
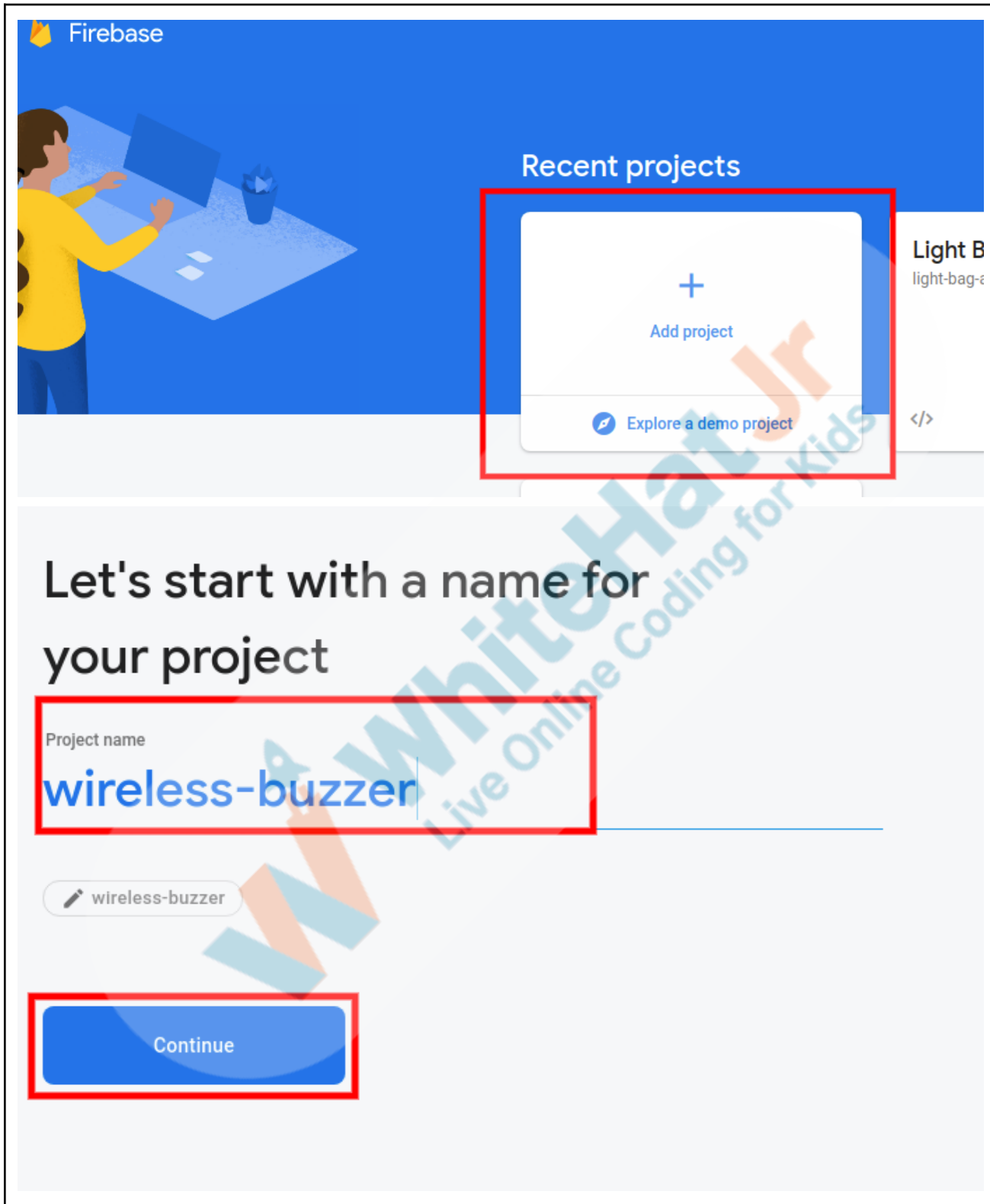


Topic	React Native Databases	
Class Description	Students learn how to connect their react native application to Firebase Realtime Database. They learn about the concept of timestamp. They also write code to rank the order in which the team pressed the buttons.	
Class	C58	
Class time	45 mins	
Goal	<ul style="list-style-type: none"> Connect the React Native Application to the Realtime database. Create timestamp for the button presses. 	
Resources Required	<ul style="list-style-type: none"> Teacher Resources <ul style="list-style-type: none"> Laptop with internet connectivity Earphones with mic Notebook and pen Android/iOS Smartphone with Expo App installed Expo snack account Student Resources <ul style="list-style-type: none"> Laptop with internet connectivity Earphones with mic Notebook and pen Android/iOS Smartphone with Expo App installed Expo snack account 	
Class structure	Warm Up Teacher-led Activity Student-led Activity Wrap up	5 mins 15 min 15 min 5 min
<p style="text-align: center;"><u>CONTEXT</u></p> <ul style="list-style-type: none"> Review code from the previous class. Introduce the problem of finding out who pressed the Buzzer first. 		

Class Steps	Teacher Action	Student Action
Step 1: Warm Up (5 mins)	Welcome back to the class. Remember what we were doing in the last class?	ESR: We were working on creating a Wireless Buzzer. We had created a two screen app where users can choose their teams and see a buzzer of the color of their chosen team.
	Great. And what is still left to do in our app?	We are yet to code for a way to find out which team pressed the button first.
	How do we plan to do that?	We plan to use Firebase Realtime database for that.
	Yes. In today's class we are going to learn how to connect our application to Firebase Realtime Database. After doing that, we will be able to find out who pressed the button first. Ready to get started for today's class?	ESR: Yes!
Teacher Initiates Screen Share		
<u>CHALLENGE</u> <ul style="list-style-type: none"> • Connect the React Native App to the Database. • Detect Buzzer Button presses. 		
Step 2: Teacher-led Activity (15 min)	Teacher Opens <u>Teacher Activity 1</u> Before we start, like in the previous classes, can you go over the code and explain what we have done so far here?	The student goes through the code and explains what different blocks are doing.

	<p>Note: Allow the student to explain the different code blocks and help them wherever their understanding is inconsistent with what is taught in the class.</p>	
	<p>Alright. So now let's get started with connecting our react native application to firebase.</p> <p>First, we need to create a firebase realtime database for our application.</p> <p>Let's login to console.firebase.com and create a firebase database.</p> <p>Teacher logs in to the console.firebase.com. She creates a new Realtime database called "Wireless Buzzer".</p> <p>Note: Create the database in test mode. This will keep the read, write permissions for all users to be true.</p>	<p>The student observes how to create a new Firebase Database.</p>
		



Recent projects

+

Add project

Explore a demo project

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



wireless-buzzer

✓ Your new project is ready

Continue

Or choose Realtime Database

Realtime Database
Firebase's original database. Like Cloud Firestore, it supports realtime data synchronization.
[View the docs](#) [Learn more](#)
Create database

Security rules for Realtime Database

Once you have defined your data structure you will have to write rules to secure your data.
[Learn more](#)

☐ **Start in locked mode**
Make your database private by denying all reads and writes

☒ **Start in test mode**
Get set up quickly by allowing all reads and writes to your database

```
{
  "rules": {
    ".read": true,
    ".write": true
  }
}
```

! Anyone with your database reference will be able to read or write to your database

Cancel **Enable**

Let's create data fields in our database.

What do you think we should write in our database fields?

Remember, the firebase database is similar to a JSON object.

ESR:
varied

	<p>Every field has a key name and a value. Each key can also contain more fields nested inside them.</p>	
	<p>We will have a data field called teams.</p> <p>Inside teams, we will have the teams - red, green, blue and yellow.</p> <p>For each team, we are going to have two fields - 'isButtonPressed' and 'timestamp'.</p> <p>Initially 'isButtonPressed' is going to have the value of "false". Whenever the team button is pressed, this value will turn to "true".</p> <p>'timestamp' will capture the time at which the button is pressed. It will contain a default value of 0.</p> <p>Teacher creates these data fields in the application.</p>	<p>The student observes and learns how to create a firebase database.</p>

wireless-buzzer

teams

bule

isButtonPressed: false

timestamp: 0

green

isButtonPressed: false

timestamp: 0

red

isButtonPressed: false

timestamp: 0

yellow

isButtonPressed: false

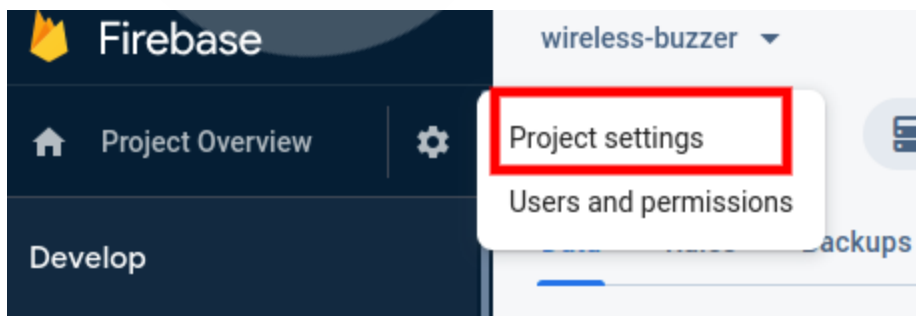
timestamp: 0

We will now need to register our react native app to use the database.

Let's do that first.

Teacher navigates to Project Settings and creates/registers for a new web application to connect to the database.

The student observes how to register the application to use the database.



Your apps

There are no apps in your project

Select a platform to get started

iOS

×

Add Firebase to your web app

1

Register app

App nickname ?

wireless-buzzer

☐ Also set up **Firebase Hosting** for this app. [Learn more](#)

Hosting can also be set up later. It's free to get started anytime.

Register app

2

Add Firebase SDK

	Do you remember how we connected the firebase database to our Car Racing Application last time?	The student tries to recall the database project config keys located under the project settings.
--	---	--

Copy and paste these scripts into the bottom of your <body> tag, but before you use any Firebase services:

```
<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="https://www.gstatic.com/firebasejs/7.6.1/firebase-app.js"></script>

<!-- TODO: Add SDKs for Firebase products that you want to use
https://firebase.google.com/docs/web/setup#available-libraries -->
<script src="https://www.gstatic.com/firebasejs/7.6.1/firebase-analytics.js"></script>

<script>
  // Your web app's Firebase configuration
  var firebaseConfig = {
    apiKey: "AIzaSyB9k0yIuJgWY490aiA8K7rKfAS45Kfj1T",
    authDomain: "wireless-buzzer.firebaseio.com",
    databaseURL: "https://wireless-buzzer.firebaseio.com",
    projectId: "wireless-buzzer",
    storageBucket: "wireless-buzzer.appspot.com",
    messagingSenderId: "919612008593",
    appId: "1:919612008593:web:25202090e7829815aeb730",
    measurementId: "G-BT3JVL78K"
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);
  firebase.analytics();
</script>
```

Learn more about Firebase for web: [Get Started](#), [Web SDK API Reference](#), [Samples](#)

Continue to console

	These config keys contain the address and access permissions to allow us to use the database from our application. We will be using this in our application.	
	Let's create a new file called "config.js" in our application folder. This file will contain the config keys for our database. We will use it to initialize firebase in our application.	The student learns how to initialize firebase using the firebase config keys.

Teacher creates a config.js file where she:

- imports firebase library.
- stores firebase config keys.
- initializes firebase app using the config keys.
- exports firebase.database() to be used in other files.


The screenshot shows a code editor interface. On the left, a file explorer shows a project structure with folders like 'assets', 'components', and 'screens', and files like 'App.js', 'config.js' (highlighted with a red box), 'package.json', and 'README.md'. The main editor area shows the content of 'config.js' with the following code:

```

1 import firebase from 'firebase';
2
3 const firebaseConfig = {
4   apiKey: "AIzaSyB5K0y1UjgWV49G0ta0K1YKPA545K0j3t1",
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();
  
```

A red box highlights the entire code block in the editor. At the bottom, a dialog box asks "Add firebase to package.json?" with two buttons: "ADD CTRL+↵" (highlighted with a red box) and "CANCEL". The status bar at the bottom left shows "✓ No errors".

	When do we want to connect to the database? - in which component?	ESR: In 'SoundButton' when a button is pressed.
	<p>Great. Let's import the firebase.database() as db from config.js file inside 'SoundButton.js'.</p> <p>Note: 'config.js' file by default exports firebase.database(). The name "db" could be anything.</p>	Student observes and asks questions.

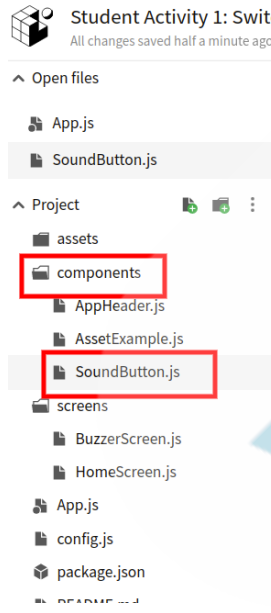


Student Activity 1: Switch Navigator Reference ⓘ
All changes saved less than 20 seconds ago. [See previous saves.](#) ✓

^ 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


```

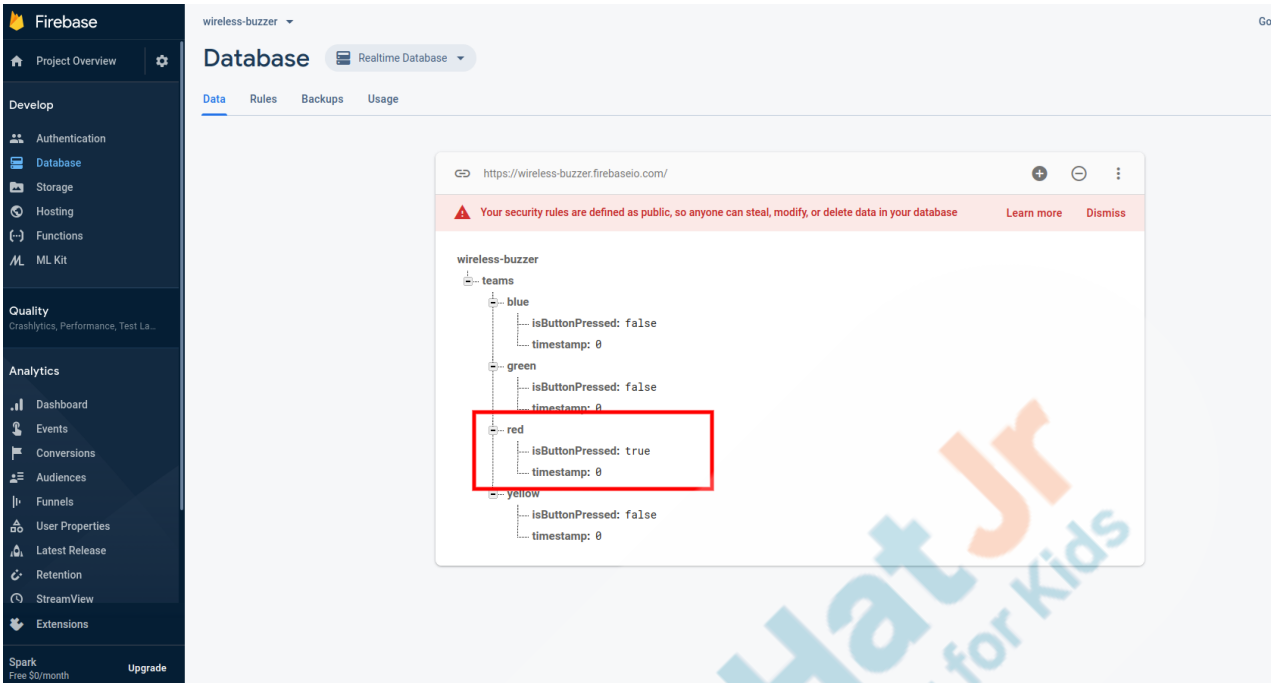
1  import * as React from 'react';
2  import { Text, View, TouchableOpacity, StyleSheet } from 'react-native';
3  import { Audio } from 'expo-av';
4  import db from '../config';
5
6
7  class SoundButton extends React.Component {
8    playSound = async () => {
9      await Audio.Sound.createAsync(
10       { 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',

```


	<p>Inside the SoundButton class, let's write a function called 'isButtonPressed()' which takes teamColor as an input(argument).</p> <p>This function should connect to the database and update the 'isButtonPressed' field in our database from "false" to "true".</p> <p>Can you help me on how to do that?</p> <p>Some guided questions:</p> <ul style="list-style-type: none"> • What do we need to write to the field in the database? • Which database function will help us write to the database? 	<p>The student helps the teacher in writing the questions.</p> <p>ESR: We need a reference to the field in the data. ESR: databaseRef.update() function.</p>
		
	<p>When do we call this function isButtonPressed()?</p>	<p>ESR: When the buzzer Button is pressed. Inside 'onPress' prop for the 'TouchableOpacity'.</p>


	<p>But we are already calling another function called 'playSound()' when the button is pressed.</p> <p>How do we call one more function now?</p> <p>Allow time for the student to think.</p>	<p>The student comes up with varied responses.</p>
	<p>A good way to solve this would be to create a third function which first calls 'isButtonPressed()' and then calls 'playSound()'.</p> <p>We can call this function inside the 'isButtonPressed()' function.</p> <p>Also, we can create this function inside the 'onPress' prop itself inside { }</p> <p>Teacher uses arrow keys to create a function which calls both 'isButtonPressed()' and 'playSound()' functions.</p> <p>This function does not have a name and is called an anonymous function.</p>	<p>The student learns how to write an anonymous function.</p>

<div>  <div> Student Activity 1: Switch Navigator Reference ⓘ </div> <div> All changes saved 2 minutes ago. See previous saves. ✓ </div> <div> Search ▶ Run </div> </div>			
<div> Open files </div> <div> App.js </div> <div> SoundButton.js </div> <div> Project </div> <div> assets </div> <div> components </div> <div> AppHeader.js </div> <div> AssetExample.js </div> <div> SoundButton.js </div> <div> screens </div> <div> BuzzerScreen.js </div> <div> HomeScreen.js </div> <div> App.js </div> <div> config.js </div> <div> package.json </div> <div> README.md </div>	<pre> 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, </pre>	<p>We have done great so far.</p> <p>Let's run the app and see what happens.</p> <p>Teacher shows the change in the 'isButtonPressed' field for the team when the button of that color is pressed.</p>	<p>The student runs and tests the app on their phone.</p> <p>He/She chooses a team color and presses the buzzer.</p>

		
	<p>Awesome. So now I hope you know how to connect to the firebase database from your react native app and update any field.</p> <p>Can you now connect firebase database to your own app and update the timeStamps field for the team when the button is pressed?</p> <p>The timeStamps should contain the time at which the button is pressed.</p> <p>Note: Remind the student about the Date object that they have already learned earlier.</p>	<p>ESR: I can try.</p>
	<p>Great. And obviously I am there to support you.</p>	<p>-</p>
<p>Teacher Stops Screen Share</p>		

	Now it's your turn. Please share your screen with me.	
<ul style="list-style-type: none"> • Ask Student to press ESC key to come back to panel • Guide Student to start Screen Share • Teacher gets into Fullscreen 		
<p style="text-align: center;"><u>ACTIVITY</u></p> <ul style="list-style-type: none"> • Add Timestamp for each button press. 		
Step 3: Student-Led Activity (15 min)	Guide the student to create a firebase database and create a new database - wireless-buzzer	The student visits console.firebase.com and creates a new firebase database.
	Guide the student to create new fields in the database.	The student creates the fields in the database.
	Guide the student to register the app and get the config keys for their app database.	The student generates config keys for the app by registering their app.
	Guide the student to create config.js file in their project. Guide them to initialize the firebase app using config keys and export firebase.database.	The student creates config.js file, initializes the firebase function and exports firebase.database().
	Guide the student to import db from the config file.	The student imports db from config.js file inside the SoundButton Component.
	Guide the student to create the isButtonPressed function.	The student creates a new function 'isButtonPressed'.

	<p>Inside isButtonPressed, let's create a new Date object. This will store the current date/time.</p> <p>We can convert the date into milliseconds using getTime.</p> <p>In computing, time is always measured since Jan 1, 1970. time.getTime() will give us the number of milliseconds passed since that time!!</p>	<p>The student creates a new date object which stores the current time.</p> <p>He/She converts the time into milliseconds.</p>
	<p>Now, let's get a reference to our team in the database and update both isButtonPressed and timeStamp when the button is pressed.</p>	<p>The student gets the database reference for the team and updates both isButtonPressed and timeStamp.</p>
 <p>Student Activity 1: Switch Navigator Reference</p> <p>All changes saved less than 10 seconds ago. See previous saves.</p> <pre> 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 time = new Date().getTime(); 17 var team = db.ref('teams/' + buttonColor + '/') 18 team.update({ 19 "isButtonPressed": true, 20 "timestamp": time 21 }) 22 } 23 24 render() { 25 return (</pre>		
	<p>Finally, we need to call both the functions isButtonPressed() and playSound() inside onPress prop. Let's do that.</p>	<p>The student creates an anonymous function inside onPress prop which calls both the functions.</p>


Student Activity 1: Switch Navigator Reference ⓘ
 All changes saved 2 minutes ago. [See previous saves.](#) ✓

Search 🔍 ▶ Rt

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,
      borderWidth: 1,

```

Ok! Now let's quickly run and test our app

The student tests the app on his/her phone. He/She checks for the change in the database when the sound button is pressed.

Awesome! You have done it!



Teacher Guides Student to Stop Screen Share


FEEDBACK

- Encourage the student to think about the bugs in their code.
- Encourage the student to make reflection notes in the markdown format.
- Complement the student for her/his effort in the class.

Step 4: Wrap-Up (5 min)

So now teams can press the buzzer sound and you can look at the database to check who pressed the buzzer first.

	<p>But you will have to compare the time in milliseconds everytime. You also have to look at your database every time.</p> <p>That is not a good solution....Is it?</p> <p>What are the other problems in our app?</p>	<p>ESR: No</p> <p>Once a button is pressed, we have to manually reset both the timestamp and isButtonPressed fields in the database.</p>
	<p>Yes. That's a problem too. We will solve this problem by creating a Quiz master app in coming classes.</p> <p>Our Quiz Master app can reset the database and read from the database to rank who pressed the button first.</p>	
	<p>Before we end today's class, can we quickly capture what we learned today?</p>	<p>ESR:</p> <ul style="list-style-type: none"> - We learned how to connect the firebase database to our application. - We learned how to export and import database config keys and update the database fields.
	<p>You get a "hats off".</p> <p>Great! See you in the next class then where we will be creating the Quiz Master App.</p>	<p>Make sure you have given at least 2 Hats Off during the class for:</p> <div data-bbox="1019 1570 1312 1675">  </div> <div data-bbox="1019 1722 1312 1816">  </div>

		 Strong Concentration
Project Pointers and Cues (5 min)	<p>*This Project will take only 30 mins to complete. Motivate students to try and finish it immediately after the class.</p> <p>Team Voting App</p> <p>Goal of the Project:</p> <p>In class 58, you have learned about “React with Databases”. You have coded for a wireless buzzer where the first team who clicks on the buzzer will be registered along with the time stamp.</p> <p>In this project, you will apply what you have learned in class to create and configure the database for a Team Voting App and update vote values in the database.</p> <p>Story:</p> <p>Every Saturday, your teacher organizes a fun game activity in which she divides students into two teams in your class. When it comes to voting for the winning teams, it becomes very complex for the teacher to count votes.</p> <p>So she started creating a Team Voting App. She has designed the user interface for the app and needs you to resolve some issues. Are you up for the challenge?</p> <p>I am very excited to see your</p>	

	project solution and I know you both will do really well. Bye Bye!	
<div> <div>Teacher Clicks</div> <div>✕ End Class</div> </div>		
Additional Activities	<p>Encourage the student to write reflection notes in their reflection journal using markdown.</p> <p>Use these as guiding questions:</p> <ul style="list-style-type: none"> • What happened today? <ul style="list-style-type: none"> - Describe what happened - Code I wrote • How did I feel after the class? • What have I learned about programming and developing games? • What aspects of the class helped me? What did I find difficult? 	The student uses the markdown editor to write her/his reflection in a reflection journal.

Activity	Activity Name	Links
Teacher Activity 1	Previous Class Reference	https://snack.expo.io/@whitehatjr/pro-c57
Student Activity 1	Previous Class Reference	https://snack.expo.io/@whitehatjr/pro-c57
Teacher Activity 2	Teacher Reference	https://snack.expo.io/@whitehatjr/pro-c58
Project Solution	Team Voting App	https://snack.expo.io/@snerrus/solution:pro-c58