

React Native Databases— Wireless Buzzer App

React
Native
Databases



What we did:


- Connected the React Native Application to the Realtime database.
- Created timestamp for the button presses.


How we did it:

Connecting our React Native application to firebase:

We created a new Realtime database called "Wireless Buzzer".




 **Firebase**



Recent projects

+

Add project

 Explore a demo project

Light B


light-bag-2

</>

Let's start with a name for your project

Project name

wireless-buzzer







 wireless-buzzer

Continue

Google Analytics for your Firebase project

Google Analytics is a free and unlimited analytics solution that enables targeting, reporting, and more in Firebase Crashlytics, Cloud Messaging, In-App Messaging, Remote Config, A/B Testing, Predictions, and Cloud Functions.

Google Analytics enables:

-  A/B testing [?](#)
-  User segmentation & targeting across Firebase products [?](#)
-  Predicting user behavior [?](#)
-  Crash-free users [?](#)
-  Event-based Cloud Functions triggers [?](#)
-  Free unlimited reporting [?](#)


☒ Enable Google Analytics for this project
Recommended

[Previous](#)

Continue

Configure Google Analytics

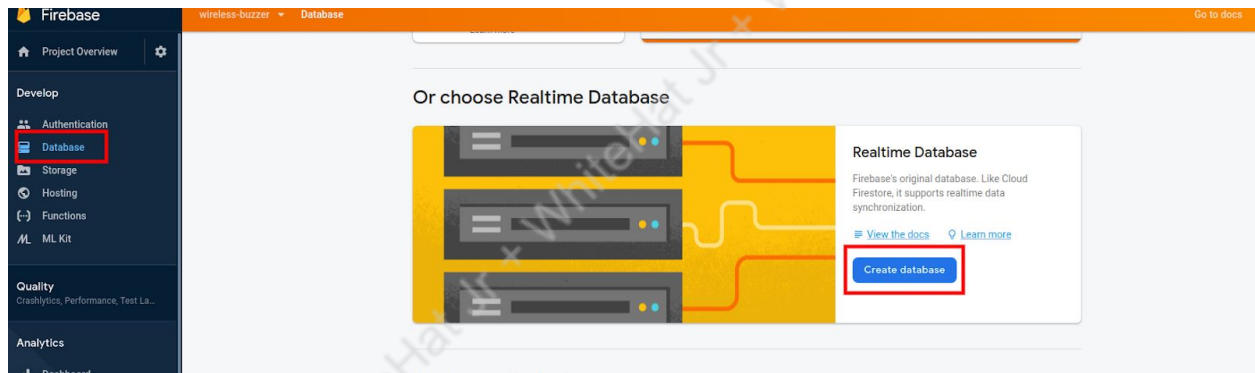
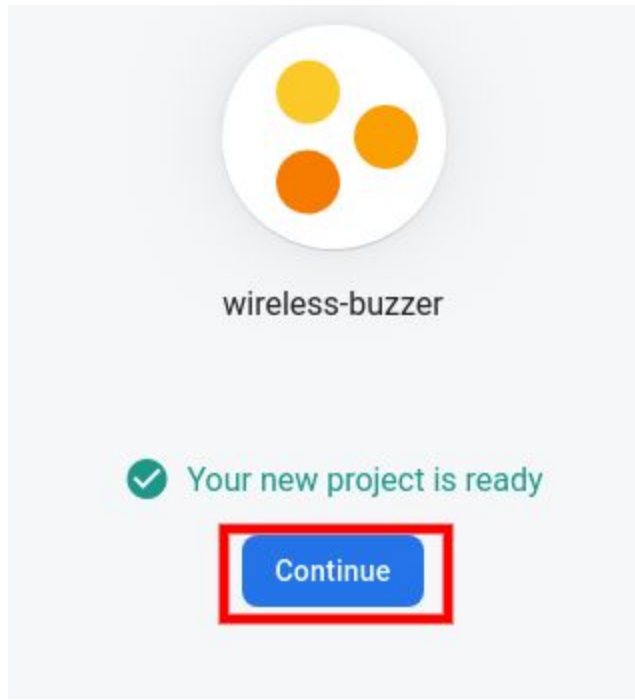
Choose or create a Google Analytics account [?](#)

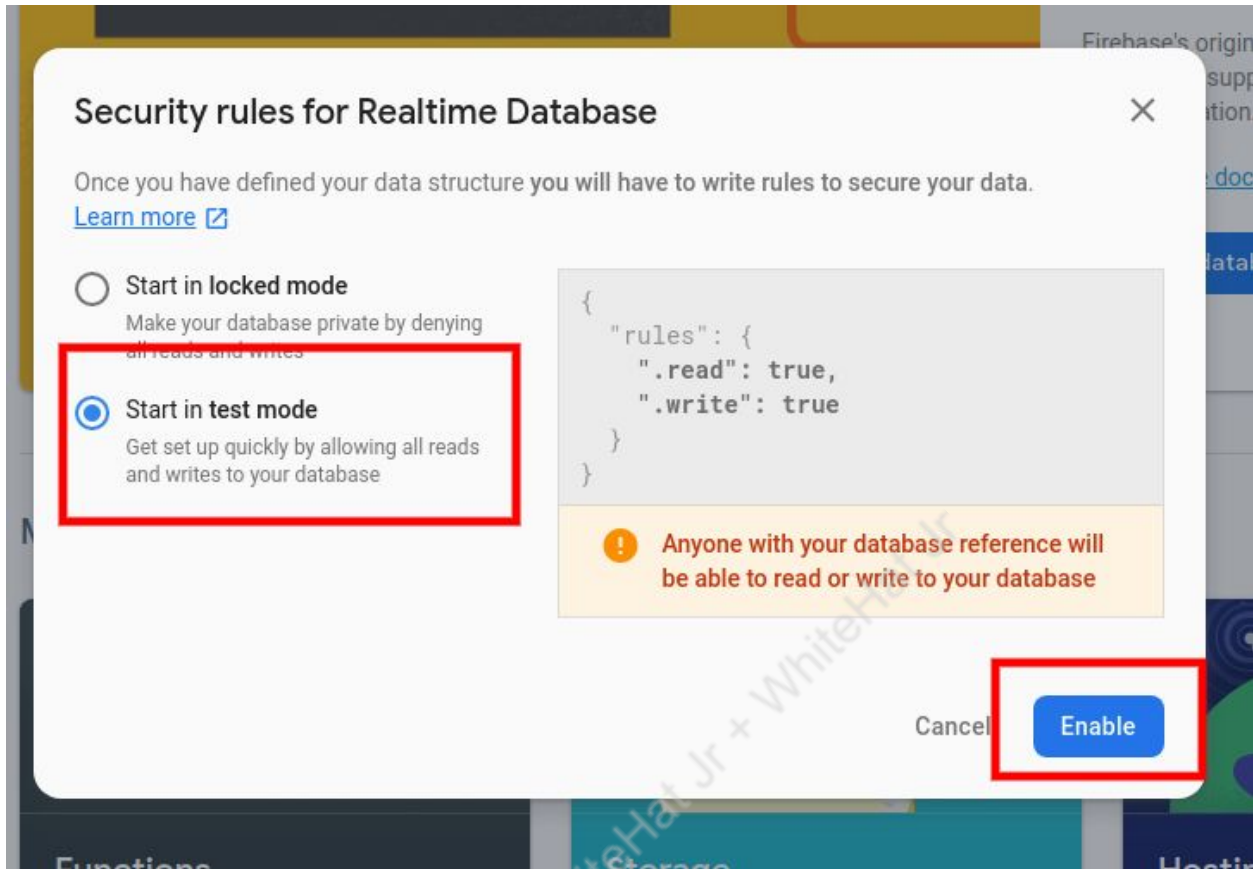
 Default Account for Firebase

Upon project creation, a new Google Analytics property will be created in your chosen Google Analytics account and linked to your Firebase project. This link will enable data flow between the products. Data exported from your Google Analytics property into Firebase is subject to the Firebase terms of service, while Firebase data imported into Google Analytics is subject to the Google Analytics terms of service. [Learn more.](#)

[Previous](#)

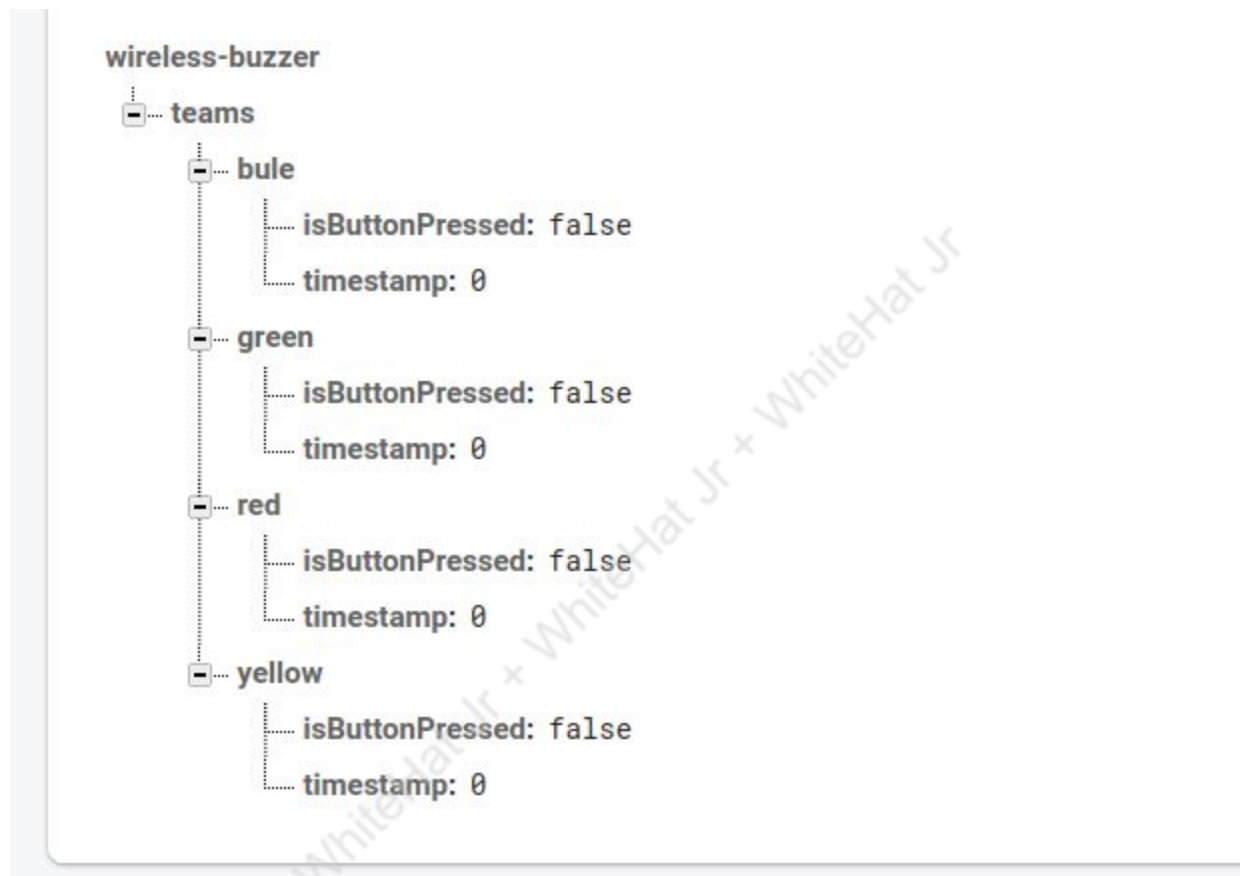
Create project



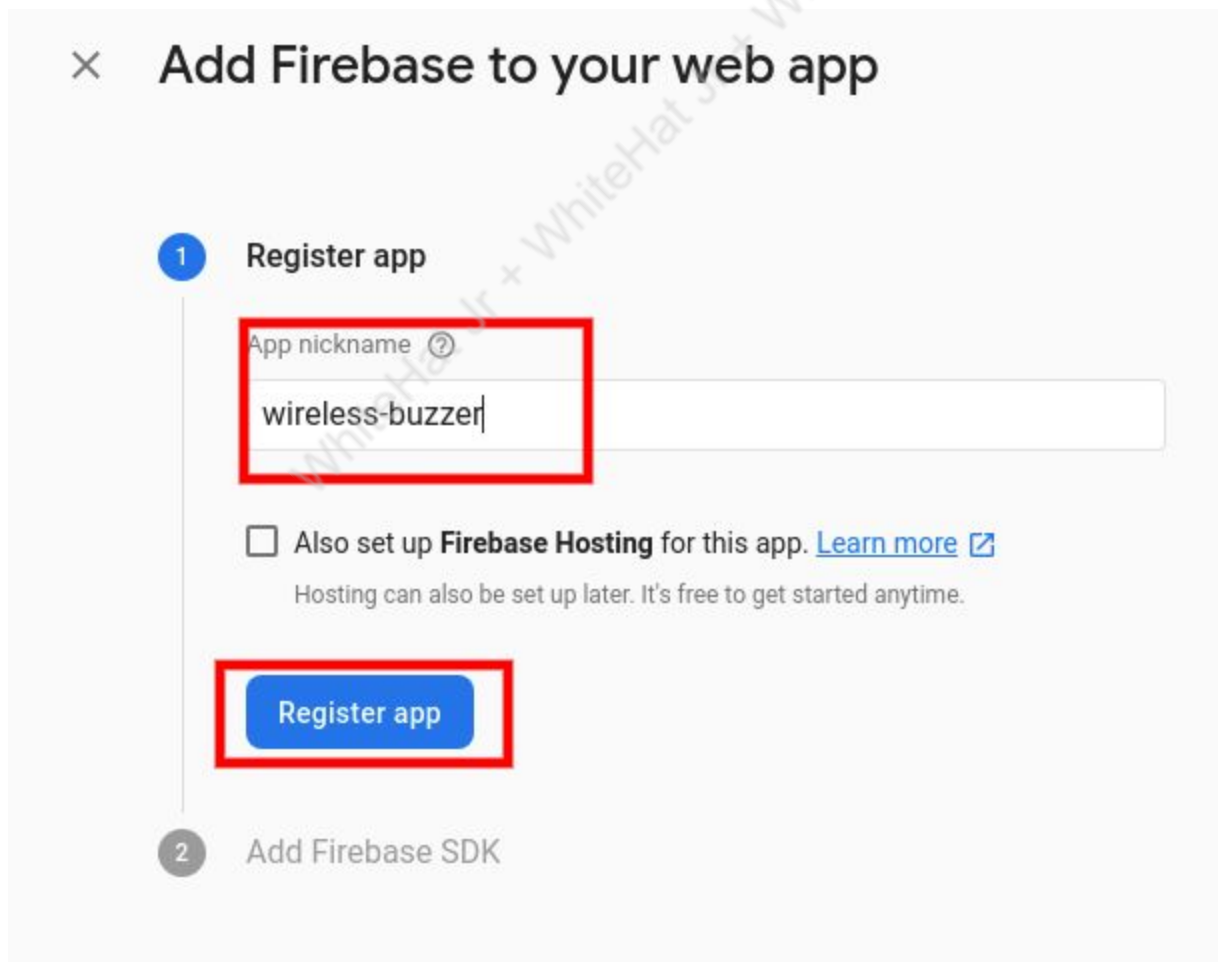
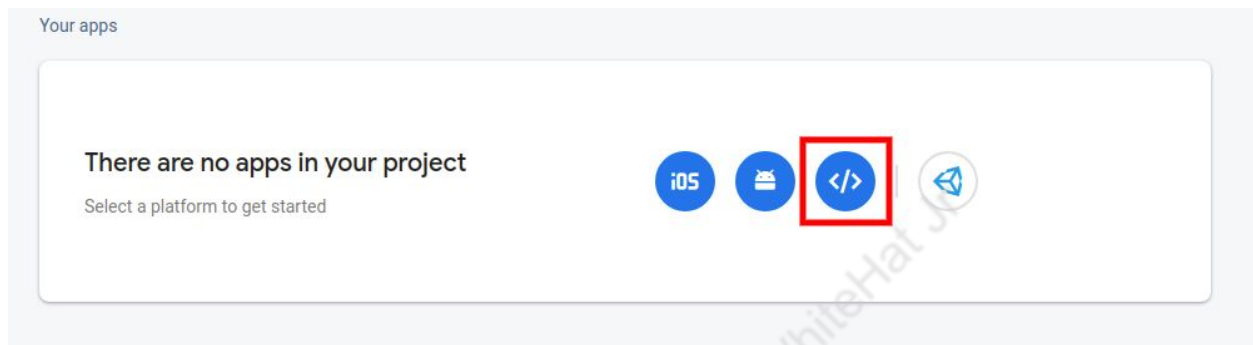
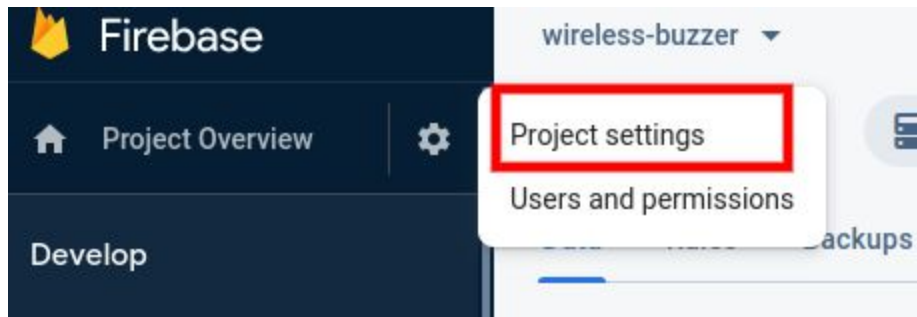


Creating Data fields:

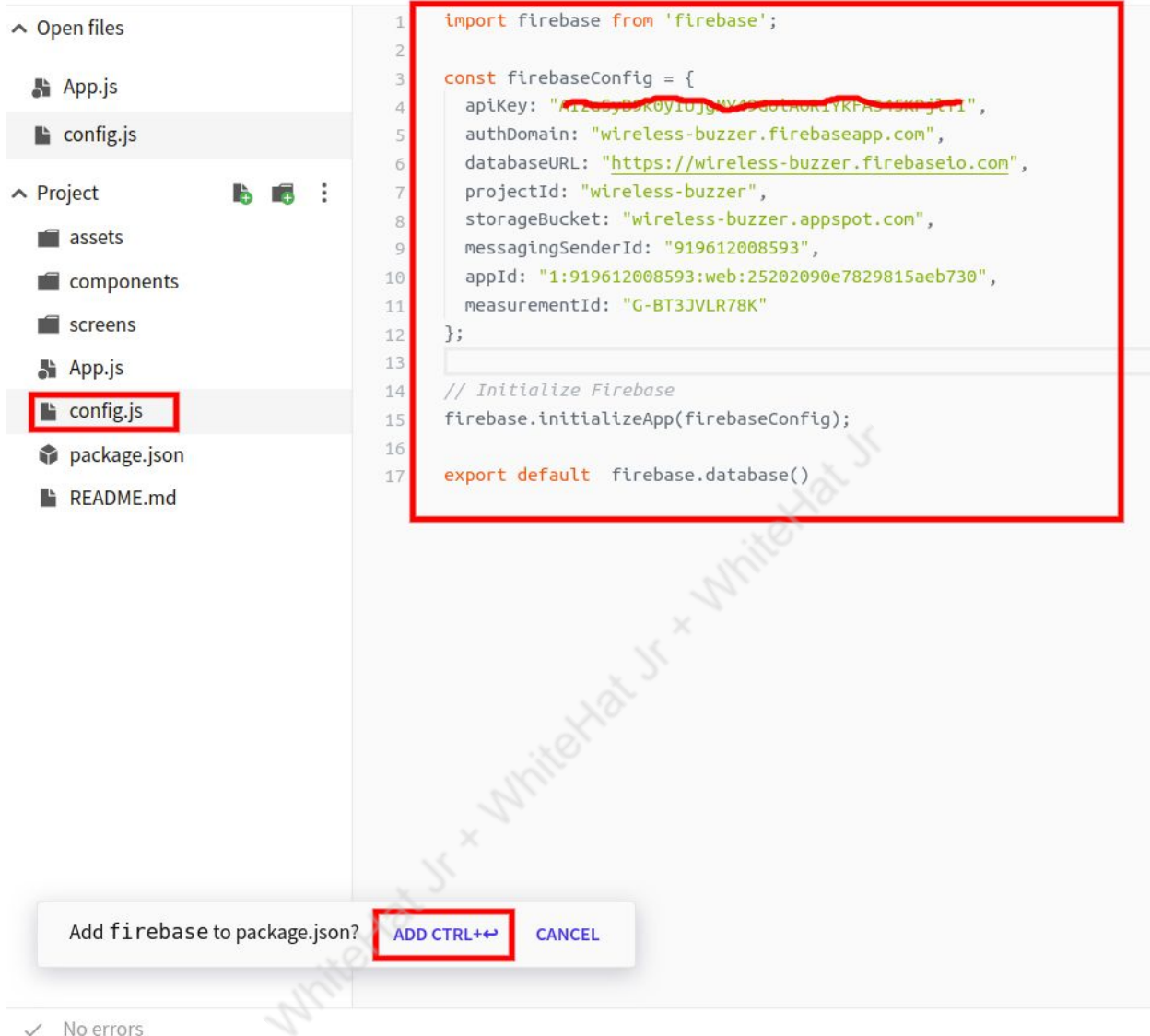
We have a data field called teams. Inside teams, we will have the teams - red, green, blue and yellow. For each team, we are going to have two fields - 'isButtonPressed' and 'timestamp'. Initially 'isButtonPressed' is going to have the value of "false". Whenever the team button is pressed, this value will turn to "true". 'timestamp' will capture the time at which the button is pressed. It will contain a default value of 0.



We registered our react native app to use the database.



create a new file called "config.js" in our application folder.




```
1  import firebase from 'firebase';
2
3  const firebaseConfig = {
4    apiKey: "AIzaSyBkRdy10jgWV120tA0K1YKFA5t5Np3t1",
5    authDomain: "wireless-buzzer.firebaseio.com",
6    databaseURL: "https://wireless-buzzer.firebaseio.com",
7    projectId: "wireless-buzzer",
8    storageBucket: "wireless-buzzer.appspot.com",
9    messagingSenderId: "919612008593",
10   appId: "1:919612008593:web:25202090e7829815aeb730",
11   measurementId: "G-BT3JVL78K"
12 };
13
14 // Initialize Firebase
15 firebase.initializeApp(firebaseConfig);
16
17 export default firebase.database()
```

Add firebase to package.json? **ADD CTRL+↵** CANCEL

✓ No errors

import the `firebase.database()` as `db` from `config.js` file inside 'SoundButton.js'.

 **Student Activity 1: Switch Navigator Reference** ⓘ
 All changes saved less than 20 seconds ago. [See previous saves.](#) ✓

Open files

- App.js
- SoundButton.js
- BuzzerScreen.js
- config.js

Project

- assets
- components**
- AppHeader.js
- AssetExample.js
- SoundButton.js**
- screens
- BuzzerScreen.js
- HomeScreen.js
- App.js
- config.js
- package.json
- README.md

```

1  import * as React from 'react';
2  import { Text, View, TouchableOpacity, StyleSheet } from 'react-native';
3  import { Audio } from 'expo-av';
4  import db from '../config';
5
6
7  class SoundButton extends React.Component {
8    playSound = async () => {
9      await Audio.Sound.createAsync(
10        { uri: 'http://soundbible.com/mp3/Buzzer-SoundBible.com-188422102.mp3' },
11        { shouldPlay: true }
12      );
13    }
14
15    render() {
16      return (
17        <TouchableOpacity
18          style={[styles.button, { backgroundColor: this.props.color }]}
19          onPress={this.playSound}>
20          <Text
21            style={styles.buttonText}>
22            Press Me
23          </Text>
24        </TouchableOpacity>
25      );
26    }
27  }
28
29  const styles = StyleSheet.create({
30    button: {
31      marginTop: 100,
32      marginLeft: 80,
33      borderWidth: 1,
34      borderColor: 'rgba(0,0,0,0.2)',
35      alignItems: 'center',
36      justifyContent: 'center',

```

We wrote a function called 'isButtonPressed()' which takes `teamColor` as an input(argument).

This function should connect to the database and update the 'isButtonPressed' field in our database from "false" to "true".



Student Activity 1: Switch Navigator Reference ⓘ

All changes saved half a minute ago. [See previous saves.](#) ✓
 Search  Run

Open files
 App.js
 SoundButton.js
 Project
 assets
 components
 AppHeader.js
 AssetExample.js
 SoundButton.js
 screens
 BuzzerScreen.js
 HomeScreen.js
 App.js
 config.js
 package.json
 README.md

```

1  import * as React from 'react';
2  import { Text, View, TouchableOpacity, StyleSheet } from 'react-native';
3  import { Audio } from 'expo-av';
4
5  import db from '../config';
6
7  class SoundButton extends React.Component {
8    playSound = async () => {
9      await Audio.Sound.createAsync(
10        { uri: 'http://soundbible.com/mp3/Buzzer-SoundBible.com-188422102.mp3' },
11        { shouldPlay: true }
12      );
13    }
14
15    isButtonPressed(buttonColor){
16      var team = db.ref('teams/' + buttonColor + '/')
17      team.update({
18        "isButtonPressed" : true,
19        "timestamp" : 0
20      })
21    }
22
23    render() {
24      return (
25        <TouchableOpacity
  
```

We created a third function which first calls 'isButtonPressed()' and then calls 'playSound()'.



Student Activity 1: Switch Navigator Reference ⓘ

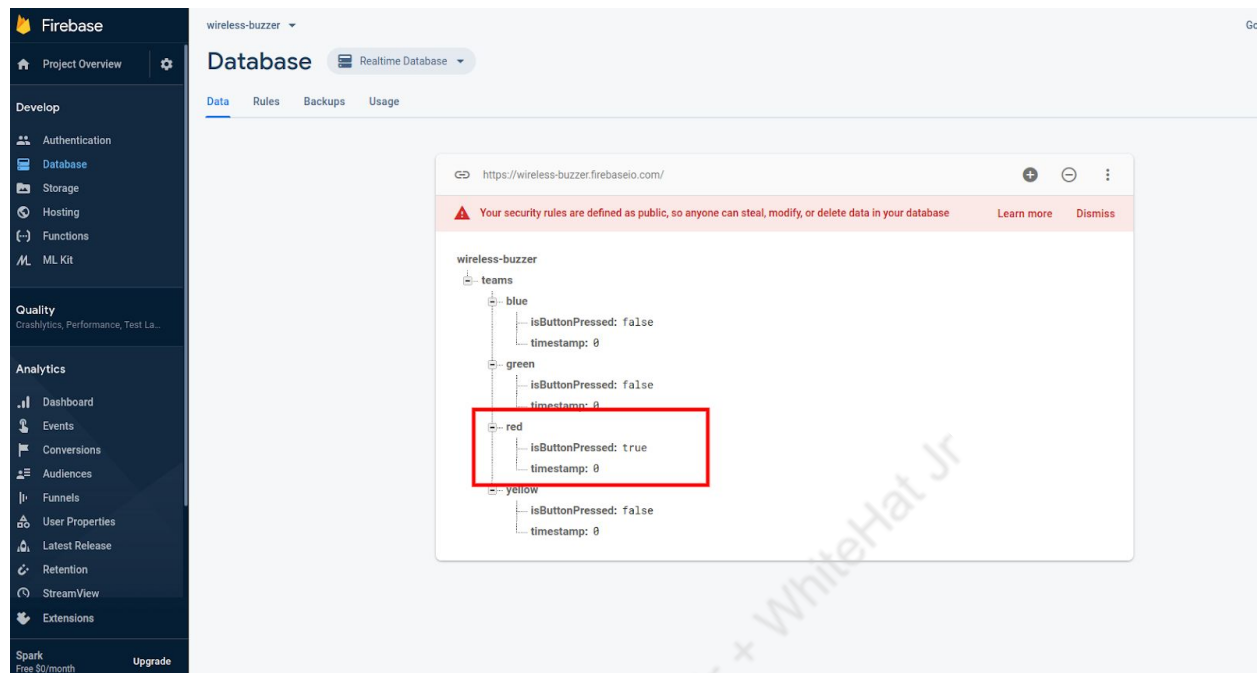
All changes saved 2 minutes ago. [See previous saves.](#) ✓
 Search  Run

Open files
 App.js
 SoundButton.js
 Project
 assets
 components
 AppHeader.js
 AssetExample.js
 SoundButton.js
 screens
 BuzzerScreen.js
 HomeScreen.js
 App.js
 config.js
 package.json
 README.md

```

10  { uri: 'http://soundbible.com/mp3/Buzzer-SoundBible.com-188422102.mp3' },
11  { shouldPlay: true }
12  );
13  }
14
15  isButtonPressed(buttonColor){
16    var team = db.ref('teams/' + buttonColor + '/')
17    team.update({
18      "isButtonPressed" : true,
19      "timestamp" : 0
20    })
21  }
22
23  render() {
24    return (
25      <TouchableOpacity
26        style={[styles.button, {backgroundColor: this.props.color}]}
27        onPress={() => {
28          var buttonColor = this.props.color
29          this.isButtonPressed(buttonColor)
30          this.playSound()
31        }}
32      <Text
33        style={styles.buttonText}>
34        Press Me
35      </Text>
36    </TouchableOpacity>
37  );
38  }
39  }
40  const styles = StyleSheet.create({
41    button: {
42      marginTop: 100,
43      marginLeft: 80,
44      borderWidth: 1,
  
```

Run



We got a reference to our team in the database and update both `isButtonPressed` and `timeStamp` when the button is pressed.



We called both the functions `isButtonPressed()` and `playSound()` inside `onPress` prop.

Student Activity 1: Switch Navigator Reference ⓘ

All changes saved 2 minutes ago. [See previous saves.](#) ✓

Q Search ▶ R

Open files

- App.js
- SoundButton.js

Project

- assets
- components**
- AppHeader.js
- AssetExample.js
- SoundButton.js**
- screens
- BuzzerScreen.js
- HomeScreen.js
- App.js
- config.js
- package.json
- README.md

```

10     { uri: 'http://soundbible.com/mp3/Buzzer-SoundBible.com-188422102.mp3' },
11     { shouldPlay: true }
12   });
13 }
14
15 isButtonPressed(buttonColor){
16   var team = db.ref('teams/' + buttonColor + "/" );
17   team.update({
18     "isButtonPressed" : true,
19     "timestamp" : 0
20   })
21 }
22
23 render() {
24   return (
25     <TouchableOpacity
26       style={styles.button, {backgroundColor: this.props.color}}
27       onPress={() => {
28         var buttonColor = this.props.color
29         this.isButtonPressed(buttonColor)
30         this.playSound()
31       }}
32     <Text
33       style={styles.buttonText}>
34       Press Me
35     </Text>
36   </TouchableOpacity>
37   );
38 }
39 }
40
41 const styles = StyleSheet.create({
42   button: {
43     marginTop: 100,
44     marginLeft: 80,
45     borderWidth: 1,

```

What's next?:

In the next class, you will be creating the Quiz Master App.