Topic	A Problem Statement: Wireless Buzzer App	
Class Description	**Students will learn how to create their own prop for a component. Students learn to execute an action on the event of a button press using the "onPress" button prop and by writing another function inside a class. Students display an alert box when a button is pressed. Students also work on creating a wireframe for an application based on a problem statement.**	
Class	**C54**	
Class time	**45 mins**	
Goal	● Create custom prop for a custom component. ● Write a function inside a React Component. ● Execute a function when a Button onPress event happens.	
Resources Required	● Teacher Resources ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Android/iOS Smartphone with Expo App installed ● Student Resources ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Android/iOS Smartphone with Expo App installed	
Class structure	**Warm Up** **Teacher-led Activity** **Student-led Activity** **Wrap up**	**5 mins** **10 mins** **20 min** **5 min**

CONTEXT

- **Review last class - React native platform, components and their props.**

Class Steps	Teacher Action	Student Action
Step 1: Warm Up (5 mins)	Hello! We had started working on the React Native platform in the last class. Was it exciting for you?	ESR: Yes!

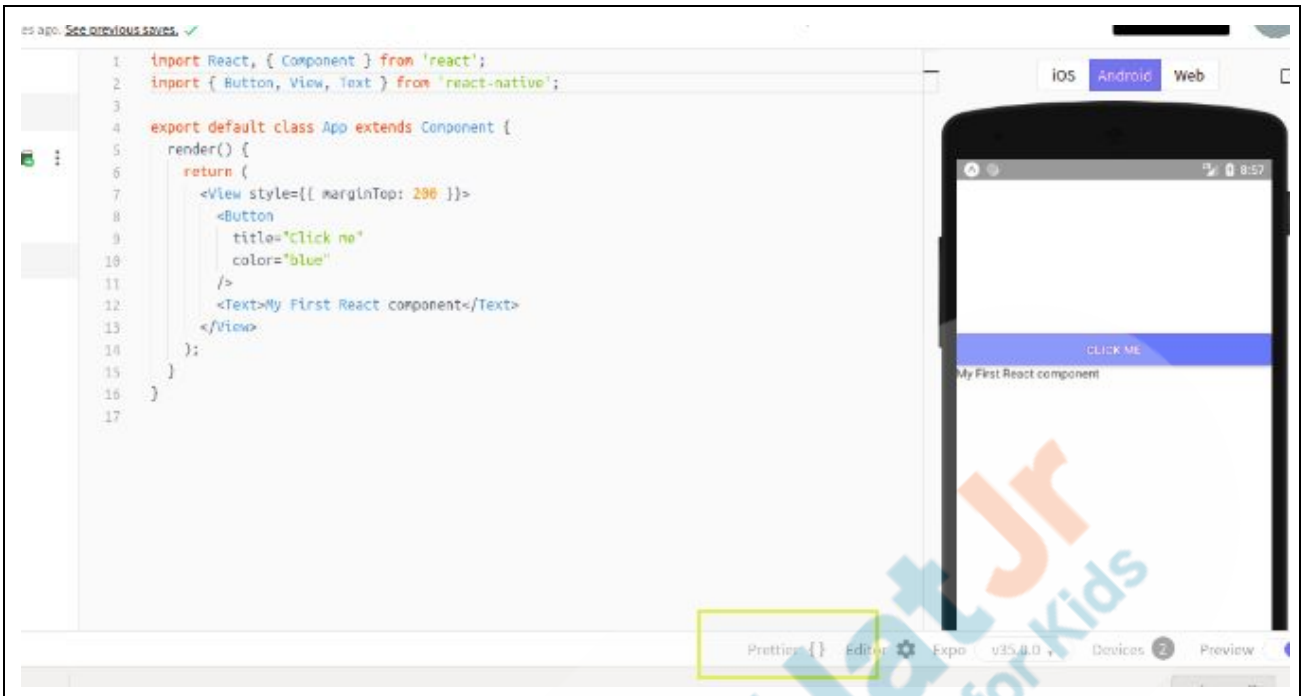
	Can you quickly recall what we learned?	ESR: - We learned how React Native uses components to make app development declarative. - Everything inside React native is made up of components - including the app itself. - We learned how components are created using class and extending the Component class already defined in React library. - We learned about some of the React Native components - Text, View, Button and their properties. - We learned how to create our own custom React Native component.
	Awesome! That was a heck of a start! But remember the button we created does not do anything as of now. Today we will learn how to add functionality to our button so that something happens when we press a button. Excited?	ESR: Yes!
	We will also start working on a case study for a problem. We will be creating an app for that problem in the next few classes. We will start working on the app by this class itself.	

	Amazing, isn't it! So, let's get started!!	ESR: yes!
Teacher Initiates Screen Share		
CHALLENGE • **Execute an action/function when a Button is pressed in the app using onPress Button prop.** • **Create custom prop for the custom React Component.**		
Step 2: Teacher-led Activity (10 mins)	Teacher Opens **[Teacher Activity 1]** Can you quickly go over the last class code and explain what is happening here?	ESR: The student reviews the code from the last class.

```
import React, { Component } from 'react';
import { Button, View, Text } from 'react-native';

class RedButton extends Component {
  render(){
    return(
      <Button title="Click Me" color="red" />
    );
  }
}
export default class App extends Component {
  render() {
    return (
      <View style={{marginTop: 200}}>
        <RedButton />
        <Text>My First React component</Text>
      </View>
    );
  }
}
```

	What is View?	ESR: View is a component defined in React native. It is an empty container which can contain other components.
	What does a render function inside a React component do? How many components can a render() function return? How do we return more than one component in React native?	The render function displays/renders the component returned by the function. render() can return only one component. We return more than one component in react native by nesting them inside a View Component.
	Awesome! You seem to be on the top of React Native so far. One last question - what are the 'title', 'color' inside the Button Component?	The title and color are called the properties of the Button component. Title is used to display the text inside the button. color is used to add color to the button.
	Excellent! In react native vocabulary, they are called 'props' for the component. We will learn to create our own custom props or properties for the component we defined. Before we do that, we will explore how to use another prop of the Button Component called "onPress".	Student listens and asks questions.

	This will help us do something when the button is pressed.	
	Teacher opens **[Teacher Activity 2]** Ok so what do you see here? Awesome! By the way, do you see the change in the style we are using to write props? Yes! It is easy on the eye to read this way. Remember how readability of our code is most important for any program. By the way, there is a feature in snack called “{} Prettier” at the bottom. You can use it to convert your code into a more readable way of writing.	A button and a text component nested inside View. The button component has two props - title and color defined on it. Yes, all the props are vertically aligned instead of being in the same way.



	Let's say we want to display an alert box when we click the button in our app. We can write a function inside our app class which does that. Let's call this function 'displayAlert'. Teacher shows by writing a function inside the App class.	Student observes.

```
import React, { Component } from 'react';
import { Button, View, Text } from 'react-native';

export default class App extends Component {
  displayAlert(){


  }
  render() {
    return (
      <View style={{ marginTop: 200 }}>
        <Button title="Click me" color="blue" />
        <Text>My First React component</Text>
      </View>
    );
  }
}
```

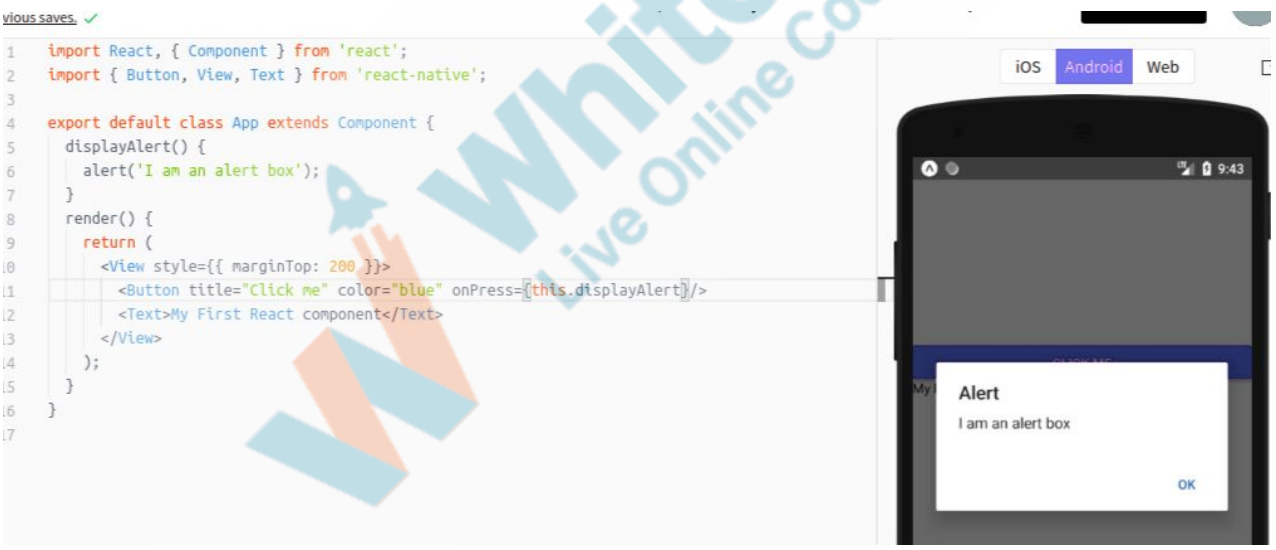
	How do we instruct the computer to display an alert inside this function? There is a function called alert() which will do that for us. I am going to write this inside the displayAlert function. Will anything happen when we click on the button now? Why?	ESR: varied Student observes. ESR: No! Because we have created this function but have no't called it anywhere.

```
import React, { Component } from 'react';
import { Button, View, Text } from 'react-native';

export default class App extends Component {
  displayAlert(){
    alert('I am an alert box');
  }
  render() {
    return (
      <View style={{ marginTop: 200 }}>
        <Button title="Click me" color="blue" />
        <Text>My First React component</Text>
      </View>
    );
  }
}

```

	Yes! All we have created is a function inside the App class. Now the App class should call this function. Button Component has another prop called 'onPress'. We can use this to call the 'displayAlert' function. Teacher shows how to call the 'displayAlert' function. Remember, 'displayAlert' is a function of App object. So it should be called using 'this' notation. Remember: 'this' points to the object which is created using the class and calls the function? Also, we are using Javascript inside JSX tags. We write javascript inside JSX in curly brackets.	Student observes and asks questions.

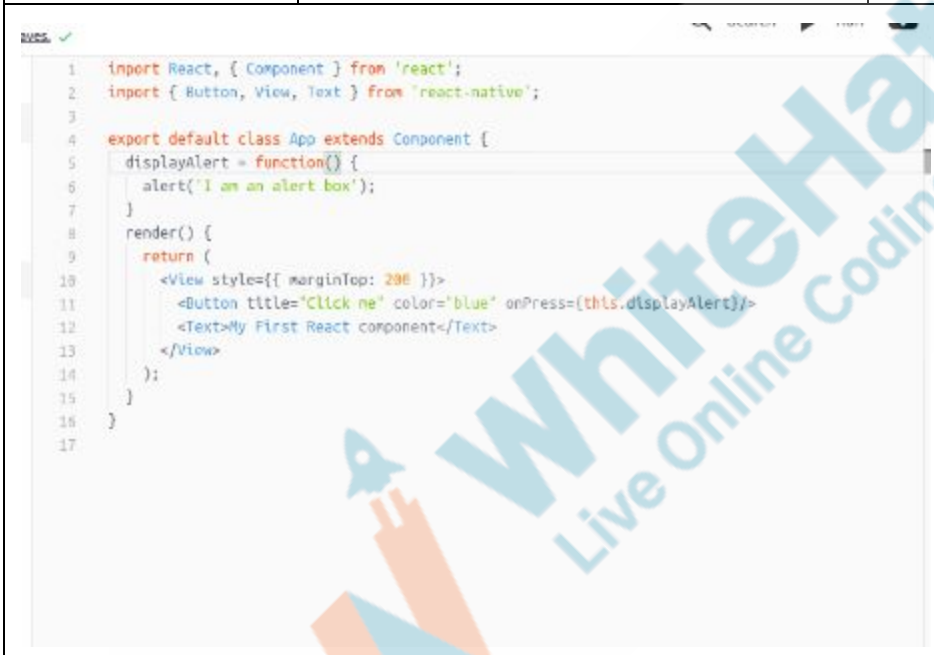
```
import React, { Component } from 'react';
import { Button, View, Text } from 'react-native';

export default class App extends Component {
  displayAlert() {
    alert('I am an alert box');
  }
  render() {
    return (
      <View style={{ marginTop: 200 }}>
        <Button title="Click me" color="blue" onPress={this.displayAlert}/>
        <Text>My First React component</Text>
      </View>
    );
  }
}
```

	Let's run the code and see what happens when we press the button.	Student observes the output.



	Great! It seems to work. There are different styles of writing a function in javascript. Teacher shows some other styles of writing function.	ESR: Yes!

	One of the styles is called an ‘arrow’ function. You have already used it in earlier classes. An arrow function binds ‘this’ to the root object rather than the component which is calling it. For example: Here inside the arrow function display, ‘this’ will bind to the App and not the button which will be calling the displayAlert function.	



```
1  import React, { Component } from 'react';
2  import { Button, View, Text } from 'react-native';
3
4  export default class App extends Component {
5    displayAlert = function() {
6      alert('I am an alert box');
7    }
8    render() {
9      return (
10       <View style={{ marginTop: 200 }}>
11         <Button title='Click me' color='blue' onPress={this.displayAlert}/>
12         <Text>My First React component</Text>
13       </View>
14     );
15   }
16 }
17
```

```
import React, { Component } from 'react';
import { Button, View, Text } from 'react-native';

export default class App extends Component {
  displayAlert = ()=>{
    alert('I am an alert box');
  }
  render() {
    return (
      <View style={{ marginTop: 200 }}>
        <Button title="Click me" color="blue" onPress={this.displayAlert}/>
        <Text>My First React component</Text>
      </View>
    );
  }
}
```

	Awesome, we have created an alert on pressing a Button for the App component. Can you do this on a custom defined RedButton component which we created in the last class? After that we will learn how to define a prop for our own custom component. Let's get started.	
Teacher Stops Screen Share		
	Now it's your turn. Please share your screen with me.	

- **Ask Student to press ESC key to come back to panel**
- **Guide Student to start Screen Share**
- **Teacher gets into Fullscreen**

ACTIVITY

- **Define the class for the React native component.**
- **Use the React Native component in the app.**

Step 3: Student-Led Activity (20 min)	Alright open **Student Activity 1**. Your task is to write the 'displayAlert' function for the custom RedButton we created in the last class. Guide the student if he/she gets stuck.	Student opens **Student Activity 1**. The student writes a displayAlert function for the RedButton using the 'onPress' prop for the Button.

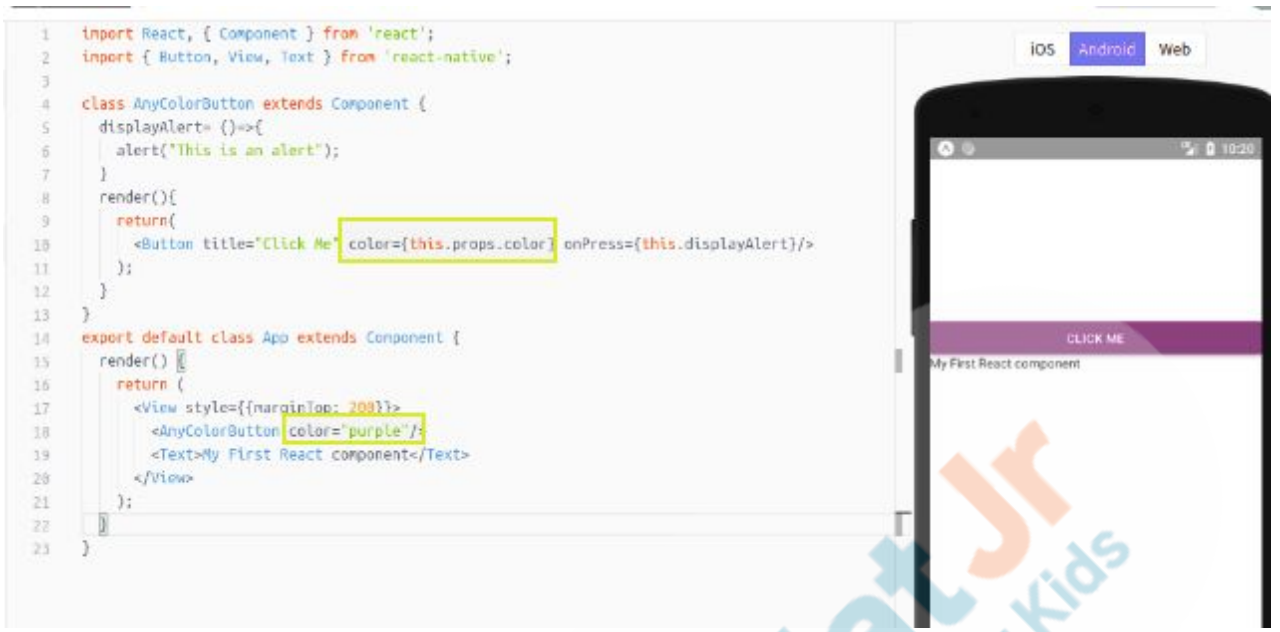
```
import React, { Component } from 'react';
import { Button, View, Text } from 'react-native';

class RedButton extends Component {
  displayAlert= ()=>{
    alert("This is an alert");
  }
  render(){
    return(
      <Button title="Click Me" color="red" onPress={this.displayAlert}/>
    );
  }
}
export default class App extends Component {
  render() {
    return (
      <View style={{marginTop: 200}}>
        <RedButton />
        <Text>My First React component</Text>
      </View>
    );
  }
}
```

	Let's run the code and see what happens.	Student runs the code on the phone and sees the output.

	Awesome! You did it! Now we have seen some props or properties for already defined components - like title, color, onPress for Button; style for Text and View. We will now be learning to create props for our own custom component.	The student listens.
	Let's name our custom component ‘AnyColorButton’ and let us define our own prop called 'mycolor' on it. You can call the prop anything else as well. We can give any color value to our color property in the JSX tags.	The student renames the custom created component. In the JSX tag for the ‘AnyColorComponent’, the student creates a prop called ‘mycolor’ and passes any color to it.

```
import React, { Component } from 'react';
import { Button, View, Text } from 'react-native';

class AnyColorButton extends Component {
  displayAlert= ()=>{
    alert("This is an alert");
  }
  render(){
    return(
      <Button title="Click Me" color="red" onPress={this.displayAlert}/>
    );
  }
}
export default class App extends Component {
  render() {
    return (
      <View style={{marginTop: 200}}>
        <AnyColorButton/>
        <Text>My First React component</Text>
      </View>
    );
  }
}
```

	Now, in our definition for the AnyColorButton, you can directly use the value inside the prop using 'this.props.mycolor'. We can give this color to the Button component inside 'AnyColorButton'. Remember, we are using javascript inside JSX, so we need to put this inside curly brackets.	The student adds the color to the Button component defined inside 'AnyColorButton'.



	Amazing! That's all! This is how you can create props for any component and later use them. Later, we will see why it can be so powerful!	-
	Now we actually know enough about React Native to create an app which solves a problem! Have you ever seen or played Quiz in your school? What happens in the buzzer round.	ESR: yes The team which presses the buzzer first gets to answer and score points.
	Yes! How does your school organize a buzzer round in the quiz?	ESR: varied - the school does no't do buzzer round because school does not have hardware for it.

		- school has purchased hardware for the buzzer or they rent it during quiz competitions. - school uses tricks like having an observer who quickly glances at who raises the hands first. etc.
	Most schools either do not do buzzer rounds in quiz competitions because they do not have the buzzer machine or they purchase/rent expensive hardware or they adopt error prone methods. In the coming few classes, we will be creating an app where four different teams will press a buzzer on their phones and we will know the order in which they pressed the buzzer.	Student is excited about creating the app.
	We already know most of what we need to create this kind of app and we will learn the rest along the way. However, the first step to creating an app is to create a wireframe of the app. Do you know what a wireframe of an app is?	ESR: varied.

	A wireframe of an app is a simple layout of the different elements/screens that will be contained in the app and how it will work. It does not involve any coding! The best way to create a wireframe for an app is to simply sketch what your app will look like using paper and pen. Can you create a simple wireframe of what our Wireless Buzzer App will look like? We will start working on this from the next class.	Student sketches and shows the wireframe of the Wireless Quiz Buzzer App to the teacher.
	Teacher offers feedback on the wireframe designed by the student.	Ideal wireframe will have two screens - Screen1: User can choose the team. Screen 2: A buzzer button which needs to be pressed. Whichever team presses the buzzer, their order gets logged into the database.
Teacher Guides Student to Stop Screen Share		

FEEDBACK

- **Encourage the student to work on the wireframe of the app using the wireframing tools available online.**
- **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)**	Alright, let's quickly capture what we learned in today's class.	- We learned how to do something when a button is pressed using the onPress prop of the button. - We learned to create our own props for the custom components we designed in React native. - And we learned how to start working on an app by creating a wireframe for the app.
	Amazing! And just like that we are ready to start working on our next app, which is Wireless Quiz Buzzer App! Using this app you can play Quiz in your school! How are you feeling?	ESR: varied
	You get a “hats off”. I will be looking forward to working with you on this Buzzer App in the next class. Have a good time till then!	Make sure you have given at least 2 Hats Off during the class for:  

<table>
<tr>
<td>Project Pointers and Cues (5 min)</td>
<td>
<h2>DJ AUDIO MIXER APP - STAGE 1</h2>
Goal of the Project:

Today you learned how to create your own prop for a component. You also learned to execute an action on the event of button press using the “onPress" button prop.

In this project, you will have to practice and apply what you have learned in the class and start with the first level of DJ mixer App!

Story:

Your cousin Nikhil loves music and wants to become a Disc Jockey (DJ). He wants to buy a DJ Sound Mixer instrument but it is too costly and he cannot afford it.

I am very excited to see your project solution and I know you both will do really well.

Bye Bye!
</td>
<td></td>
</tr>
<tr>
<td colspan="3">Teacher Clicks ✖ End Class</td>
</tr>
<tr>
<td>Additional Activities</td>
<td>Encourage the student to work on the wireframe of the app using the wireframing tools available online.</td>
<td>The student can create an account on https://www.mockplus.com/</td>
</tr>
</table>

		and create a wireframe of the app online.
	Encourage the student to write reflection notes in their reflection journal using markdown. Use these as guiding questions: ● What happened today? - 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 code	https://snack.expo.io/@rajeevtfi/alert-button-app:-teacher-activity
Teacher Activity 2	Teacher Reference 1	https://snack.expo.io/@rajeevtfi/alert-button-app:-teacher-activity
Student Activity 1	Alert Button on Custom Component	https://snack.expo.io/@rajeevtfi/my-first-app:-teacher-reference