

Fixing Bugs on the Buzzer App



What is our GOAL for this MODULE?

We fixed the timestamp bug in our Quiz Buzzer app. We also learned to make the buttons inactive once a team was chosen.

What did we ACHIEVE in the class TODAY?

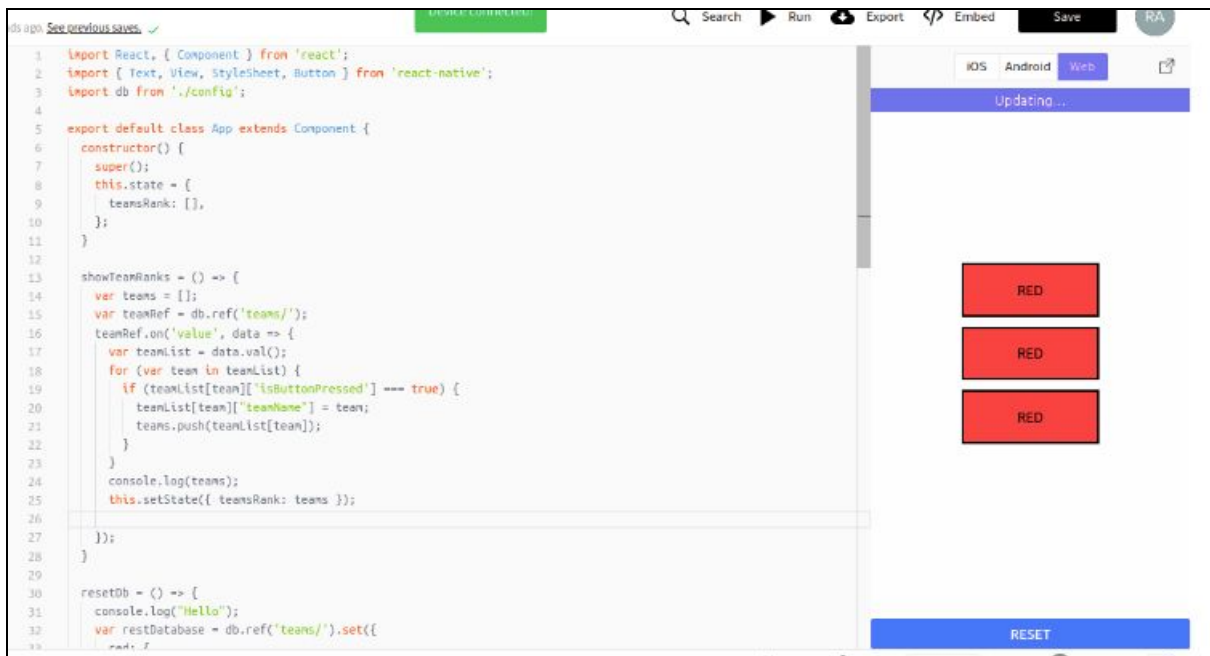
- Fixed the timestamp bug.
- Made buttons inactive once a team was chosen.

Which CONCEPTS/CODING BLOCKS did we cover today?

- Timestamp
- Debugging

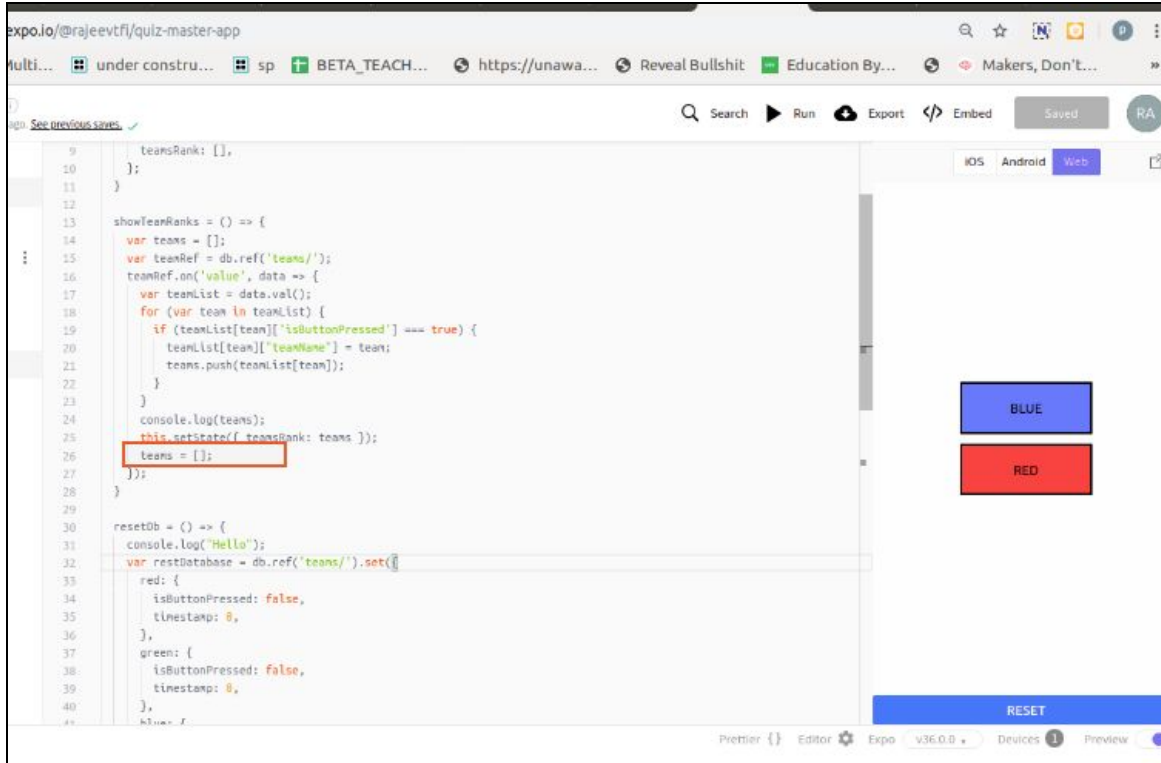
How did we DO the activities?

1. Open the Quiz Buzzer and Quiz Master App.
2. Here, we can see the bug.
 - When the user presses the button repeatedly, the team name comes on the quiz master app repeatedly.
 - 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.



```
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       // ...
33     });
34   }
35 }
```

- Fixing the bug: If we simply empty the array every time, our code will always read from the database again and there will be no duplicity entered in the array.

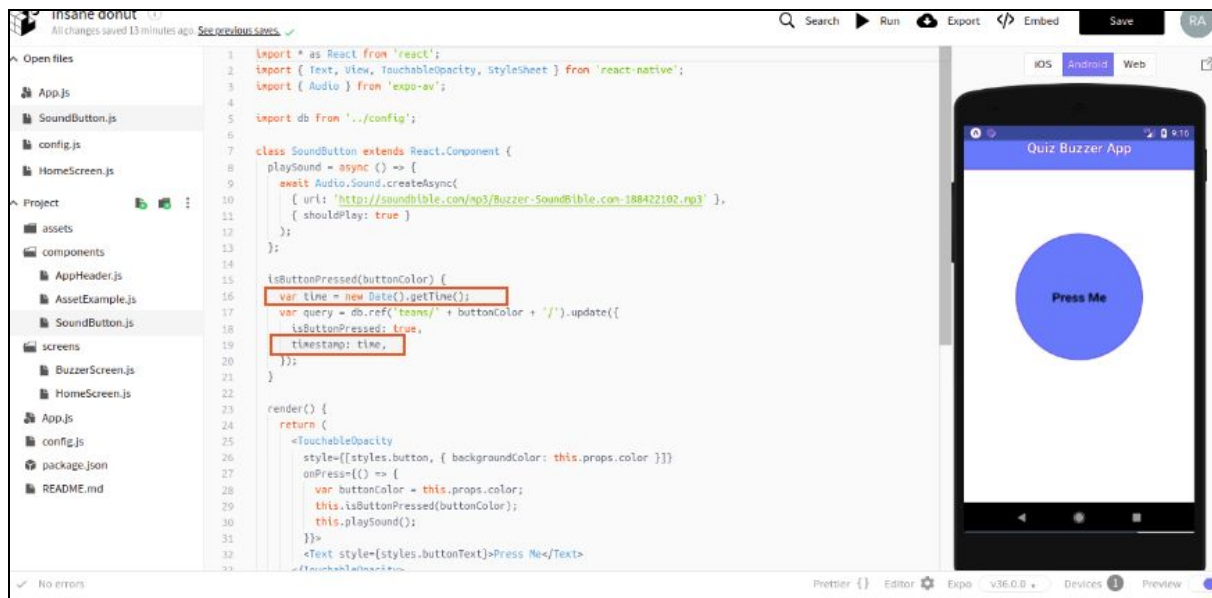


```

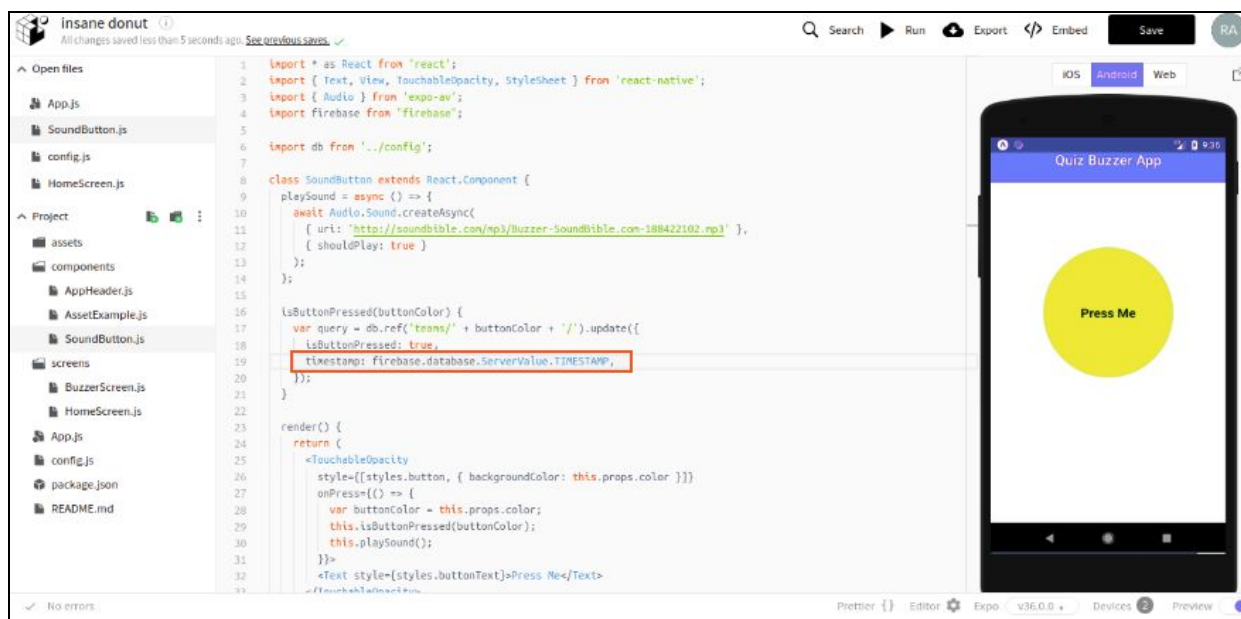
9      teamsRank: [],
10    };
11  }
12
13  showTeamRanks = () => {
14    var teams = [];
15    var teamRef = db.ref('teams/');
16    teamRef.on('value', data => {
17      var teamList = data.val();
18      for (var team in teamList) {
19        if (teamList[team]['isButtonPressed'] === true) {
20          teamList[team]['teamName'] = team;
21          teams.push(teamList[team]);
22        }
23      }
24      console.log(teams);
25      this.setState({ teamsRank: teams });
26      teams = [];
27    });
28  }
29
30  resetDb = () => {
31    console.log("Hello");
32    var restDatabase = db.ref('teams/').set({
33      red: {
34        isButtonPressed: false,
35        timestamp: 0,
36      },
37      green: {
38        isButtonPressed: false,
39        timestamp: 0,
40      },
41    });
  
```

3. Another bug: Let's look at the Quiz Buzzer App (SoundButton.js).

- 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.
- 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.
- 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

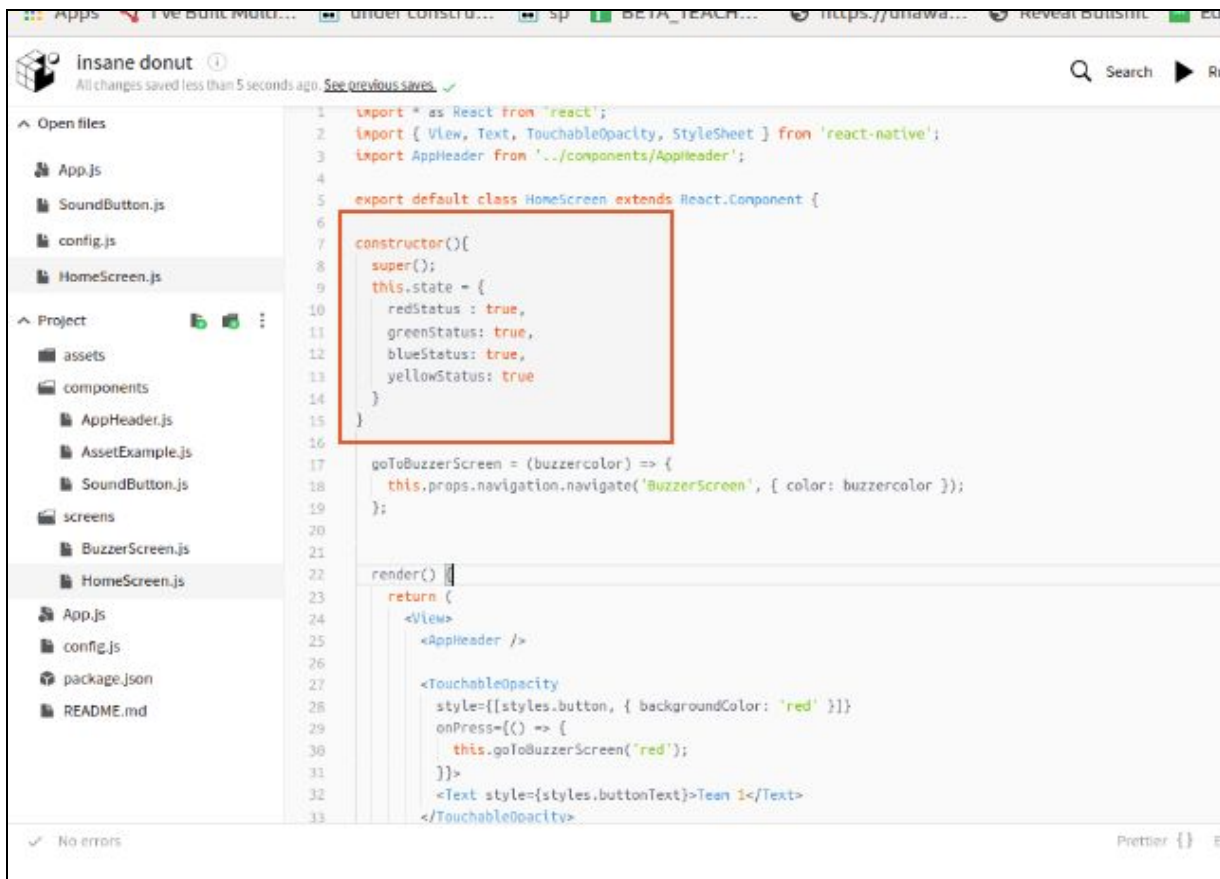


- To fix this bug:
 - Instead of taking the time from the client machine, we should be taking the time from the server. The server in our application is the firebase server where our data is getting stored.
 - Google's servers are keeping the time on the firebase server and it is impossible for any team to change that.
 - We can fetch the timestamp directly from the firebase server and set the timestamp for each team.



4. There is one more bug we fixed:

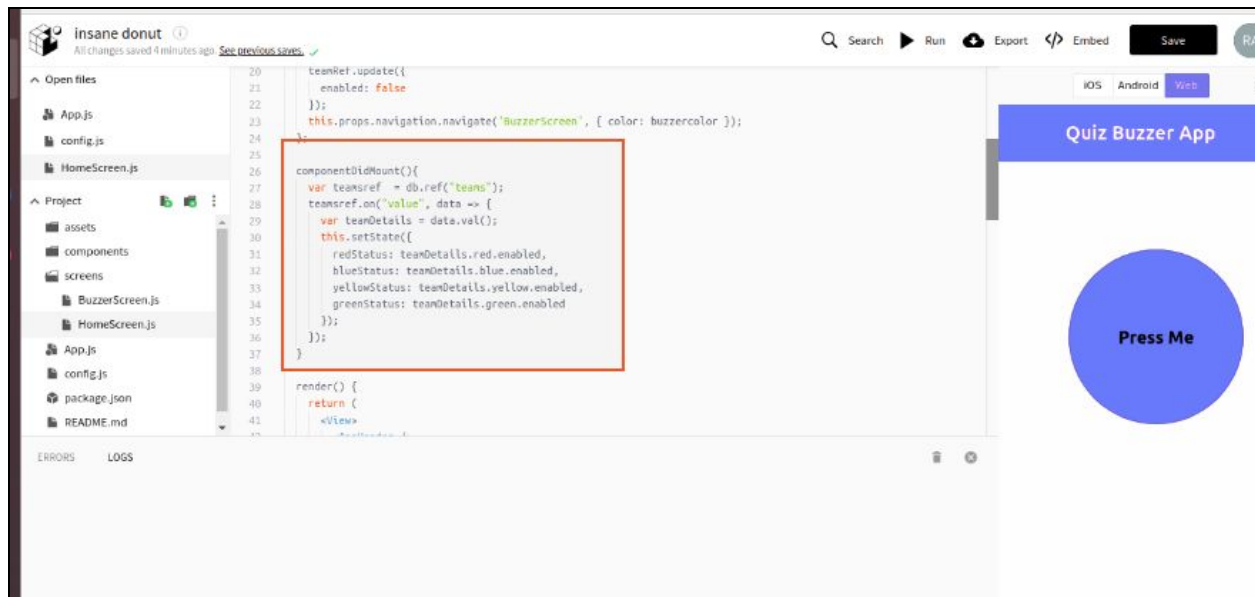
- Currently, more than two teams can opt for the same color button.
For example, Two teams can choose red.
- Ideally, we would like the button to get deactivated once a team is chosen.
- To fix it:
 - The component 'TouchableOpacity' which we are using to create the buttons has a prop called 'disabled'.
 - If 'disabled' is set to 'true', the button will become inactive in our app.
 - We will also need to have something stored in our database which will tell whether a team has chosen the colour or not.
 - 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.
 - Let's create a new field inside 'teams' called "enabled". This should reflect whether the button should be enabled or not.
 - Now, we want to read the enabled status for each team when we render the buttons.
 - We can declare the status of each key inside the state which keeps track of their enabled status for the buttons.



```

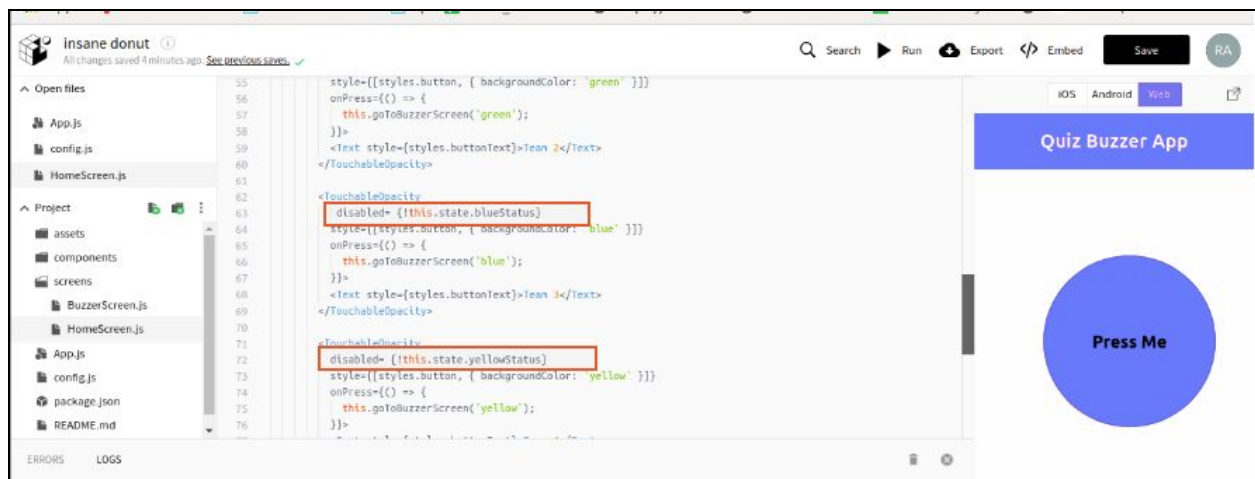
1  import * as React from 'react';
2  import { View, Text, TouchableOpacity, StyleSheet } from 'react-native';
3  import AppHeader from '../components/AppHeader';
4
5  export default class HomeScreen extends React.Component {
6
7    constructor() {
8      super();
9      this.state = {
10        redStatus: 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          <TouchableOpacity
26            style={[styles.button, { backgroundColor: 'red' }]}
27            onPress={() => {
28              this.goToBuzzerScreen('red');
29            }}
30          >
31            <Text style={styles.buttonText}>Team 1</Text>
32          </TouchableOpacity>
33        </View>
34      );
35    }
36  }
  
```

- We want to read the enabled status for each team when the component mounts so that we can assign that to the state.
- For this, we created a database reference in 'componentDidMount()' and changed the state of each button depending on what it is in the database.



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

*Note : ! is used for the word 'NOT' !true = false !false = true



- 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.
- If this isn't done, the code will still the button state from the database and keep it enabled.



```
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     });
36   });
37 }
```

What's NEXT?

In the next class, we will actually be converting our Wireless Buzzer and Quiz Master app into the apk and ipa for android Play Store and AppStore publishing.