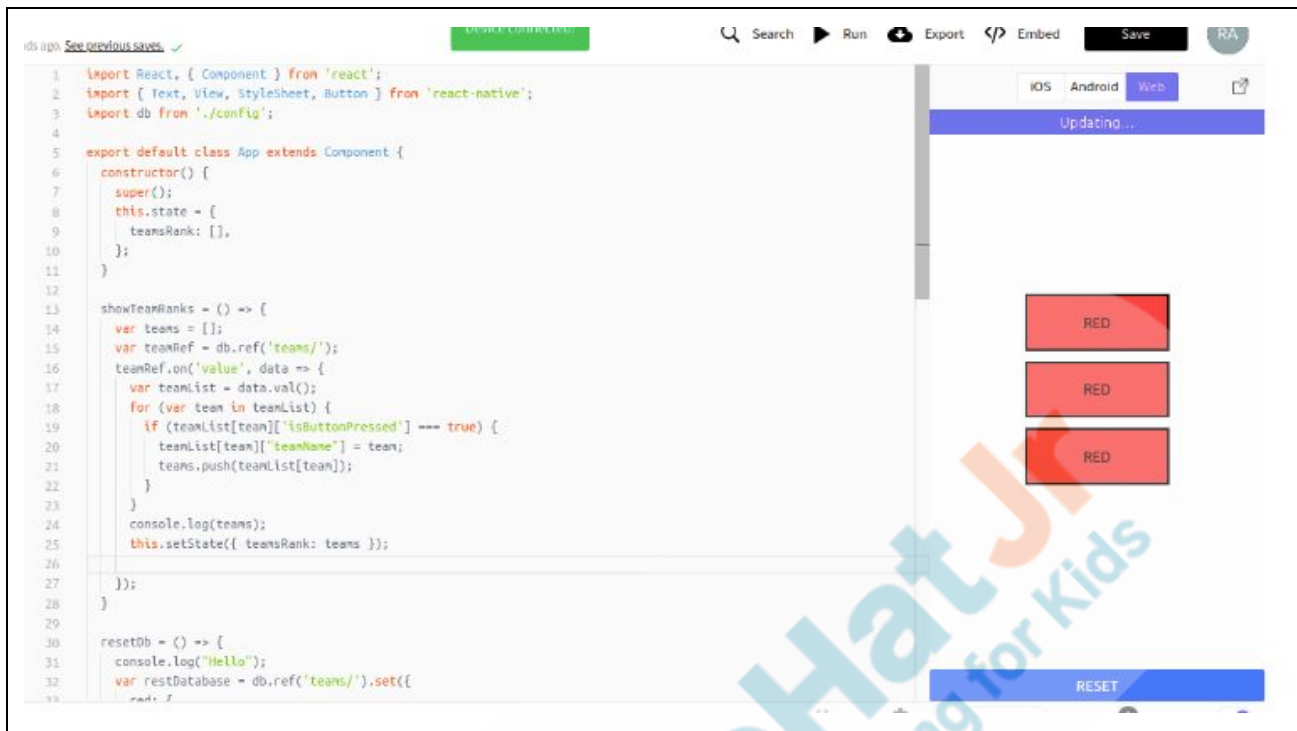


Topic	Fixing Bugs on the Buzzer App	
Class Description	Students will fix the timestamp bug in their Quiz Buzzer app. They will also learn to make the buttons inactive once a team is chosen.	
Class	C61	
Class time	45 mins	
Goal	<ul style="list-style-type: none"> Fix timestamp bug. Make buttons inactive once a team is chosen. 	
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
CONTEXT		
<ul style="list-style-type: none"> Brainstorm on the possible bugs in the app. 		
Class Steps	Teacher Action	Student Action
Step 1: Warm Up (5 mins)	We have almost finished our Quiz Buzzer and Quiz Master Apps. We will soon learn how to generate apk or ipa files to be published on the	

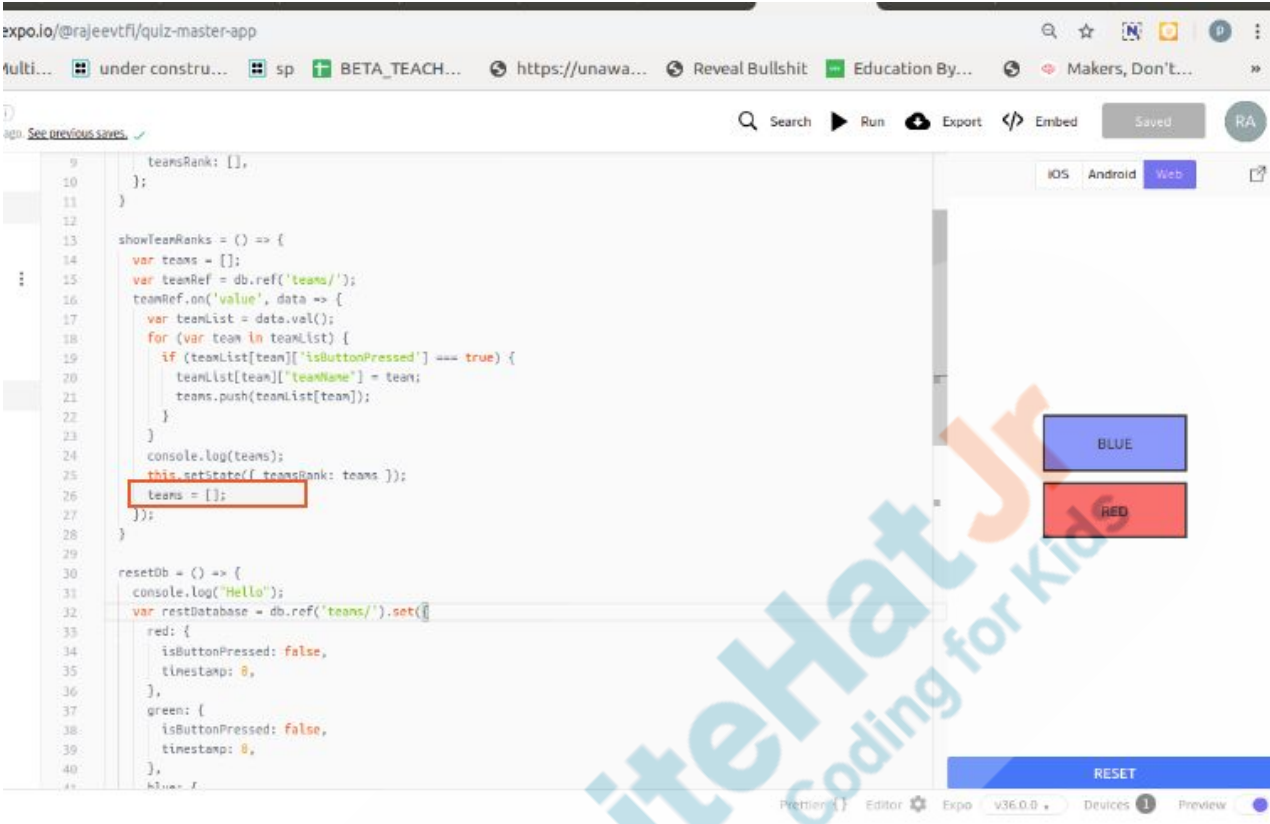
	playstore. Excited ?	ESR: Yes!
	Before, we learn to do that however, there are still a few bugs in our app which we will try to fix. Did you notice any bugs in our app so far?	ESR: varied
	Alright, let's look at a few bugs and fix them.	
Teacher Initiates Screen Share		
<p style="text-align: center;"><u>CHALLENGE</u></p> <ul style="list-style-type: none"> Fetch date/time from the server to fix the timestamp bug. 		
Step 2: Teacher-led Activity (15 min)	<p>There are a few bugs which we can easily see and fix them immediately. Other bugs are more subtle and can only be found out if you use the app for sometime.</p> <p>One bug which we know is - when the user presses the button repeatedly, the team name comes on the quiz master app repeatedly.</p> <p>Teacher opens the Quiz Buzzer and Quiz Master App and shows the bug.</p>	<p>The student looks at the output bug.</p>

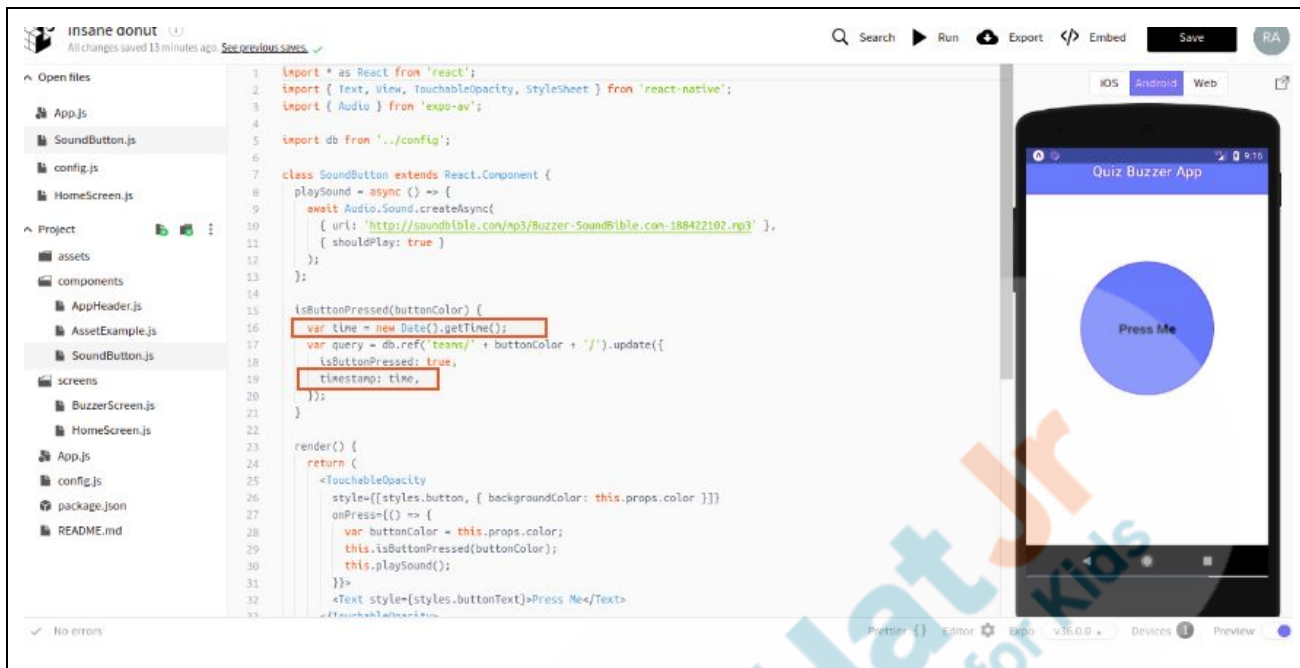


```

1 import React, { Component } from 'react';
2 import { Text, View, StyleSheet, Button } from 'react-native';
3 import db from './config';
4
5 export default class App extends Component {
6   constructor() {
7     super();
8     this.state = {
9       teamsRank: [],
10     };
11   }
12
13   showTeamRanks = () => {
14     var teams = [];
15     var teamRef = db.ref('teams/');
16     teamRef.on('value', data => {
17       var teamList = data.val();
18       for (var team in teamList) {
19         if (teamList[team]['isButtonPressed'] === true) {
20           teamList[team]['teamName'] = team;
21           teams.push(teamList[team]);
22         }
23       }
24       console.log(teams);
25       this.setState({ teamsRank: teams });
26     });
27   }
28
29   resetDb = () => {
30     console.log("Hello");
31     var restDatabase = db.ref('teams/').set({
32       name: ' '
33     });
34   }
35 }
  
```

	<p>Can you think why this happens and what could be a fix for this?</p> <p>Allow the student some time to look at the code and think of a fix for this bug.</p>	<p>ESR:</p> <p>The student suggests ways to fix this bug.</p>
	<p>This bug happens because we keep reading from the database and pushing the teams to the 'teams' array without emptying the array ever. The array then gets written to the state of the component.</p> <p>If we simply empty the array every time, our code will always read from the database again and there will be no duplicacy entered in the array.</p> <p>Teacher shows how to fix the bug inside the Quiz Master App.js</p>	<p>The student thinks about the bug and fixes it.</p>

		
	<p>Let's look at a more subtle bug now.</p> <p>Let's look at the Quiz Buzzer App (SoundButton.js).</p> <p>We have a piece of code where we are trying to get the time at which the buzzer is pressed. We do this using the 'new Date().getTime()' function.</p> <p>Where do you think the code gets the time information from?</p>	<p>ESR: varied</p>



```

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       [ { url: 'http://soundbible.com/mp3/Buzzer-Sound@bible.com-188422102.mp3' } ],
11       { shouldPlay: true }
12     );
13   };
14
15   isButtonPressed(buttonColor) {
16     var time = new Date().getTime();
17     var query = db.ref('teams/' + buttonColor + '/').update({
18       isButtonPressed: true,
19       timestamp: time,
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       >
33         <Text style={styles.buttonText}>Press Me</Text>
34       </TouchableOpacity>
35     );
36   }
37 }

```

The code actually gets the time from the client - that is the smartphone on which it is running the code on.

When you write 'new Date()', it asks the client machine what the time is on its system.

What does this mean?

ESR:

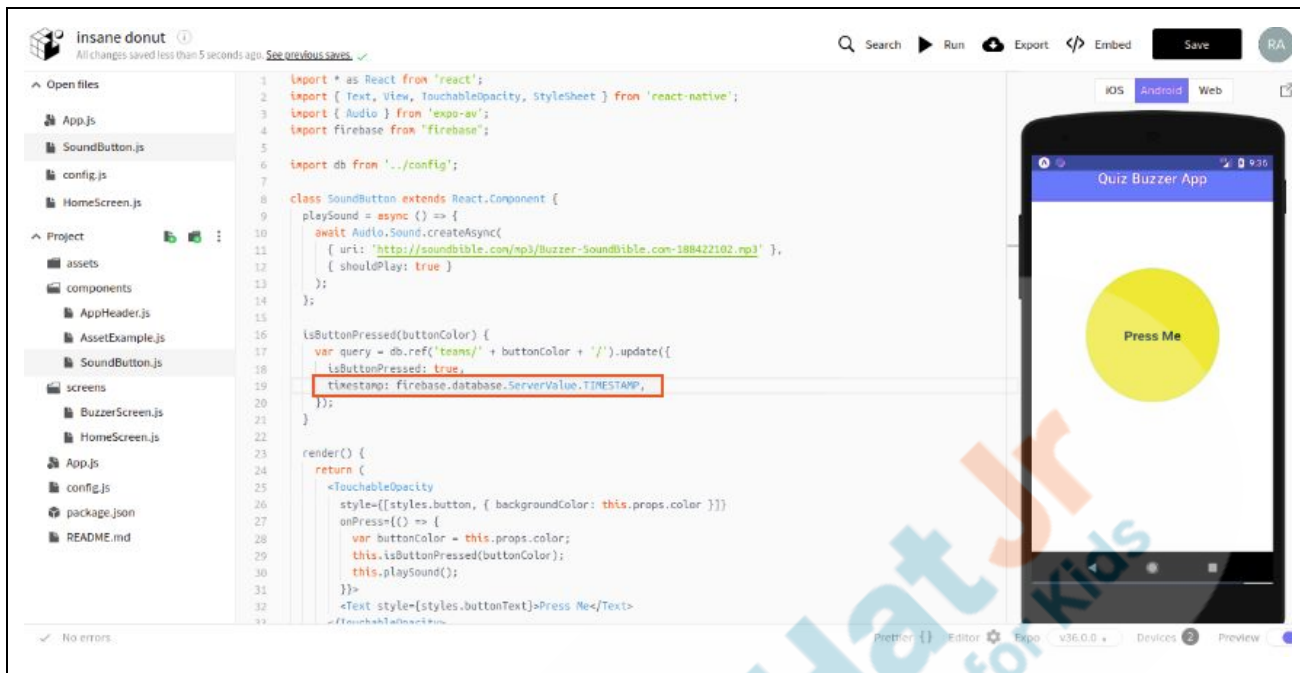
This means that a team can change the time on their system to a previous time and always appear first in the Quiz Master App.

Do you want to try it and see?

The student and teacher run the Quiz Buzzer App on Expo client after changing the time on their smartphone.

The student sees how the code picks up client time

		and arranges the teams according to that.
	What can we do to fix this?	ESR: varied
	<p>Instead of taking the time from the client machine, we should be taking the time from the server.</p> <p>Server in our application is the firebase server where our data is getting stored.</p> <p>Google's servers are keeping the time on the firebase server and it is impossible for any team to change that.</p> <p>Thankfully, we can fetch timestamp directly from the firebase server and set the timestamp for each team.</p> <p>Teacher shows how to get a timestamp directly from the firebase server.</p>	The student observes and asks questions.



```

1  import * as React from 'react';
2  import { Text, View, TouchableOpacity, StyleSheet } from 'react-native';
3  import { Audio } from 'expo-av';
4  import firebase from 'firebase';
5
6  import db from '../config';
7
8  class SoundButton extends React.Component {
9    playSound = async () => {
10      await Audio.Sound.createAsync(
11        { uri: 'http://soundbible.com/mp3/Buzzer-SoundBible.com-188422102.mp3' },
12        { shouldPlay: true }
13      );
14    };
15
16    isButtonPressed(buttonColor) {
17      var query = db.ref('teams/' + buttonColor + '/').update({
18        isButtonPressed: true,
19        timestamp: firebase.database.ServerValue.TIMESTAMP,
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 style={styles.buttonText}>Press Me</Text>
33      );
34    }
35  }

```

Great! Now let's test the app again.

The student and teacher test the app again by changing the time on the local client machine.

Wow! Great! We fixed the bug in the app where anyone could potentially cheat.


Now, there is one bug I would like you to fix.

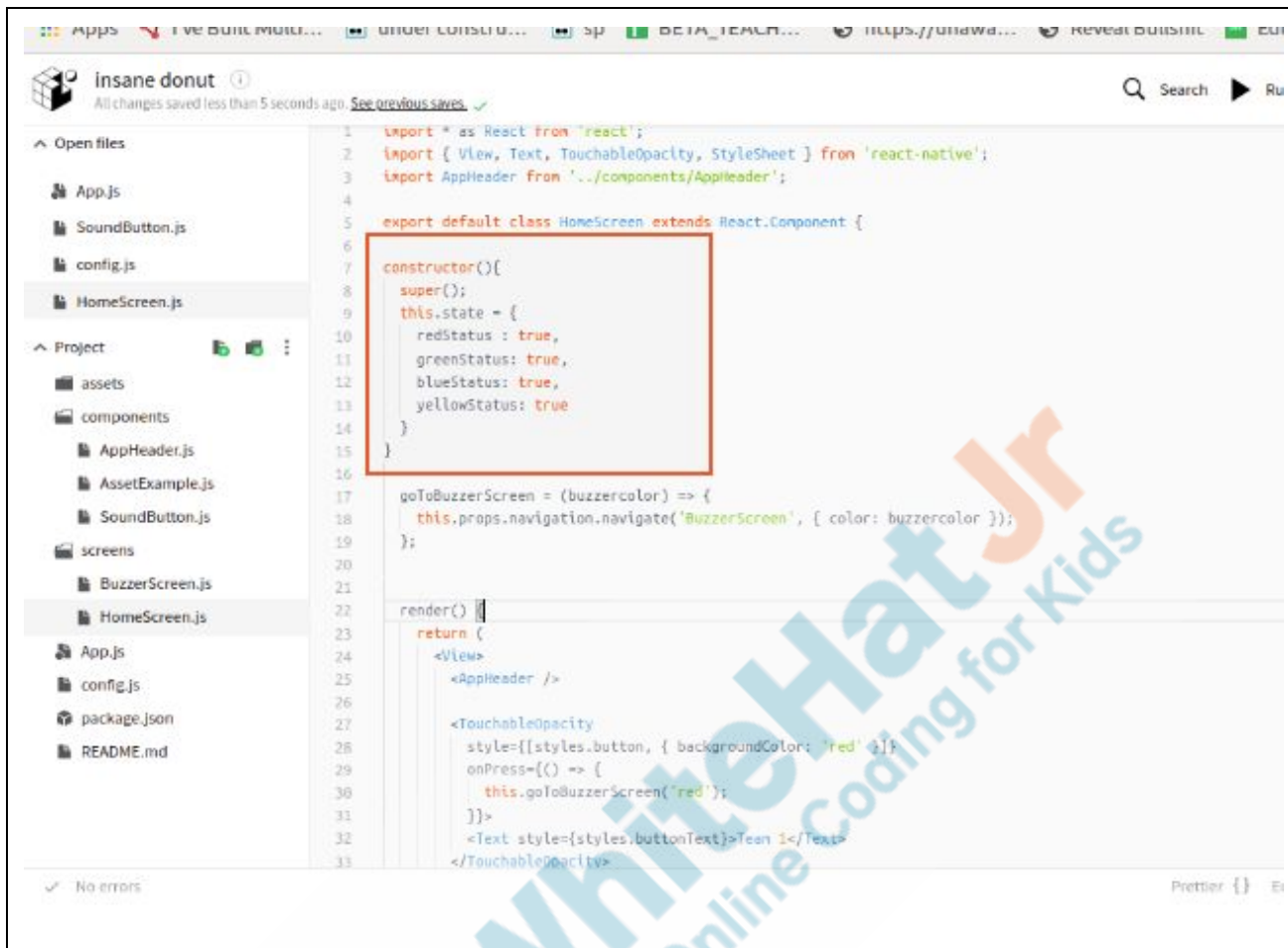
Currently more than two teams can opt for the same color button.
For example: Two teams can choose red.
Ideally, we would like the button to get deactivated once a team is chosen.

Can you try to do that?

ESR:
Yes

	I will help you wherever you get stuck. Let's get started.	
Teacher Stops Screen Share		
	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"> • Write code to make the buttons inactive once a team has chosen a color. 		
Step 3: Student-Led Activity (15 min)	Before we start working on the task, let's take a moment to think about what we are going to do.	The student spends some time thinking, asking questions and discussing how to disable the button once a team selects it.
	<p>The component 'TouchableOpacity' which we are using to create the buttons has a prop called 'disabled'.</p> <p>If 'disabled' is set to 'true', the button will become inactive in our app.</p> <p>We will also need to have something stored in our database which will tell whether a team has chosen a color or not.</p> <p>When the app renders on the screen, it will read from the database and make the button inactive depending on which particular teams have been selected.</p>	The student listens and asks questions.

	Let's create a new field inside 'teams' called "enabled". This should reflect whether the button should be enabled or not.	The student creates a new field called "enabled" inside each team and sets its value to true.
		
	<p>Great, now we want to read the enabled status for each team when we render the buttons.</p> <p>We can declare the status of each key inside the state which keeps track of their enabled status for the buttons.</p>	In 'HomeScreen.js', in the HomeScreen class constructor, the student creates a state for each of the buttons.



```

1  import * as React from 'react';
2  import { View, Text, TouchableOpacity, StyleSheet } from 'react-native';
3  import AppHeader from '../components/AppHeader';
4
5  export default class HomeScreen extends React.Component {
6
7    constructor(){
8      super();
9      this.state = {
10        redStatus: true,
11        greenStatus: true,
12        blueStatus: true,
13        yellowStatus: true
14      }
15    }
16
17    gotoBuzzerScreen = (buzzercolor) => {
18      this.props.navigation.navigate('BuzzerScreen', { color: buzzercolor });
19    };
20
21    render() {
22      return (
23        <View>
24          <AppHeader />
25
26          <TouchableOpacity
27            style={styles.button, { backgroundColor: 'red' }}
28            onPress={() => {
29              this.gotoBuzzerScreen('red');
30            }}
31          >
32            <Text style={styles.buttonText}>Team 1</Text>
33          </TouchableOpacity>

```

	<p>We want to read the enabled status for each team when the component mounts so that we can assign that to the state.</p> <p>Where can we do that?</p>	<p>ESR: Inside 'componentDidMount()'</p>
	<p>Let's do that then.</p>	<p>The student creates a database reference in 'componentDidMount()' and changes the state of each button depending on what it is in the database.</p>

```

20 teamRef.update({
21   enabled: false
22 });
23 this.props.navigation.navigate('BuzzerScreen', { color: buzzercolor });
24
25
26
27
28 componentDidMount(){
29   var teamRef = db.ref('teams');
30   teamRef.on('value', data => {
31     var teamDetails = data.val();
32     this.setState({
33       redStatus: teamDetails.red.enabled,
34       blueStatus: teamDetails.blue.enabled,
35       yellowStatus: teamDetails.yellow.enabled,
36       greenStatus: teamDetails.green.enabled
37     });
38   });
39 }
40
41 render() {
42   return (
43     <View>

```

Now we can simply assign the disabled state of the 'TouchableOpacity' component of each button depending on the state of that button stored in the component state.

The student assigns the disabled prop for each button.

Note : ! is used for the word 'NOT'


!true = false


!false = true



```

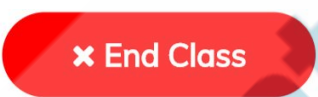
55 style={styles.button, { backgroundColor: 'green' }}}
56 onPress={() => {
57   this.goToBuzzerScreen('green');
58 }}
59 <Text style={styles.buttonText}>Team 2</Text>
60 </TouchableOpacity>
61
62 <TouchableOpacity
63   disabled={!this.state.blueStatus}
64   style={styles.button, { backgroundColor: 'blue' }}
65   onPress={() => {
66     this.goToBuzzerScreen('blue');
67   }}
68 <Text style={styles.buttonText}>Team 3</Text>
69 </TouchableOpacity>
70
71 <TouchableOpacity
72   disabled={!this.state.yellowStatus}
73   style={styles.button, { backgroundColor: 'yellow' }}
74   onPress={() => {
75     this.goToBuzzerScreen('yellow');
76   }}

```

	<p>Let's test if the buttons get disabled by manually changing the enabled state in the database.</p>	<p>The student manually changes the enabled state in the database for the buttons to check if the buttons on the homescreen get disabled.</p>
	<p>Now one final thing.</p> <p>When we are clicking on the team button to navigate to a different screen, we should also update the enabled state for the team in the database to false.</p> <p>If this isn't done, the code will still the button state from the database and keep it enabled.</p>	<p>The student updates the enabled state in the database for each team when the team button is pressed on the Homescreen.</p>
 <pre> 14 yellowStatus: true 15 } 16 } 17 18 goToBuzzerScreen = (buzzercolor) => { 19 var teamRef = db.ref('teams/' + buzzercolor); 20 teamRef.update({ 21 enabled: false 22 }); 23 this.props.navigation.navigate('BuzzerScreen', { color: buzzercolor }); 24 }; 25 26 componentDidMount(){ 27 var teamsref = db.ref("teams"); 28 teamsref.on("value", data => { 29 var teamDetails = data.val(); 30 this.setState({ 31 redStatus: teamDetails.red.enabled, 32 blueStatus: teamDetails.blue.enabled, 33 yellowStatus: teamDetails.yellow.enabled, 34 greenStatus: teamDetails.green.enabled 35 }); </pre>		
	<p>Let's test our app now.</p>	<p>The student tests the app to check if everything is working as expected.</p>

	Amazing! We have been able to fix quite a few bugs in this class.	
Teacher Guides Student to Stop Screen Share		
<p align="center"><u>FEEDBACK</u></p> <ul style="list-style-type: none"> Encourage the student to look for more bugs and fix them. 		
Step 4: Wrap-Up (5 min)	<p>There might be more bugs in our Buzzer App. So here is a challenge for you. Try to find and fix as many bugs as you can.</p> <p>Programming is an iterative process of writing code, finding bugs and fixing them to make your program better.</p>	The student takes up the challenge.
	<p>In the next class, we will actually be converting our Wireless Buzzer and Quiz Master app into the apk and ipa for android playstore and appstore publishing.</p> <p>Excited?</p>	ESR: Yes!
	Also, so far we have been coding in an online editor called expo snack. In the next class, we will learn how to set our system up so that we can code on our local machine, inside our very own visual code studio!	
	<p>You get a “hats off”.</p> <p>Till next class then. See you.</p>	<p>Make sure you have given at least 2 Hats Off during the class for:</p> <div>  +10 Creatively Solved Activities </div>

		<div>Great Question  +10</div> <div>Strong Concentration  +10</div>
Project Pointers and Cues (5 min)	<p>CALL OF DUTY</p> <p>Goal of the Project:</p> <p>Today you have learnt about debugging your application.</p> <p>In this project, you have an app created using concepts learnt so far. It has a lot of bugs in it. So you will have to debug the application.</p> <p>Story:</p> <p>Good evening Agent X! Terrorists have attacked Mumbai and have planted biochemical bomb in the city. Our team had ambushed the terrorist eliminating all of them. The bomb is controlled electronically by a program and can only be diffused through the controlling program. The last terrorist, before being eliminated, modified the code. The bomb can only be defused by running code.</p> <p>I am very excited to see your project solution and I know you both will do really well.</p> <p>Bye Bye!</p>	
	We are almost there. The most awaited class! Yes. We are talking	

	<p>about the Capstone class! Are you prepared to rise and shine?</p> <p>In the upcoming class, we will build a native app in the local expo environment to forecast weather.</p> <p>Please request your parents to join the class.</p> <p>Bye Bye!</p>	
<p style="text-align: center;">Teacher Clicks</p> <div style="text-align: center;">  </div>		
Additional Activities	Encourage the student to identify and fix other bugs in the app.	The student identifies and fixes other bugs in the app.
	<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	Quiz Master App	https://snack.expo.io/@rajeevtfi/quiz-master-app
Teacher Activity 2	Quiz Buzzer App	https://snack.expo.io/@rajeevtfi/student-activity-1-reference:-database
Student Activity 1	Quiz Master App	https://snack.expo.io/@rajeevtfi/quiz-master-app
Student Activity 2	Quiz Buzzer App	https://snack.expo.io/@rajeevtfi/student-activity-1-reference:-database
Teacher Reference	Final Reference	https://snack.expo.io/@rajeevtfi/3eff2d