<https://reactnative.dev/docs/tutorial>

<https://reactnative.dev/docs/intro-react>

cd "C:\Users\Deni\AwesomeProject"

**create –**

npx react-native init PROJECT\_NAME

**--version**

npx react-native init AwesomeProject --version X.XX.X

**--template**

npx react-native init AwesomeTSProject --template react-native-template-typescript

**Run -**

npx react-native run-android

**Connected Devices -**

adb devices

**adb reverse tcp:8081 tcp:8081**

**HTML/CSS**

**JAVASCRIPT**

**FLEXBOX CSS**

**NODE JS BASIC**

**REACT JS/JSX**

**Loding**

**Button Click to on**

**Button Click to off**

<https://www.positronx.io/react-native-firebase-login-and-user-registration-tutorial/>

class First extends Component {

    state = {

        name:['Denish','Mayuri','Hina','Binita','Dhumu','Dhutu']

    }

    renderSub() {

        return this.state.name.map((value, index) => {

            return (

                <Text key={index} style={{ textAlign: "center" }}>{value}</Text>

            )

        })

    }

    render() {

        return (

            <View>

                {this.renderSub()}

            </View>

        );

    }

}

export default First;

const styles = StyleSheet.create({

    container: {

        flex: 0,

        justifyContent: 'center',

        alignItems: 'center',

    },

})

import React, { Component } from 'react';

import { StyleSheet, ActivityIndicator, Text, View, Button } from 'react-native';

class FirstChild extends Component {

    state = { isHungry: true };

    render(props) {

        return (

            <View>

                <Text>your Sister Name - {this.props.name}</Text>

                <Button title={this.props.name} onPress={

                    () => {

                        console.log("button Clicked");

                    }

                }

                />

            </View>

        );

    }

}

class First extends Component {

    state = { animating: true, count: 0 }

    onPress = () => {

        this.setState({

            count: this.state.count + 1

        })

    }

    render() {

        //const animating = this.state.animating

        //const count = this.state.count

        return (

            <View style={styles.container}>

                <Text>Heloo Denish</Text>

                <FirstChild name='mayuri'></FirstChild>

                <FirstChild name='Hina'></FirstChild>

                <ActivityIndicator size="large" color="#00ff00" animating={this.state.animating} />

                <View style={styles.container}>

                    <Text>Counter</Text>

                    <Button title='Count' onPress={this.onPress} />

                    <Text style>{this.state.count}</Text>

                </View>

            </View>

        );

    }

}

export default First;

const styles = StyleSheet.create({

    container: {

        flex: 0,

        justifyContent: 'center',

        alignItems: 'center',

    },

})

import React, { Component } from 'react';

import { StyleSheet, Text, View, Button,Switch, Alert, Image, TouchableHighlight, Dimensions, Platform,AppState } from 'react-native';

import Constants from '.././utilities/Constants'

import {Screens} from '.././navigation/Screens'

import {ScreenClass} from '.././navigation/Screens';

import TextInputCustom from '.././components/TextInputCustom';

import {SessionManager} from '../utilities/SessionManager';

import {WebServiceCallManager} from '.././WebServiceCallManager';

import {NavigationManager} from '.././navigation/NavigationManager';

import Fingerprint from '.././fingerprint/FingerprintAndroid';

const dismissKeyboard = require('dismissKeyboard');

const { width, height } = Dimensions.get("window");

let opacity = 1;

// source address of login background image

const loginBackgroundViewImg = require('.././assets/login\_box.png');

const biometricImage = require('.././assets/biometric\_icon.png');

var langs = new Constants(); // language localization

/\* Login View for displaying TextInputs(username, passwords),

Buttons (Login, Register) and Labels (Terms & condition , forget password)

\*/

export class LoginContainer extends Component {

constructor(props){

super(props);

this.state = {

username: '' , // user33

password: '', // Awaqas@2

emptyInputFields : '',

falseSwitchIsOn : true,

phase: 'normal',

message: '',

cancelled: false

}

}

componentDidMount() {

// Fingerprint.saveCredentials("user33","Awaqas@2",()=>{

// this.setState({

// phase: 'saved',

// message: ''

// });

// },()=>{});

this.authenticate();

AppState.addEventListener("change", async(state) => {

try {

if(state === "active" && await Fingerprint.isAuthenticationCanceled()) {

this.authenticate()

}

}

catch(z) {

console.error(z)

}

})

}

async componentWillUnmount() {

try {

if(!Fingerprint.isAuthenticationCanceled()) {

//stop listening to authentication.

await Fingerprint.cancelAuthentication();

}

} catch(z) {

console.error(z);

}

}

async authenticate() {

try {

// do sanity checks before starting authentication flow.

// HIGHLY recommended in real life usage. see more on why you should do this in the readme.md

const hardware = await Fingerprint.isHardwareDetected();

const permission = await Fingerprint.hasPermission();

const enrolled = await Fingerprint.hasEnrolledFingerprints();

if (!hardware || !permission || !enrolled) {

let message = !enrolled ? 'No fingerprints registered.' : !hardware ? 'This device doesn\'t support fingerprint scanning.' : 'App has no permission.'

this.setState({

phase: 'fail',

message

});

return;

}

await Fingerprint.authenticate(warning => {

this.setState({

phase: 'warn',

message: warning.message

})

});

// if we got this far, it means the authentication succeeded.

this.setState({

phase: 'success',

message: ''

});

// in real life, we'd probably do something here (process the payment, unlock the vault, whatever)

// but this is a demo. so restart authentication.

// setTimeout(() => this.authenticate(), 3000);

} catch (error) {

if(error.code == Fingerprint.FINGERPRINT\_ERROR\_CANCELED) {

// we don't show this error to the user.

// we will check if the auth was cancelled & restart the flow when the appstate becomes active again.

return;

}

this.setState({

phase: 'fail',

message: error.message

})

}

}

buttonsHandler(type){

switch (type) {

case 'forgotpassword':

this.props.navigation.push(Screens.ForgotPasswordScreen);

break;

case 'forgotuserid':

this.props.navigation.push(Screens.ForgotUserIDScreen);

break;

case 'unlockprofile':

this.props.navigation.push(Screens.UnlockProfileScreen);

break;

case 'register':

dismissKeyboard();

this.props.navigation.push(Screens.RegistrationWelcomeScreen);

break;

case 'login':

this.loginWebServiceCall();

break;

default:

alert(type + ' is pressed');

}

}

// this will be called when user hit login button

loginWebServiceCall()

{

if(this.state.username.length === 0 && this.state.password.length === 0){

this.setState({emptyInputFields:langs.strings.login\_userid\_password\_empty});

this.userName.textFocus();

}

else if (this.state.username.length === 0 ) {

this.setState({emptyInputFields:langs.strings.login\_userid\_empty});

this.userName.textFocus();

}

else if ( this.state.password.length === 0) {

this.setState({emptyInputFields:langs.strings.login\_password\_empty});

this.password.textFocus();

}else{

this.setState({emptyInputFields:''});

var params = {

"Password": this.state.password,

"UserName": this.state.username,

"LoginType": 'Manual',

};

this.webservicemanager.callWebService("LOGIN","TRANSACTION",params,(response) => {this.handleWebServiceCallResponse(response);});

}

}

/\* handle the web service successfull response error

response will be handled inside WebServiceCallManager \*/

handleWebServiceCallResponse(data){

dismissKeyboard();

var userData = {

"username":this.state.username,

"password":this.state.password

}

var passwordPolicy = {

"passwordPolicy" : data.Body.Transaction.PasswordPolicy,

"passwordPolicyRegex" : data.Body.Transaction.PasswordPolicyRegex

}

SessionManager.setSessionValue(Constants.FASTTRANSFER\_BENEFICIARY\_BRANCH\_LIST,data.Body.Transaction.BranchList);

SessionManager.setSessionValue(Constants.BENEFICIARY\_COUNTRY,data.Body.Transaction.CountryList);

SessionManager.setSessionValue(Constants.BENEFICIARY\_RELATIONSHIP, data.Body.Transaction.RelationList);

SessionManager.setSessionValue(Constants.LOGIN\_USERDATA, userData);

SessionManager.setSessionValue(Constants.CUSTOMER\_NUMBER,data.Body.Transaction.CUSTNO);

SessionManager.setSessionValue(Constants.PASSWORD\_POLICY, passwordPolicy);

SessionManager.setSessionValue(Constants.SECURITY\_QUESTIONS\_LIST,data.Body.Transaction.Questions);

var nextScreenName = data.Body.Transaction.NextScreenName;

const SpecificScreenClass = ScreenClass.getClassFromClassName(nextScreenName);

SessionManager.setSessionValue('nextScreenName', nextScreenName);

this.props.navigation.push(SpecificScreenClass);

this.setState({

username:'',

password:'',

emptyInputFields:''

});

this.userName.textClear();

this.password.textClear();

dismissKeyboard();

}

// handling text input field focus

textHandler(){

this.password.focus();

}

onSwitchToggle(value){

if(value){

opacity = 1;

}

else{

opacity= 0.4;

}

this.setState({falseSwitchIsOn: value});

}

render(){

this.fetchCredentials(this.webservicemanager,this.handleWebServiceCallResponse.bind(this));

return(

<View style={ styles.loginView}>

<Image style={ styles.loginViewBackground} source={loginBackgroundViewImg}>

<View>

<TextInputCustom

ref ={(ref) => this.userName = ref}

placeholder={langs.strings.login\_userid\_placeholder}

secureTextEntry={false}

onChangeTextCallback={val => this.setState({'username' : val})}

returnKeyType="next"

textInputWidth = {((width\*86)/100)}

// onEndEditingCallback = {() => this.password.textFocus()}

/>

<TextInputCustom

ref ={(ref) => this.password =ref}

placeholder={langs.strings.login\_password\_placeholder}

secureTextEntry={true}

onChangeTextCallback={val => this.setState({'password' : val})}

returnKeyType="done"

textInputWidth = {((width\*86)/100)}

/>

<Text style={ styles.emptyInputFields}>{this.state.emptyInputFields}</Text>

</View>

<View style={ styles.middleContainerViewButtons}>

<View style={ styles.middleContainerViewButtonsBtn}>

<TouchableHighlight onPress={ () => this.buttonsHandler('login')}>

<Text style={ styles.btnTextLabels}>{langs.strings.login\_btnLogin}</Text>

</TouchableHighlight>

</View>

<View style={ styles.middleContainerViewButtonsBtn}>

<TouchableHighlight onPress={() => this.buttonsHandler('register')}>

<Text style={ styles.btnTextLabels}>{langs.strings.login\_btnRegister}</Text>

</TouchableHighlight>

</View>

</View>

<TouchableHighlight onPress={() => {this.buttonsHandler('forgotpassword')}} underlayColor = {'transparent'}>

<Text style={ styles.labels} >

Forogot Password

</Text>

</TouchableHighlight>

<TouchableHighlight onPress={() => {this.buttonsHandler('forgotuserid')}} underlayColor = {'transparent'}>

<Text style={ styles.labels} >

Forogot User ID

</Text>

</TouchableHighlight>

<TouchableHighlight onPress={() => this.buttonsHandler('terms')} underlayColor = {'transparent'}>

<View >

<Text style={ styles.labels}>

{langs.strings.login\_txtTermsAndConditions}

</Text>

</View>

</TouchableHighlight>

<View style={styles.fingerPrintLayout}>

<TouchableHighlight underlayColor = {'transparent'}>

<View >

<Image style={styles.biometricImage} source={biometricImage}/>

</View>

</TouchableHighlight>

<View style={styles.switchRow} >

<Text style={ styles.labels} >

Enable Finger Print Login

</Text>

<Switch

onValueChange={(value) => this.onSwitchToggle(value) }

style={styles.switchControl}

value={this.state.falseSwitchIsOn} />

</View>

<Text>{this.state.message}</Text>

</View>

</Image>

<WebServiceCallManager visible={false} nav = {this.props.navigation} ref={ (input) => {this.webservicemanager = input;}}/>

</View>

);

}

fetchCredentials(web,resHandler) {

if(this.state.phase === 'success') {

Fingerprint.fetchCredentials(

(...res) => {

console.log(res);

var params = {

"Password": res[1],

"UserName": res[0],

"LoginType": 'Biometric'

};

this.setState({username:params.UserName,password:params.Password,phase:''})

this.webservicemanager.callWebService("LOGIN","TRANSACTION",params,(response) => {this.handleWebServiceCallResponse(response);},

(err)=> {

this.authenticate();});

},

(res) => {

console.log(res);

return null;

}

);

}

}

} // end of class