import Expo from 'expo';

import React from 'react';

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

import \* as firebase from 'firebase';

import { Kaede } from 'react-native-textinput-effects';

// Firebase SDK Connection

export const config = {

apiKey: "AIzaSyDu7lwUQeF0rXLTLEP5dzq6tIH8V2kOJgE",

authDomain: "awsomepiggy-aacf0.firebaseapp.com",

databaseURL: "https://awsomepiggy-aacf0.firebaseio.com",

projectId: "awsomepiggy-aacf0",

storageBucket: "awsomepiggy-aacf0.appspot.com",

messagingSenderId: "875965742442"

};

var fire = firebase.initializeApp(config);

// Define Clickable Button for Main Render Function

export default class Button extends React.Component{

render () {

return (

<TouchableHighlight>

<View style={styles.button}>

<Text style={styles.buttonText}>Tap Me</Text>

</View>

</TouchableHighlight>

)

}

}

// Initial App Class Component

class App extends React.Component {

constructor () {

super();

this.state = {

balance: "",

value: 1000

};

}

// This will be triggered once Render is Finished and will Update Data in Firebase.

componentDidMount () {

const rootRef = fire.database().ref().child('react');

const balanceRef = rootRef.child('balance');

// console.log(speedRef);

balanceRef.on('value', snap => {

this.setState({

balance: snap.val()

});

});

// Push data to the Database with Push Key

const valueRef = fire.database().ref().child('value').push().key;

var updates = {};

updates['value'] = this.state.value;

return fire.database().ref().update(updates);

}

// The Initial Main Render Function

render() {

return (

// <View style={styles.container}>

// <Text style = {styles.Header}>Place your Amount:</Text>

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

// <Button/>

<View style={[styles.card1, { backgroundColor: '#F9F7F6' }]}>

<Text style={styles.title}></Text>

<Kaede

label={'Euro'}

defaultValue={'Place Amount: '}

editable={false}

/>

<Kaede

style={styles.input}

label={'Amount:'}

labelStyle={{

color: 'white',

backgroundColor: '#fcb794',

}}

inputStyle={{

color: 'white',

backgroundColor: '#db8d67',

}}

keyboardType="numeric"

/>

</View>

// </View>

);

}

}

const styles = StyleSheet.create({

container: {

flex: 1,

backgroundColor: '#fff',

alignItems: 'center',

justifyContent: 'center',

},

button: {

backgroundColor: "#33AAFF",

borderWidth: 10,

borderRadius: 20,

borderColor: "#33AAFF",

padding: 5

},

buttonText: {

fontSize: 24,

fontWeight: "bold",

color: "#FFFFFF"

},

Header: {

fontSize: 30,

fontWeight: "bold",

color: "#33AAFF",

alignItems: 'center',

justifyContent: 'center',

position: 'absolute',

top: 18,

borderColor: '#33AAFF',

borderRadius: 30,

}

});

Expo.registerRootComponent(App);

Dashboard:

import Expo from 'expo';

import React from 'react';

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

import \* as firebase from 'firebase';

import { Kaede } from 'react-native-textinput-effects';

// Firebase SDK Connection

const config = {

apiKey: "AIzaSyDu7lwUQeF0rXLTLEP5dzq6tIH8V2kOJgE",

authDomain: "awsomepiggy-aacf0.firebaseapp.com",

databaseURL: "https://awsomepiggy-aacf0.firebaseio.com",

projectId: "awsomepiggy-aacf0",

storageBucket: "awsomepiggy-aacf0.appspot.com",

messagingSenderId: "875965742442"

};

var fire = firebase.initializeApp(config);

// Define Clickable Button for Main Render Function

// Initial App Class Component

class Dashboard extends React.Component {

constructor () {

super();

this.state = {

balance: "",

value: 1000

};

}

// This will be triggered once Render is Finished and will Update Data in Firebase.

componentDidMount () {

const rootRef = fire.database().ref().child('react');

const balanceRef = rootRef.child('balance');

// console.log(speedRef);

balanceRef.on('value', snap => {

this.setState({

balance: snap.val()

});

});

// Push data to the Database with Push Key

const valueRef = fire.database().ref().child('value').push().key;

var updates = {};

updates['value'] = this.state.value;

return fire.database().ref().update(updates);

}

// The Initial Main Render Function

render() {

return (

<View style = {styles.container} >

<Text style = {styles.text}>Balance: {this.state.balance} Deposit: {this.state.value}</Text>

</View>

)

};

}

const styles = StyleSheet.create({

container: {

flex: 1,

justifyContent: 'center',

alignItems: 'center',

backgroundColor: '#43A047',

flexDirection: 'column',

},

text: {

backgroundColor: '#43A047',

color: '#37474F',

}

});

Expo.registerRootComponent(Dashboard);

Json:

{

<<<<<<< HEAD

"name": "piggybank",

"version": "0.0.0",

"description": "Hello Expo!",

"author": null,

"private": true,

"main": "main.js",

"dependencies": {

"expo": "17.0.0",

"firebase": "^4.1.2",

"react": "16.0.0-alpha.6",

"react-native": "https://github.com/expo/react-native/archive/sdk-17.0.0.tar.gz",

"react-native-textinput-effects": "^0.3.1"

}

=======

"name": "@expo/vector-icons",

"version": "5.0.0",

"description": "Built-in support for 10 popular icon fonts and the tooling to create your own Icon components from your font and glyph map. This is a wrapper around react-native-vector-icons to make it compatible with Expo.",

"main": "index.js",

"scripts": {},

"repository": {

"type": "git",

"url": "https://github.com/expo/vector-icons.git"

},

"keywords": [

"expo"

],

"author": "Brent Vatne",

"license": "MIT",

"bugs": {

"url": "https://github.com/expo/vector-icons/issues"

},

"homepage": "https://expo.github.io/vector-icons",

"dependencies": {

"react-native-vector-icons": "4.1.1"

},

"devDependencies": {}

>>>>>>> f05e92dde539398e93bb4aeaf07abd5258f9b769

}

exp json

{

"name": "piggybank",

"description": "An empty new project",

"slug": "piggybank",

"privacy": "public",

"sdkVersion": "17.0.0",

"version": "1.0.0",

"orientation": "portrait",

"primaryColor": "#cccccc",

"icon": "./assets/icons/app.png",

"loading": {

"icon": "./assets/icons/loading.png",

"hideExponentText": false

},

"packagerOpts": {

"assetExts": ["ttf", "mp4"]

},

"ios": {

"supportsTablet": true

}

}