masing2 component yang bisa navigasi (dibawah ini page EmployeeCreate) :

```
<View>
  <Header
    placement="left"
    leftComponent={{
      icon: 'menu',
      color: '#fff',
      onPress: () => this.props.navigation.toggleDrawer()
    }}
    centerComponent={{ text: 'Add Employee', style: { color: '#fff' } }}
    rightComponent={{
      icon: 'home',
      color: '#fff',
      onPress: () => this.props.navigation.navigate('EmployeeList')
    }}
  />
  <Text>Ini Page Employee Create</Text>
```

Step 44

- Ini page EmployeeEdit :

```
<View>
  <Header
    placement="left"
    leftComponent={{
      icon: 'menu',
      color: '#fff',
      onPress: () => this.props.navigation.toggleDrawer()
    }}
    centerComponent={{ text: 'Edit Employee', style: { color: '#fff' } }}
    rightComponent={{
      icon: 'home',
      color: '#fff',
      onPress: () => this.props.navigation.navigate('EmployeeList')
    }}
  />
  <Text>Ini Page Employee Edit</Text>
```

Step 45

- Ini page Profile :

```
<View>
  <Header
    placement="left"
    leftComponent={{
      icon: 'menu',
      color: '#fff',
      onPress: () => this.props.navigation.toggleDrawer()
    }}
    centerComponent={{ text: 'Profile', style: { color: '#fff' } }}
    rightComponent={{
      icon: 'home',
      color: '#fff',
      onPress: () => this.props.navigation.navigate('EmployeeList')
    }}
  />
  <Text>Ini Page Profile</Text>
</View>
```

Step 46

- Kemudian untuk mengganti teks di drawer navigation kita caranya dengan menambahkan static object navigationOptions dengan props drawerLabel yang merubah teksnya (dibawah contoh di component EmployeeList dan component EmployeeCreate) :

```
class EmployeeList extends Component {
  static navigationOptions = {
    drawerLabel: 'Employee List',
  };

  render() {

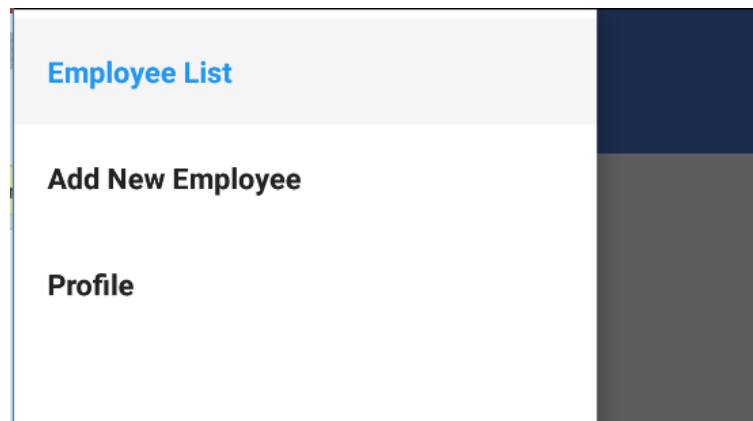
class EmployeeCreate extends Component {
  static navigationOptions = {
    drawerLabel: 'Add New Employee'
  };

  render() {
```


Step 47

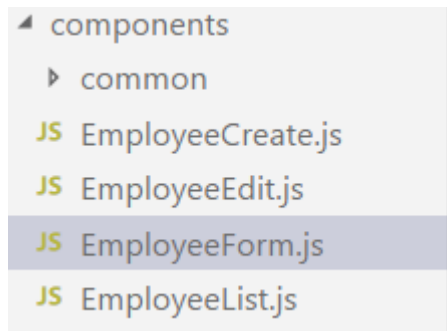
- Dan kemudian bila kita mau menghide sesuatu menu seperti contoh EmployeeEdit kita tidak mau muncul dipilihan tinggal tambahkan drawerLabel yang diisi dengan function yang mereturn null :

```
class EmployeeEdit extends Component {  
  static navigationOptions = {  
    drawerLabel: () => null  
  };  
  
  render() {
```



Step 48

- Kemudian tambahkan file component baru EmployeeForm.js, untuk sebagai form inputnya EmployeeCreate sama EmployeeEdit biar tidak ketik 2 kali / dapat reusable



Step 48 (continue)

- Ini isinya EmployeeForm sementara :

```
import React, { Component } from 'react';
import { View } from 'react-native';
import { CardSection, Input } from './common';

class EmployeeForm extends Component {
  render() {
    return (
      <View>
        <CardSection>
          <Input
            label="Name"
            placeholder="Jane"
          />
        </CardSection>
        <CardSection>
          <Input
            label="Phone"
            placeholder="081287247009"
          />
        </CardSection>
      </View>
    );
  }
}

export default EmployeeForm;
```

Step 48 (continue)

- Kemudian import Picker, dan Text dari react-native di EmployeeForm

```
import { Picker, Text, View } from 'react-native';
```

- Lalu tambahkan CardSection baru dipaling bawah dan isinya picker

```
</CardSection>
<CardSection style={{ flexDirection: 'column' }}>
  <Text>Shift</Text>
  <Picker
    style={{ width: '100%' }}
  >
    <Picker.Item label="Sunday" value="Sunday" />
    <Picker.Item label="Monday" value="Monday" />
    <Picker.Item label="Tuesday" value="Tuesday" />
    <Picker.Item label="Wednesday" value="Wednesday" />
    <Picker.Item label="Thursday" value="Thursday" />
    <Picker.Item label="Friday" value="Friday" />
    <Picker.Item label="Saturday" value="Saturday" />
  </Picker>
</CardSection>
iew>
```

Step 49

- Kemudian import EmployeeForm tersebut di EmployeeCreate beserta import Card, CardSection, dan Button

```
import { Card, CardSection, Button } from './common';  
import EmployeeForm from './EmployeeForm';
```

- Kemudian pasang dibawah component Header di EmployeeCreate

```
    />  
    <Card>  
      <EmployeeForm />  
      <CardSection>  
        <Button>  
          Save  
        </Button>  
      </CardSection>  
    </Card>  
  </View>
```

Add Employee	
Name	Jane
Phone	081287247009
Shift	Sunday
<button>Save</button>	

Step 49 (continue)

- Kemudian tambahkan styles di EmployeeForm dibagian bawah

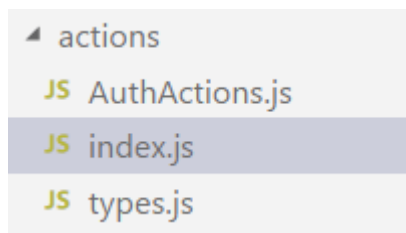
```
const styles = {  
  pickerLabelStyle: {  
    fontSize: 18,  
    paddingLeft: 20  
  }  
};  
  
export default EmployeeForm;
```

- Lalu pasang ke Text Shift Picker

```
<CardSection style={{ flexDirection: 'column' }}>  
  <Text style={styles.pickerLabelStyle}>Shift</Text>  
  <Picker
```

Step 50

- Lalu buat file baru AuthActions.js di dalam folder actions

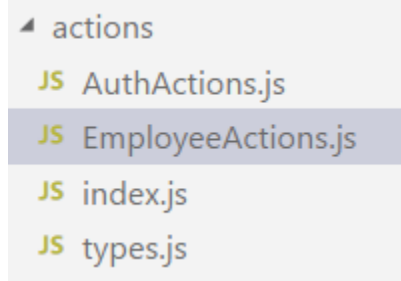


- Kemudian pindahkan semua isi dari index.js di folder actions ke AuthActions.js
- Lalu isi index.jsnya jadi ini saja

```
export * from './AuthActions';
```

Step 51

- Kemudian buat file baru lagi di folder actions namanya EmployeeActions.js



```
actions
  JS AuthActions.js
  JS EmployeeActions.js
  JS index.js
  JS types.js
```

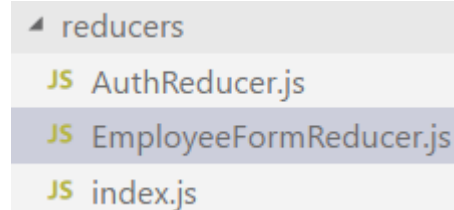
- Kemudian tambahkan di index.js lagi untuk export EmployeeActions jg

```
export * from './AuthActions';
export * from './EmployeeActions';
```


Step 52

- Kemudian difolder reducers tambahkan file baru Namanya EmployeeFormReducer.js dan isinya ini

```
const INITIAL_STATE = {  
  name: '',  
  phone: '',  
  shift: ''  
};  
  
export default (state = INITIAL_STATE, action) => {  
  switch (action.type) {  
    default:  
      return state;  
  }  
};
```



```
reducers  
JS AuthReducer.js  
JS EmployeeFormReducer.js  
JS index.js
```

Step 52 (continue)

- Kemudian di index.js di folder reducers tambahkan state baru

```
import { combineReducers } from 'redux';
import AuthReducer from './AuthReducer';
import EmployeeFormReducer from './EmployeeFormReducer';

export default combineReducers({
  auth: AuthReducer,
  employeeForm: EmployeeFormReducer
});
```

Step 53

- Kemudian tambahkan type baru EMPLOYEE_UPDATE di types.js didalam folder actions

```
export const LOGIN_USER = 'login_user';  
export const EMPLOYEE_UPDATE = 'employee_update';
```

- Lalu sekarang buat action creator employeeUpdate di file EmployeeActions.js difolder actions

```
import { EMPLOYEE_UPDATE } from './types';  
  
export const employeeUpdate = (prop, value) => {  
  return {  
    type: EMPLOYEE_UPDATE,  
    payload: { prop, value }  
  };  
};
```

Step 54

- Kemudian update isi file EmployeeFormReducer.js seperti dibawah ini

```
import {  
  EMPLOYEE_UPDATE  
} from '../actions/types';  
  
const INITIAL_STATE = {  
  name: '',  
  phone: '',  
  shift: ''  
};  
  
export default (state = INITIAL_STATE, action) => {  
  switch (action.type) {  
    case EMPLOYEE_UPDATE:  
      //action.payload === { prop: 'name', value: 'jane' }  
      return { ...state, [action.payload.prop]: action.payload.value };  
    default:  
      return state;  
  }  
};
```

Step 55

- Kemudian sekarang di EmployeeForm.js tambahkan import function connect dari react-redux dan action creator employeeUpdate yang tadi

```
import { Picker, Text, View } from 'react-native';  
import { connect } from 'react-redux';  
import { employeeUpdate } from '../actions';  
import { CardSection, Input } from './common';
```

- Kemudian tambahkan juga function mapStateToProps dan hubungkan semua dengan function connect

```
const mapStateToProps = (state) => {  
  const { name, phone, shift } = state.employeeForm;  
  
  return { name, phone, shift };  
};
```

```
export default connect(mapStateToProps, { employeeUpdate })(EmployeeForm);
```

Step 55 (continue)

- Setelah dihubungkan, sekarang pasang props2 tersebut yang dari mapStateToProps dan juga action creator employeeUpdate kita
- Pertama input Nama :

```
<Input
  label="Name"
  placeholder="Jane"
  value={this.props.name}
  onChangeText={this.onNameChange}
/>
```

- Tambahkan function onNameChange nya diatas render()

```
onNameChange = (text) => {
  this.props.employeeUpdate('name', text);
}
```

```
render() {
```

Step 55 (continue)

- Kemudian input Phonenya :

```
<Input
  label="Phone"
  placeholder="081287247009"
  value={this.props.phone}
  onChangeText={this.onPhoneChange}
/>
```

- Tambahkan function onPhoneChange diatas render() dan dibawah onNameChange

```
onPhoneChange = (text) => {
  |   this.props.employeeUpdate('phone', text);
}

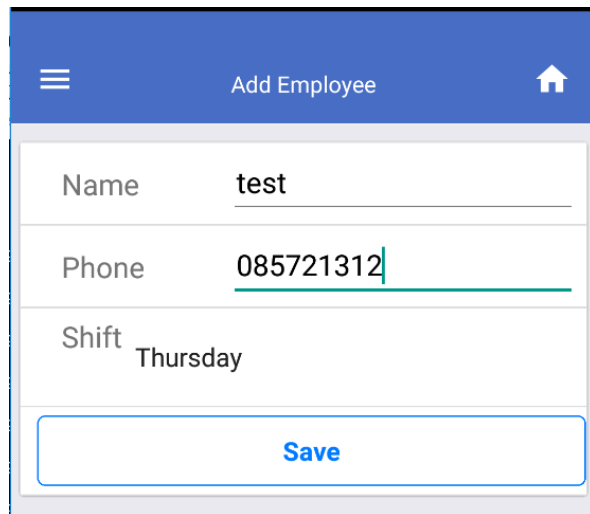
render() {
```

Step 55 (continue)

- Kemudian sekarang Picker Shiftnya

```
<Picker
  selectedValue={this.props.shift}
  onValueChange={day => this.props.employeeUpdate('shift', day)}
  style={{ width: '100%' }}
/>
```

- Sekarang EmployeeForm kita sudah terhubung dengan globalState



Add Employee	
Name	test
Phone	085721312
Shift	Thursday
<button>Save</button>	

Step 56

- Kemudian sekarang kita akan membuat function untuk ngesave data employee baru kita ke firebase, karena itu sekarang tambahkan function action creator baru difile EmployeeActions.js.

- Sebelum buat functionnya, kita harus import library firebase dulu:

```
import firebase from 'firebase';  
import { EMPLOYEE_UPDATE } from './types';
```

- Ini functionnya :

```
export const employeeCreate = (name, phone, shift) => {  
  const { currentUser } = firebase.auth();  
  
  return (dispatch) => {  
    firebase.database().ref(`/users/${currentUser.uid}/employees`)  
      .push({ name, phone, shift })  
      .then(() => {  
        dispatch({ type: 'employee_create' });  
      });  
  };  
};
```

Step 57

- Tambahkan types baru EMPLOYEE_CREATE di types.js

```
export const EMPLOYEE_UPDATE = 'employee_update';  
export const EMPLOYEE_CREATE = 'employee_create';
```

- Kemudian import ke EmployeeActions.js

```
import firebase from 'firebase';  
import { EMPLOYEE_UPDATE, EMPLOYEE_CREATE } from './types';
```

- Kemudian pasang ke props type tadi yang diisi string employee_create

```
return (dispatch) => {  
  firebase.database().ref(`/users/${currentUser.uid}/employees`)  
    .push({ name, phone, shift })  
    .then(() => {  
      dispatch({ type: EMPLOYEE_CREATE });  
    });  
}
```

Step 58

- Lalu kemudian import types EMPLOYEE_CREATE tersebut di EmployeeFormReducer.js

```
import {  
  |   EMPLOYEE_UPDATE, EMPLOYEE_CREATE  
} from '../actions/types';
```

- Kemudian tambahkan case baru untuk EMPLOYEE_CREATE tersebut dibawah EMPLOYEE_UPDATE (case ini berguna untuk menghapus bersih kembali input2 dari EmployeeForm)

```
case EMPLOYEE_CREATE:  
  |   return INITIAL_STATE;  
default:  
  |   return state;
```

Step 59

- Kemudian import function connect dari react-redux dan action creator yang baru kita buat di EmployeeCreate.js

```
import { Header } from 'react-native-elements';  
import { connect } from 'react-redux';  
import { employeeCreate } from '../actions';
```

- Lalu buat function mapStateToProps untuk get state name,phone,shift dari global state untuk proses save, dan connect mapStateToProps kita beserta actionCreator kita ke component EmployeeCreate kita

```
const mapStateToProps = (state) => {  
  const { name, phone, shift } = state.employeeForm;  
  
  return { name, phone, shift };  
};
```

```
export default connect(mapStateToProps, { employeeCreate })(EmployeeCreate);
```

Step 60

- Kemudian tambahkan function `onButtonPress` untuk button Save kita di `EmployeeCreate` diatas function `render()`

```
onButtonPress = () => {  
  const { name, phone, shift } = this.props;  
  
  this.props.employeeCreate(name, phone, shift || 'Sunday');  
}
```

```
render() {
```

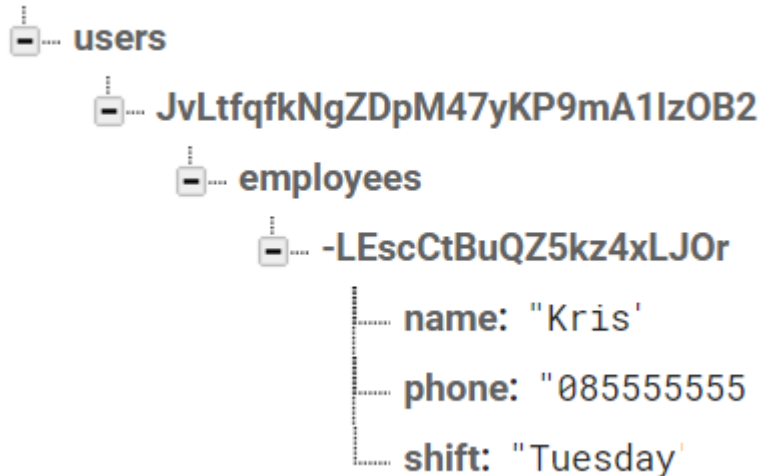
- Lalu pasang function tersebut ke event handler button save `EmployeeCreate`

```
<EmployeeForm />  
<CardSection>  
  <Button onPress={this.onButtonPress}>  
    Save  
  </Button>  
</CardSection>
```

Step 61

- Sekarang app kita sudah bisa save data new Employee ke database firebase, ini contoh isi database firebase saya (saya save new Employee dengan nama Kris, phonenya 085555555, dan shiftnya Tuesday)

latmanager-87824 + ×



Step 62

- Sekarang kita akan mulai mengerjakan component EmployeeList kita, langkah pertama adalah buat action Creator baru di EmployeeActions.js untuk ngeget data list Employee kita dari firebase

```
export const getEmployeeList = () => {  
  const { currentUser } = firebase.auth();  
  
  return (dispatch) => {  
    firebase.database().ref(`/users/${currentUser.uid}/employees`)  
      .on('value', snapshot => {  
        dispatch({ type: EMPLOYEES_GETLIST_SUCCESS, payload: snapshot.val() });  
      });  
  };  
};
```

Step 62 (continue)

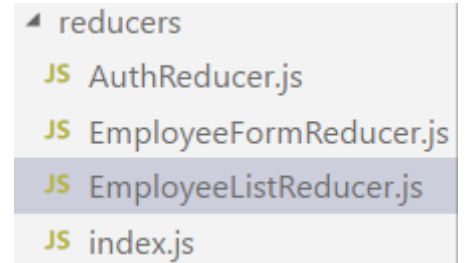
- Tambahkan type EMPLOYEES_GETLIST_SUCCESS tersebut di types.js

```
export const EMPLOYEE_CREATE = 'employee_create';  
export const EMPLOYEES_GETLIST_SUCCESS = 'employees_getlist_success';
```

- Lalu import ke EmployeeActions.js

```
import { EMPLOYEE_UPDATE, EMPLOYEE_CREATE, EMPLOYEES_GETLIST_SUCCESS } from './types';
```


Step 63



- Kemudian buat file baru EmployeeListReducer.js didalam folder reducers dan isinya ini

```
import { EMPLOYEES_GETLIST_SUCCESS } from '../actions/types';

const INITIAL_STATE = {};

export default (state = INITIAL_STATE, action) => {
  switch (action.type) {
    case EMPLOYEES_GETLIST_SUCCESS:
      return action.payload;
    default:
      return state;
  }
};
```

Step 64

- Lalu import EmployeeListReducer tersebut di index.js folder reducers dan buat global state baru :

```
import EmployeeListReducer from './EmployeeListReducer';

export default combineReducers({
  auth: AuthReducer,
  employeeForm: EmployeeFormReducer,
  employees: EmployeeListReducer
});
```

Step 65

- Sekarang install lodash dengan mengetik di terminal
`npm install --save lodash`
- Lalu di EmployeeList.js component kita tambahkan import connect, listview, lodash, dan getEmployeeList action creator kita

```
import React, { Component } from 'react';  
import { ListView, View, Text } from 'react-native';  
import _ from 'lodash';  
import { connect } from 'react-redux';  
import { Header } from 'react-native-elements';  
import { getEmployeeList } from '../actions';
```

Step 66

- Lalu tambahkan function `mapStateToProps` dan hubungkan dengan `connect` beserta action creator kita ke `EmployeeList`

```
const mapStateToProps = (state) => {  
  const employees = _.map(state.employees, (val, uid) => {  
    return { ...val, uid };  
  });  
  
  return { employees };  
};  
  
export default connect(mapStateToProps, { getEmployeeList })(EmployeeList);
```

Step 67

- Kemudian buat function createDataSource dan lifecyclemethod componentWillMount buat isi datasource ListView kita nanti di EmployeeList

```
componentWillMount() {  
  this.props.getEmployeeList();  
  
  this.createDataSource(this.props);  
}  
  
createDataSource({ employees }) {  
  const ds = new ListView.DataSource({  
    rowHasChanged: (r1, r2) => r1 !== r2  
  });  
  
  this.dataSource = ds.cloneWithRows(employees);  
}
```

Step 67 (continue)

- Lalu tambahkan juga lifecycleMethod `componentWillReceiveProps` di `EmployeeList` dibawah `componentWillMount()`

```
componentWillReceiveProps(nextProps) {  
    //nextProps are the next set of props that this component  
    //will be rendered with  
    //this.props is still the old set of props  
    this.createDataSource(nextProps);  
}
```

Step 68

- Sekarang buat file component baru EmployeeListItem.js di folder components untuk renderRow ListView kita, dan isinya ini :

```
import React, { Component } from 'react';
import { Text, View } from 'react-native';
import { CardSection } from '../common';

class EmployeeListItem extends Component {
  render() {
    const { name } = this.props.employee;

    return (
      <View>
        <CardSection>
          <Text>{name}</Text>
        </CardSection>
      </View>
    );
  }
}

export default EmployeeListItem;
```

Step 69

- Kemudian import `EmployeeListItem` ke `EmployeeList.js`

```
import EmployeeListItem from './EmployeeListItem';
```

- Lalu tambahkan function `renderRow()` diatas `render()` di `EmployeeList`

```
renderRow = (employee) => {  
  |   return <EmployeeListItem employee={employee} />;  
  |  
  }  
}
```

- Kemudian sekarang pasang `ListView` kita ke function `render()` dibawah Header kita menggantikan text Ini page `EmployeeList`

```
<ListView  
  |   enableEmptySections  
  |   dataSource={this.dataSource}  
  |   renderRow={this.renderRow}  
  |  
  />
```

List Employee	
☰	
Kris	
Baron	
Test	

Step 70

- Sekarang tambahkan style untuk Text diEmployeeListItem.js

```
    <Text style={styles.nameStyle}>{name}</Text>
  </CardSection>
</View>
);
}
}

const styles = {
  nameStyle: {
    fontSize: 18,
    paddingLeft: 15
  }
};
```

List Employee	
☰	
Kris	
Baron	
Test	

Step 71

- Kemudian sekarang import diEmployeeListItem.js jadi sebanyak itu

```
import React, { Component } from 'react';
import { Text, View, TouchableWithoutFeedback } from 'react-native';
import _ from 'lodash';
import { connect } from 'react-redux';
import { CardSection } from '../common';
import { employeeUpdate } from '../actions';
```

- Kemudian hubungkan action creatornya dengan function connect

```
export default connect(null, { employeeUpdate })(EmployeeListItem);
```

- Kemudian tambahkan function ini diatas render()

```
onRowPress = () => {
  _.each(this.props.employee, (value, prop) => {
    this.props.employeeUpdate(prop, value);
  });
  this.props.navigation.navigate('EditEmployee');
}
```

Step 71 (continue)

- Kemudian tambahkan Touchable di function render() seperti ini :

```
<TouchableWithoutFeedback onPress={this.onRowPress}>
  <View>
    <CardSection>
      <Text style={styles.nameStyle}>{name}</Text>
    </CardSection>
  </View>
</TouchableWithoutFeedback>
```

- Kemudian tambahkan di EmployeeList.js di function renderRow() untuk kirim props navigation juga

```
renderRow = (employee) => {
  return <EmployeeListItem employee={employee} navigation={this.props.navigation} />;
}
```

- Sekarang begitu salah satu EmployeeListItem dipencet, dia akan pindah page ke EmployeeEdit dan mengisi global state employeeForm

Step 72

- Sekarang tambahkan import di EmployeeEdit seperti dibawah ini:

```
import { Card, CardSection, Button } from './common';  
import EmployeeForm from './EmployeeForm';
```

- Lalu tambahkan UI code tersebut di render() dibawah Header

```
<Card>  
  <EmployeeForm />  
  <CardSection>  
    <Button>  
      Save  
    </Button>  
  </CardSection>  
  <CardSection>  
    <Button>  
      Text Schedule  
    </Button>  
  </CardSection>  
  <CardSection>  
    <Button>  
      Fire  
    </Button>  
  </CardSection>  
</Card>
```

Step 73

- Kemudian sekarang tambahkan action creator baru di EmployeeActions.js untuk update data Employee

```
export const employeeSave = (name, phone, shift, uid) => {  
  const { currentUser } = firebase.auth();  
  
  return (dispatch) => {  
    firebase.database().ref(`/users/${currentUser.uid}/employees/${uid}`)  
      .set({ name, phone, shift })  
      .then(() => {  
        dispatch({ type: EMPLOYEE_CREATE });  
      });  
  };  
};
```

Step 74

- Di EmployeeEdit.js kemudian tambahkan import ini

```
import { connect } from 'react-redux';  
import { employeeSave } from '../actions';
```

- Lalu tambahkan function mapStateToProps dan connect in beserta action creator ke component EmployeeEdit

```
const mapStateToProps = (state) => {  
  const { name, phone, shift, uid } = state.employeeForm;  
  
  return { name, phone, shift, uid };  
};  
  
export default connect(mapStateToProps, { employeeSave })(EmployeeEdit);
```

Step 75

- Kemudian tambahkan function ini diatas render() di EmployeeEdit

```
onButtonPress = () => {  
  const { name, phone, shift, uid } = this.props;  
  
  this.props.employeeSave(name, phone, shift, uid);  
}
```

- Lalu pasang ke button Save kita

```
<EmployeeForm />  
<CardSection>  
  <Button onPress={this.onButtonPress}>  
    Save  
  </Button>  
</CardSection>
```

- Sekarang kita sudah bisa update data Employee

Step 76

- Sekarang tambahkan action creator ini di EmployeeActions.js

```
export const employeeDelete = (uid) => {  
  const { currentUser } = firebase.auth();  
  
  return (dispatch) => {  
    firebase.database().ref(`/users/${currentUser.uid}/employees/${uid}`)  
      .remove()  
      .then(() => {  
        dispatch({ type: EMPLOYEE_CREATE });  
      });  
  };  
};
```

- Kemudian import ke EmployeeEdit lagi dan pasang juga ke connect

```
import { employeeSave, employeeDelete } from '../actions';  
export default connect(mapStateToProps, { employeeSave, employeeDelete })(EmployeeEdit);
```


Step 77

- Kemudian import alert dari react-native di EmployeeEdit.js

```
import { Alert, View } from 'react-native';
```

- Lalu tambahkan 2 function ini di EmployeeEdit.js diatas render()

```
onButtonFirePress = () => {  
  //this.setState({ showModal: true });  
  Alert.alert(  
    'Are you sure to fire him/her?',  
    '',  
    [  
      { text: 'No', onPress: () => {}, style: 'cancel' },  
      { text: 'Yes', onPress: this.onAccept },  
    ],  
    { cancelable: false }  
  );  
}
```

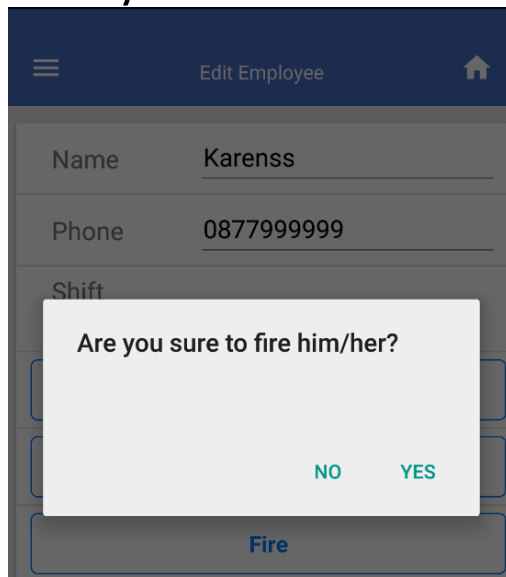
```
onAccept = () => {  
  this.props.employeeDelete(this.props.uid);  
}
```

Step 77 (continue)

- Kemudian pasang function tersebut ke button Fire

```
<CardSection>
  <Button onPress={this.onButtonFirePress}>
    Fire
  </Button>
</CardSection>
```

- Sekarang sudah bisa delete data Employee bila pencet fire button dan click yes



Step 78

- Kemudian sekarang ketik di terminal untuk install

`npm install --save react-native-communications`

- Kemudian import ke EmployeeEdit.js

```
import { text } from 'react-native-communications';
```

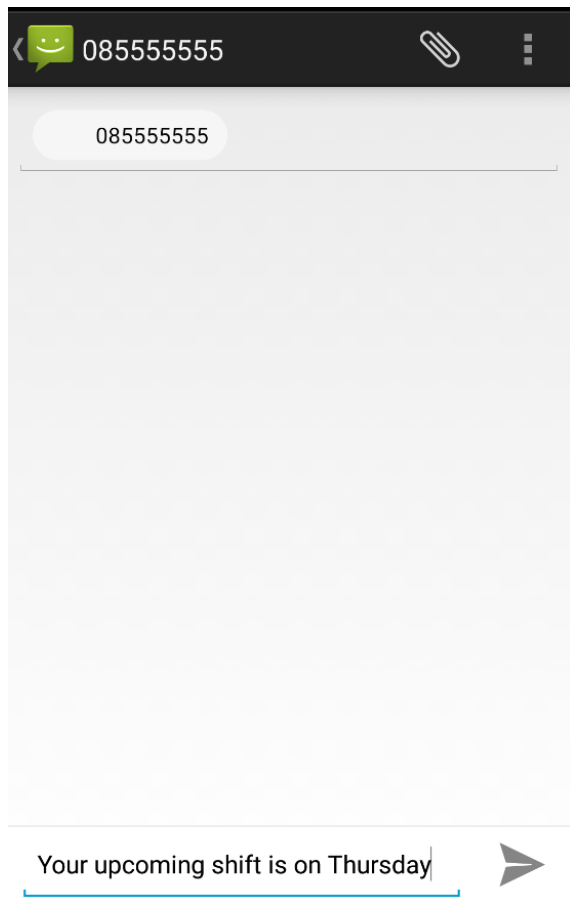
- Kemudian tambahkan function ini diatas render() dan pasang ke button Text Schedule

```
onButtonTextPress = () => {  
  const { phone, shift } = this.props;  
  
  text(phone, `Your upcoming shift is on ${shift}`);  
}
```

```
<Button onPress={this.onButtonTextPress}>  
  Text Schedule  
</Button>
```

Step 78 (continue)

- Sekarang kalau button Text Schedule ditekan akan membuka message apps dan terisi nomornya dan text messagesnya



Step 79

- Kemudian sekarang tambahkan import di Profile.js

```
import { connect } from 'react-redux';  
import { Card, CardSection, Button } from './common';
```

- Terus tambahkan mapStateToProps dan hubungkan dengan connect

```
const mapStateToProps = (state) => {  
  const { user } = state.auth;  
  
  return { user };  
};  
  
export default connect(mapStateToProps)(Profile);
```

Step 79 (continue)

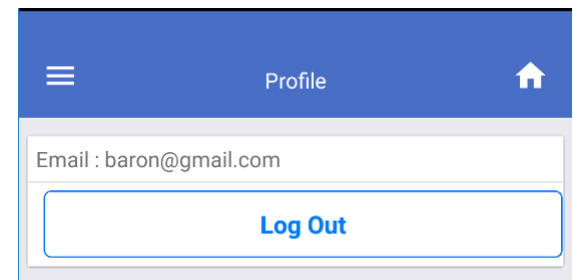
- Terus tambahkan state dan componentWillMount ini di Profile.js

```
state = { email: '' }
```

```
componentWillMount() {  
  if (this.props.user) {  
    this.setState({ email: this.props.user.email });  
  }  
}
```

- Lalu tambahkan code ini dibawah Header didalam render()

```
<Card>  
  <CardSection>  
    <Text>Email : {this.state.email}</Text>  
  </CardSection>  
  <CardSection>  
    <Button>  
      Log Out  
    </Button>  
  </CardSection>  
</Card>
```



Step 80

- Kemudian di types.js tambahkan type baru yaitu LOGOUT_USER

```
export const LOGIN_USER = 'login_user';  
export const LOGOUT_USER = 'logout_user';
```

- Kemudian import type tersebut ke AuthActions.js, lalu tambahkan export action creator baru juga logoutUser()

```
export const logoutUser = () => {  
  return (dispatch) => {  
    firebase.auth().signOut();  
    dispatch({ type: LOGOUT_USER });  
  };  
};
```

- Lalu import juga type tersebut ke AuthReducer.js dan tambahkan case baru LOGOUT_USER

```
case LOGOUT_USER:  
  return INITIAL_STATE;
```

Step 81

- Kemudian import action creator tersebut di Profile.js

```
import { logoutUser } from '../actions';
```

- Lalu pasang ke function connect()

```
export default connect(mapStateToProps, { logoutUser })(Profile);
```

- Kemudian tambahkan function baru ini diatas function render()

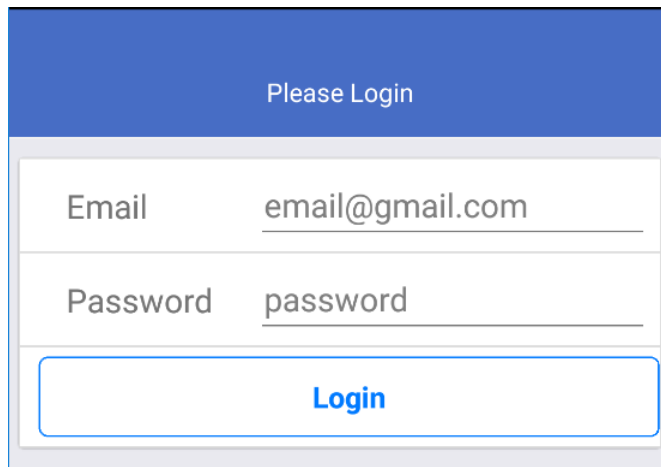
```
logout = () => {  
  this.props.logoutUser();  
  this.props.screenProps.rootNavigation.navigate('Login');  
}
```

- Lalu pasang ke button Log Out

```
<Button onPress={this.logout}>  
  Log Out  
</Button>
```


Step 82

- Sekarang sudah bisa logout bila pencet button logout dipage profile akan logout dan redirect ke page login lagi



Please Login

Email

Password

[Login](#)

Step 83

- Sekarang tambahkan action creator ini di AuthActions.js

```
export const alreadyLogin = (user) => {  
  return {  
    type: LOGIN_USER_SUCCESS,  
    payload: user  
  };  
};
```

- Lalu buat file component baru AppInit.js di folder components

```
JS AppInit.js  
JS EmployeeCreate.js  
JS EmployeeEdit.js  
JS EmployeeForm.js
```

Step 84

- Isi Applnit.js, ini importnya :

```
import React, { Component } from 'react';  
import { connect } from 'react-redux';  
import firebase from 'firebase';  
import { alreadyLogin } from '../actions';  
import Main from './Main';
```

Step 84 (continue)

- Kemudian isi component AppInitnya ada componentWillMount() yang dipindahkan dari App.js dan ditambahkan untuk mengecek sudah login atau belum

```
class AppInit extends Component {
  componentWillMount() {
    const config = {
      apiKey: 'AIzaSyB6EcZTf_4ESRePdR6KJ7xQHM_cxiSlfFQ',
      authDomain: 'latmanager-87824.firebaseio.com',
      databaseURL: 'https://latmanager-87824.firebaseio.com',
      projectId: 'latmanager-87824',
      storageBucket: 'latmanager-87824.appspot.com',
      messagingSenderId: '41841854914'
    };
    firebase.initializeApp(config);

    firebase.auth().onAuthStateChanged((user) => {
      if (user) {
        this.props.alreadyLogin(user);
      }
    });
  }
}
```

Step 84 (continue)

- Dibawah componentWillMount() isinya ini:

```
render() {  
  return (  
    <Main />  
  );  
}  
}  
  
export default connect(null, { alreadyLogin })(AppInit);
```

Step 85

- Isi App.js jadi ini saja

```
import React, { Component } from 'react';
import { Provider } from 'react-redux';
import { createStore, applyMiddleware } from 'redux';
import ReduxThunk from 'redux-thunk';
import reducers from './src/reducers';
import AppInit from './src/components/AppInit';

class App extends Component {
  render() {
    const store = createStore(reducers, {}, applyMiddleware(ReduxThunk));
    console.disableYellowBox = true;
    return (
      <Provider store={store}>
        <AppInit />
      </Provider>
    );
  }
}

export default App;
```

Step 86

- Sekarang apps kita bila kita sudah login, gaperlu login lagi kalau buka appsnya lagi
- Lalu sekarang kita akan membuat apps kita loginform atau mainmenu tidak akan muncul melainkan spinner saat sedang melakukan verifikasi ke firebase apakah user sudah login atau belum

Step 87

- Tambahkan type baru di types.js NOT_LOGIN_YET

```
export const LOGOUT_USER = 'logout_user';  
export const NOT_LOGIN_YET = 'not_login_yet';
```

- Kemudian import type tersebut ke AuthActions.js dan buat action creator baru

```
export const notLoginYet = () => {  
  return {  
    type: NOT_LOGIN_YET,  
    payload: 'Belum Login'  
  };  
};
```


Step 88

- Kemudian tambahkan di AuthReducer.js InitialState status props

```
const INITIAL_STATE = {  
  email: '',  
  password: '',  
  user: null,  
  error: '',  
  loading: false,  
  status: ''  
};
```

- Lalu import type NOT_LOGIN_YET dan tambahkan case baru

```
case NOT_LOGIN_YET:  
  return { ...state, ...INITIAL_STATE, status: action.payload };
```

Step 89

- Kemudian import action creator tersebut ke AppInit.js

```
import { alreadyLogin, notLoginYet } from '../actions';
```

- dan sambungkan ke connect

```
export default connect(null, { alreadyLogin, notLoginYet })(AppInit);
```

- Lalu tambahkan else di function onAuthStateChanged di componentWillMount()

```
firebase.auth().onAuthStateChanged((user) => {  
  if (user) {  
    this.props.alreadyLogin(user);  
  } else {  
    this.props.notLoginYet();  
  }  
});
```

Step 90

- Tambahkan di mapStateToProps di LoginForm.js untuk ngeget status state yang baru kita buat

```
const mapStateToProps = (state) => {  
  const { email, password, error, loading, user, status } = state.auth;  
  
  return { email, password, error, loading, user, status };  
};
```

Step 90 (continue)

- Kemudian tambahkan state dan edit componentWillReceiveProps di LoginForm jg

```
class LoginForm extends Component {
  state = { statusChecked: false }

  componentWillReceiveProps(newProps) {
    if (newProps.user) {
      const resetAction = StackActions.reset({
        index: 0,
        actions: [NavigationActions.navigate({ routeName: 'MainMenu' })],
      });
      this.props.navigation.dispatch(resetAction);
    }
    this.setState({ statusChecked: true });
  }
}
```

Step 90 (continue)

- Kemudian tambahkan function renderContent() di LoginForm dan isinya ini, 1 tag Card dari render() dipindahkan ke return renderContent()

```
renderContent() {  
  if (this.state.statusChecked === false) {  
    return <Spinner size="large" />;  
  }  
  
  return (  
    <Card>  
      <CardSection>  
        <Input  
          label="Email"  
          placeholder="email@gmail.com"  
          onChangeText={this.onEmailChange}  
          value={this.props.email}  
        />  
      </CardSection>  
      <CardSection>  
        <Input
```

Step 90 (continue)

- Isi function render jadi ini saja

```
render() {  
  return (  
    <View>  
      <Header  
        placement="left"  
        centerComponent={{ text: 'Please Login', style: { color: '#fff' } }}  
      />  
      {this.renderContent()}  
    </View>  
  );  
}
```

Finish!!

- Finally we finish our last mobile App!!

