Topic	**Adding Phonic sounds**	
Class Description	**Student reviews the learnings so far in React Native - react native philosophy, props, states, using pre-designed react components, designing custom react components, importing and exporting components from react native library etc. Students also design a phonicButton component which takes the word chunk and plays the sound of a phonic chunk corresponding to it.**	
Class	**C65**	
Class time	**45 mins**	
Goal	● Review learnings in React Native. ● Design 'phonicButton' component which takes the word chunk and plays the sound of phonic chunk corresponding to it.	
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** **15 min** **15 min** **5 min**

CONTEXT

- **Review the Monkey-Chunky Code Base so far.**

<table>
<tr><th>Class Steps</th><th>Teacher Action</th><th>Student Action</th></tr>
<tr><td><b>Step 1:
Warm Up
(5 mins)</b></td><td>Hello! In the last class, we were almost close to finishing off the Monkey-Chunky App.

Do you remember what was left in the App?</td><td>ESR:
Adding sounds for each word chunk.</td></tr>
<tr><td></td><td>Yes! This is a crucial part of our app and you will be doing that on your own today with little to no help from me.

How do you feel about that?</td><td>ESR:
varied</td></tr>
<tr><td></td><td>Before we get down to doing that, let's quickly review all that we have learned in and about making React Native apps.

We will also quickly review the code base for the Monkey-Chunk App we have built so far.</td><td></td></tr>
<tr><td colspan="3"><b>Teacher Initiates Screen Share</b></td></tr>
<tr><td colspan="3"><b><u>CHALLENGE</u></b>
● <b>Recall the learnings in React Native so far:</b>
<b>- React philosophy</b>
<b>- States and Props</b>
<b>- Using React components</b>
<b>- Designing React components</b>
<b>- Importing/Exporting components from libraries</b>
<b>- Connecting to Remote/local database</b></td></tr>
<tr><td><b>Step 2:
Teacher-led Activity</b></td><td>Alright. So can you explain in your own words what reactive native philosophy?</td><td>React philosophy believes that everything in an application is a component.</td></tr>
</table>

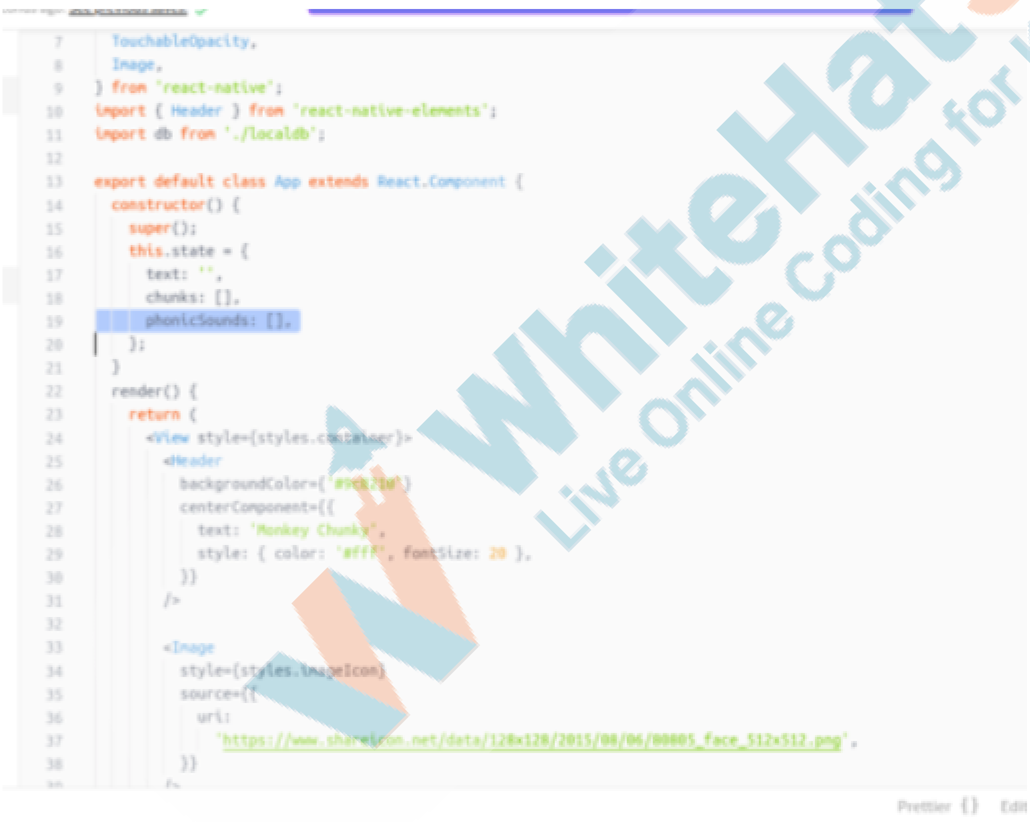
(15 min)		Each component has its own properties, states, functions etc.
	What is the difference between the javascript code we wrote normally while designing games versus the javascript code we are writing now.	The javascript code we wrote normally was imperative - we gave detailed instructions to the computer on what to do and how to do it. The javascript code we write in React native is declarative. We just declare the components we need and what it will do. The how part of it is abstracted inside the component.
	How is declarative code better than imperative code?	Declarative code helps in organizing and makes thinking about our program easier. More organized program allows us to write more complex code bases to do more complex tasks.
	Awesome! You already know why React philosophy is so powerful and how react re-imagines all programs as made up of components. Let's talk more about props and states of a component. What are props and states in a component?	Props are like attributes/properties of a component. It can be accessed using this.props.<propName>. Props can be used to pass data to a component. States are the different states of a component. The component can change

		depending on the current state. For example, a component button can have a state whether it is pressed or not. Depending on that, the component can appear or do different things.
	You are doing very well <student name> Can you also recall the Component Lifecycle?	Every component has a defined lifecycle stages: - Initialization: When the component is assigned its initial values. - Mounting: When the component mounts on the screen. - Updating: When the props or state values of a component change. - Unmounting: When the component unmounts from the screen.
	Brilliant! Do you also remember any of the methods associated with these stages in the lifecycle of React Components:	ESR: Initialization: ‘constructor()’ in the class of the component Mounting: ‘render()’, ‘componentWillMount()’, ‘componentDidMount()’ Updating: ‘shouldComponentUpdate()’, ‘componentWillUpdate()’, ‘componentDidUpdate()’ Unmounting: ‘componentDidUnmount()’, ‘componentWillUnMount’

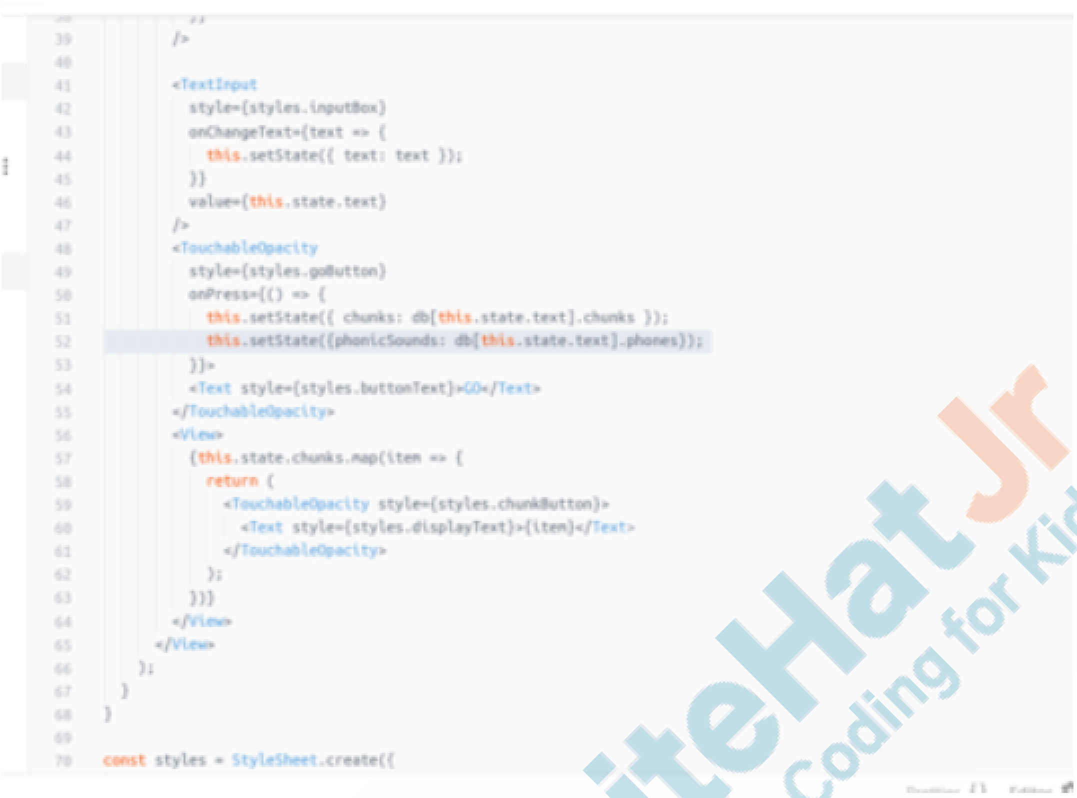
	When are these methods called?	These methods get automatically called at the different stages of the lifecycle.
	Do you remember how to define your own component in React Native?	ESR: Yes. We can create a new component class by extending the Component class from react-native library. We can then export this component. The component can be imported and used inside the App class or any other component.
	What are the pre-defined React native components we have used in our code so far? Note: Let the student explain each of the components and what it does.	ESR: Text, View, StyleSheet etc.
	What are the other libraries and components we have used in our code for Monkey-Chunky or Wireless Buzzer Apps we have created so far? Note: The teacher can ask the student to revisit the links for these projects and recall how these components were used.	ESR: firebase, Audio from expo-av.
	We also used data in our app in two different ways. Can you recall them?	ESR: We learned how to access data from a remote

		database server using firebase. We also learned how to access data locally from a local json file.
	You remember a great deal! You are in a perfect place to finish off the final crucial part of the Monkey-Chunky app where we add sounds to our word chunks. 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**		
<u>ACTIVITY</u> ● **Design a phonic button which takes the word chunk and plays the sound of phonic chunk corresponding to it.**		
Step 3: **Student-Led Activity** **(15 min)**	Let's quickly open the last project and review the code base. Can you loudly articulate what we are doing in our codebase?	Student opens **<u>Student Activity 1</u>** and articulates what the code is doing so far.

	So far, when the ‘Go’ button is pressed, we are only getting the word chunks. Now, we can also get the phonic sound chunks. We will have to declare another state called ‘phonicSounds’ as an empty array. This will hold the phonic sounds associated with each word chunk.	The student creates a new state called ‘phonicSounds’ and gets the phonic sound names associated with each word chunk inside it.

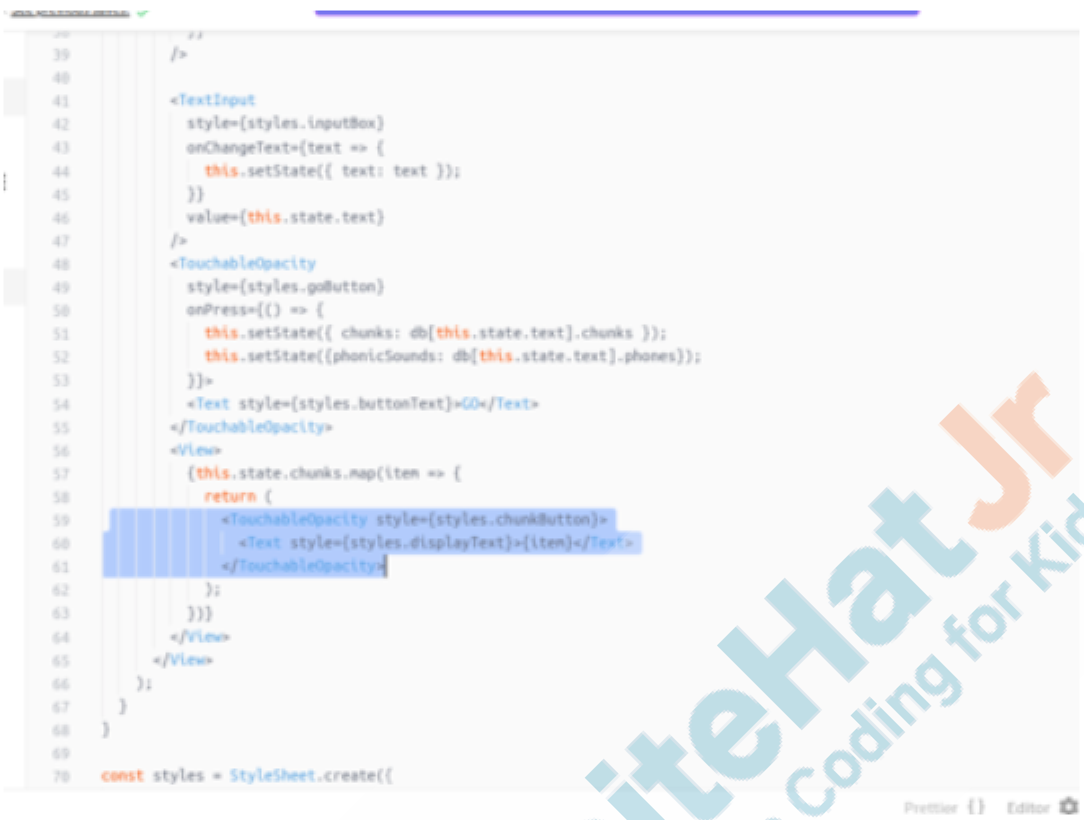


```
    TouchableOpacity,
    Image,
  } from 'react-native';
  import { Header } from 'react-native-elements';
  import db from './localdb';

  export default class App extends React.Component {
    constructor() {
      super();
      this.state = {
        text: '',
        chunks: [],
        phonicSounds: [],
      };
    }
    render() {
      return (
        <View style={styles.container}>
          <Header
            backgroundColor={'#9c8210'}
            centerComponent={{
              text: 'Monkey Chunky',
              style: { color: '#fff', fontSize: 20 },
            }}
          />

          <Image
            style={styles.imageIcon}
            source={{
              uri:
                'https://www.shareicon.net/data/128x128/2015/08/06/80805_face_512x512.png',
            }}
          />
```



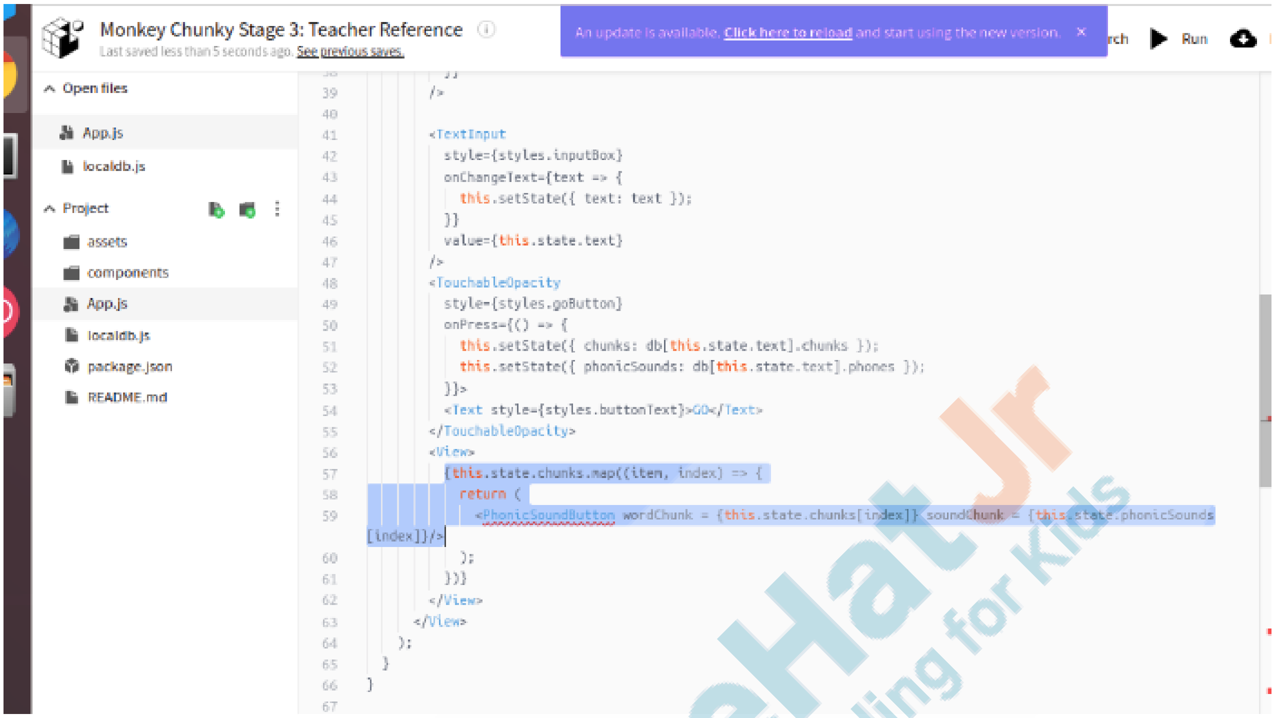
```
39          />
40
41          <TextInput
42            style={styles.inputBox}
43            onChangeText={text => {
44              this.setState({ text: text });
45            }}
46            value={this.state.text}
47          />
48          <TouchableOpacity
49            style={styles.goButton}
50            onPress={() => {
51              this.setState({ chunks: db[this.state.text].chunks });
52              this.setState({phonicSounds: db[this.state.text].phones});
53            }}>
54            <Text style={styles.buttonText}>GO</Text>
55          </TouchableOpacity>
56          <View>
57            {this.state.chunks.map(item => {
58              return (
59                <TouchableOpacity style={styles.chunkButton}>
60                  <Text style={styles.displayText}>{item}</Text>
61                </TouchableOpacity>
62              );
63            })}
64          </View>
65        </View>
66      );
67    }
68  }
69
70  const styles = StyleSheet.create({
```

	If you look in line numbers 59 to 61, we are mapping over each word chunk and creating a 'TouchableOpacity' button for each chunk. Now, we have to write code so that pressing the button plays the sound corresponding to the word chunk. Since this component has a distinct functionality of its own, we can create a separate component of its own. Let's call this component 'PhonicSoundButton'. It can take the word chunk and phone as props.	Student writes the related code.



```
          />

          <TextInput
            style={styles.inputBox}
            onChangeText={text => {
              this.setState({ text: text });
            }}
            value={this.state.text}
          />
          <TouchableOpacity
            style={styles.goButton}
            onPress={() => {
              this.setState({ chunks: db[this.state.text].chunks });
              this.setState({phonicSounds: db[this.state.text].phones});
            }}>
            <Text style={styles.buttonText}>GO</Text>
          </TouchableOpacity>
          <View>
            {this.state.chunks.map(item => {
              return (
                <TouchableOpacity style={styles.chunkButton}>
                  <Text style={styles.displayText}>{item}</Text>
                </TouchableOpacity>
              );
            })}
          </View>
        </View>
      );
    }
  }

const styles = StyleSheet.create({
```

	Actually before even creating the component, we can write code as if we have already created the component and it does what we expect it to do! Later we can write code for the component and make it behave exactly as we want it to. Teacher explains that .map() passes both item and index for each item in the array to the callback function.	The student writes the code.


Monkey Chunky Stage 3: Teacher Reference
Last saved less than 5 seconds ago. See previous saves.
An update is available. Click here to reload and start using the new version. ×
rch
Run
Open files
App.js
localdb.js
Project
assets
components
App.js
localdb.js
package.json
README.md

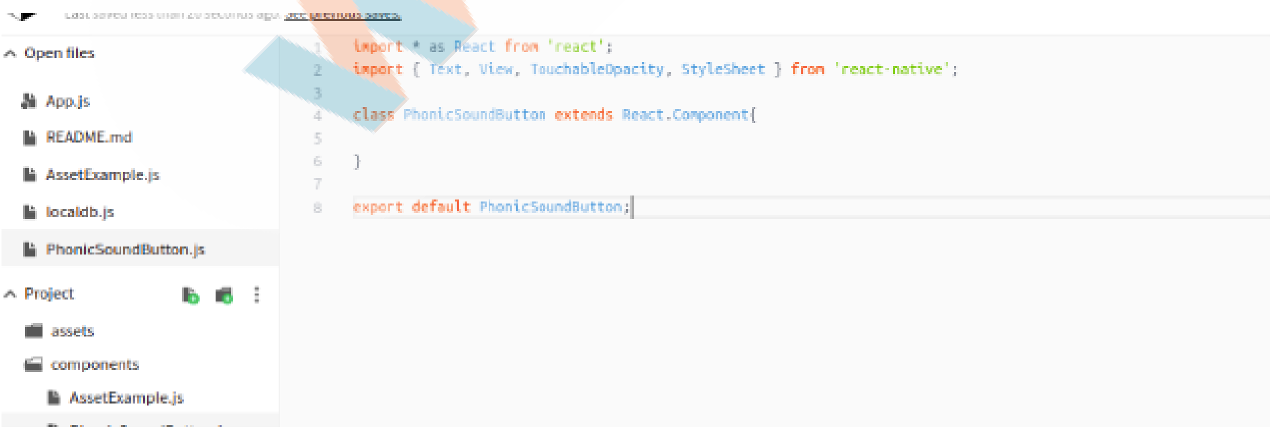
```
          />

          <TextInput
            style={styles.inputBox}
            onChangeText={text => {
              this.setState({ text: text });
            }}
            value={this.state.text}
          />
          <TouchableOpacity
            style={styles.goButton}
            onPress={() => {
              this.setState({ chunks: db[this.state.text].chunks });
              this.setState({ phonicSounds: db[this.state.text].phones });
            }}>
            <Text style={styles.buttonText}>GO</Text>
          </TouchableOpacity>
          <View>
            {this.state.chunks.map((item, index) => {
              return (
                <PhonicSoundButton wordChunk = {this.state.chunks[index]} soundChunk = {this.state.phonicSounds
[index]}/>
              );
            })}
          </View>
        </View>
      );
    }
  }
```

You can see PhonicSoundButton is underlined with red because the computer does not know what it is.

Let's now define the 'PhonicSoundButton'.

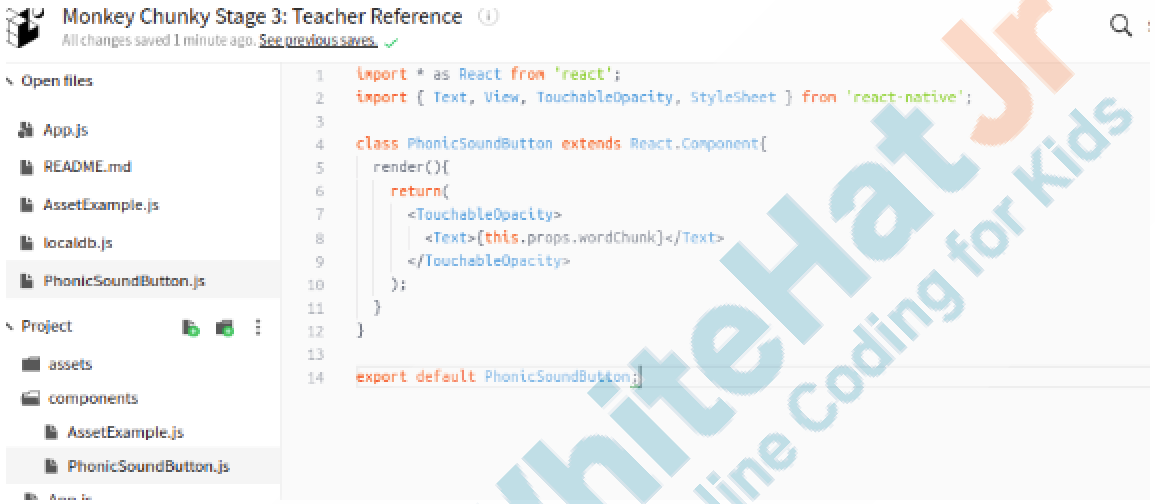
Can you quickly do that?

Student creates a new file called 'PhonicSoundButton.js' and starts with a template code for creating a new Component class.



```
import * as React from 'react';
import { Text, View, TouchableOpacity, StyleSheet } from 'react-native';

class PhonicSoundButton extends React.Component{

}

export default PhonicSoundButton;
```

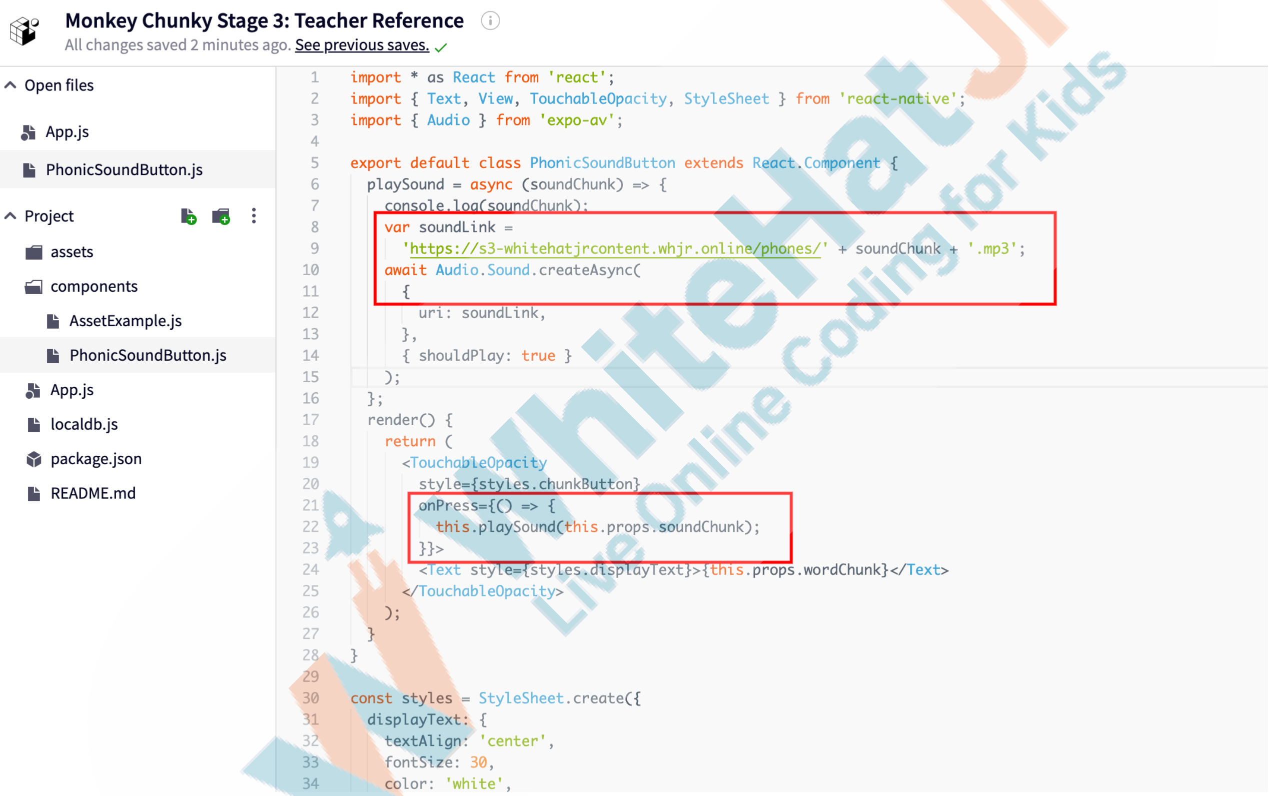
	Let's write a render() method for this component. We want our word chunk to contain a text and act like a button. We can use a ‘TouchableOpacity’ Component with a Text inside it containing the text prop passed from the App Component.	The student writes the code.



```
import * as React from 'react';
import { Text, View, TouchableOpacity, StyleSheet } from 'react-native';

class PhonicSoundButton extends React.Component{
  render(){
    return(
      <TouchableOpacity>
        <Text>{this.props.wordChunk}</Text>
      </TouchableOpacity>
    );
  }
}

export default PhonicSoundButton;
```

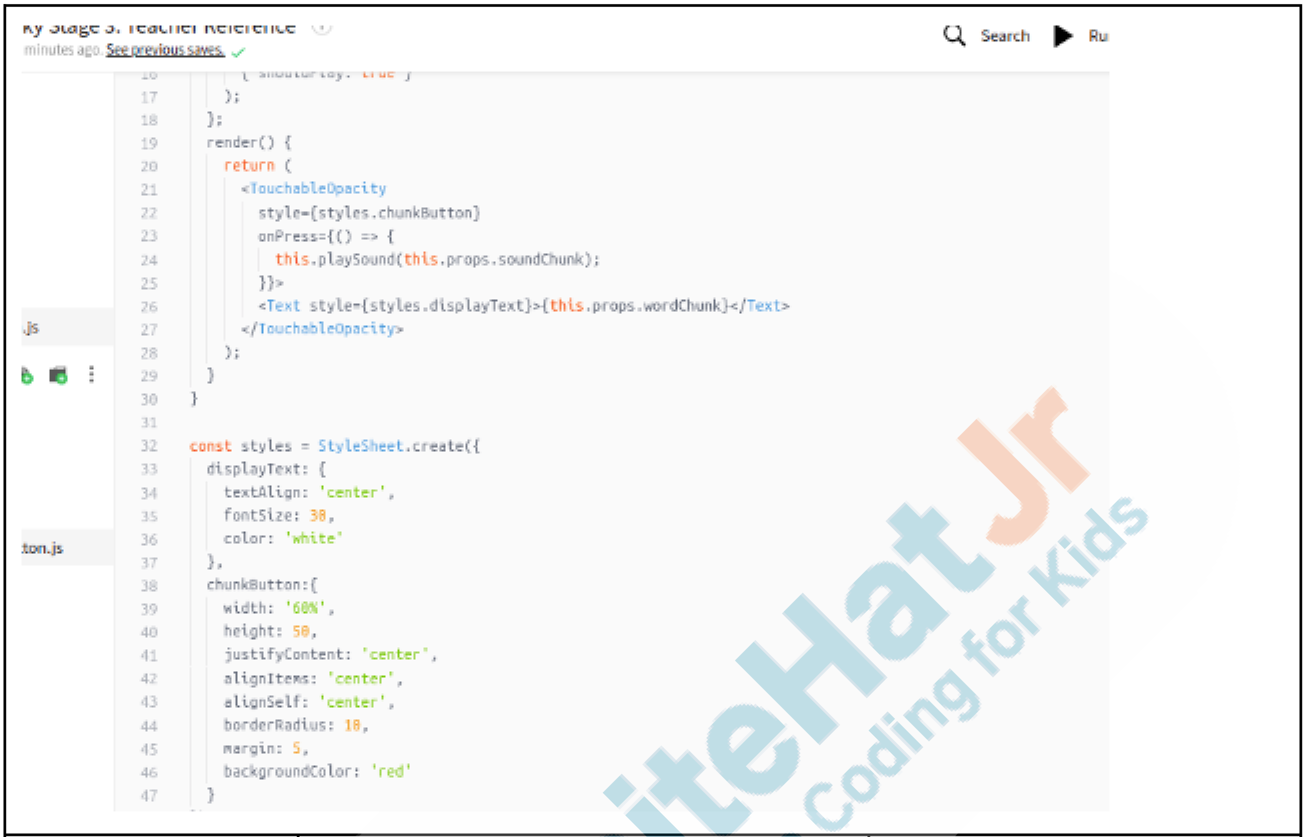
	Now, on pressing the ‘TouchableOpacity’, the sound corresponding to the phonic sounds should be played. Look at all the sounds given in **Student Activity 2**. These are the same sounds which are stored in the phones array of the word in the database. Note: You can just change the file name at the end of the URL with the phone name to listen to the sound.	The student looks at the sounds uploaded on Amazon.

	When the button is pressed, you can concatenate the phone name with the URL to create the sound URI. Let's play the sound of the phone when the button is pressed. Remember, you will have to import Audio from 'expo-av' library.	The student writes code to play the sound of the word chunk when the button is pressed. Student can refer to the Quiz Buzzer App written earlier.



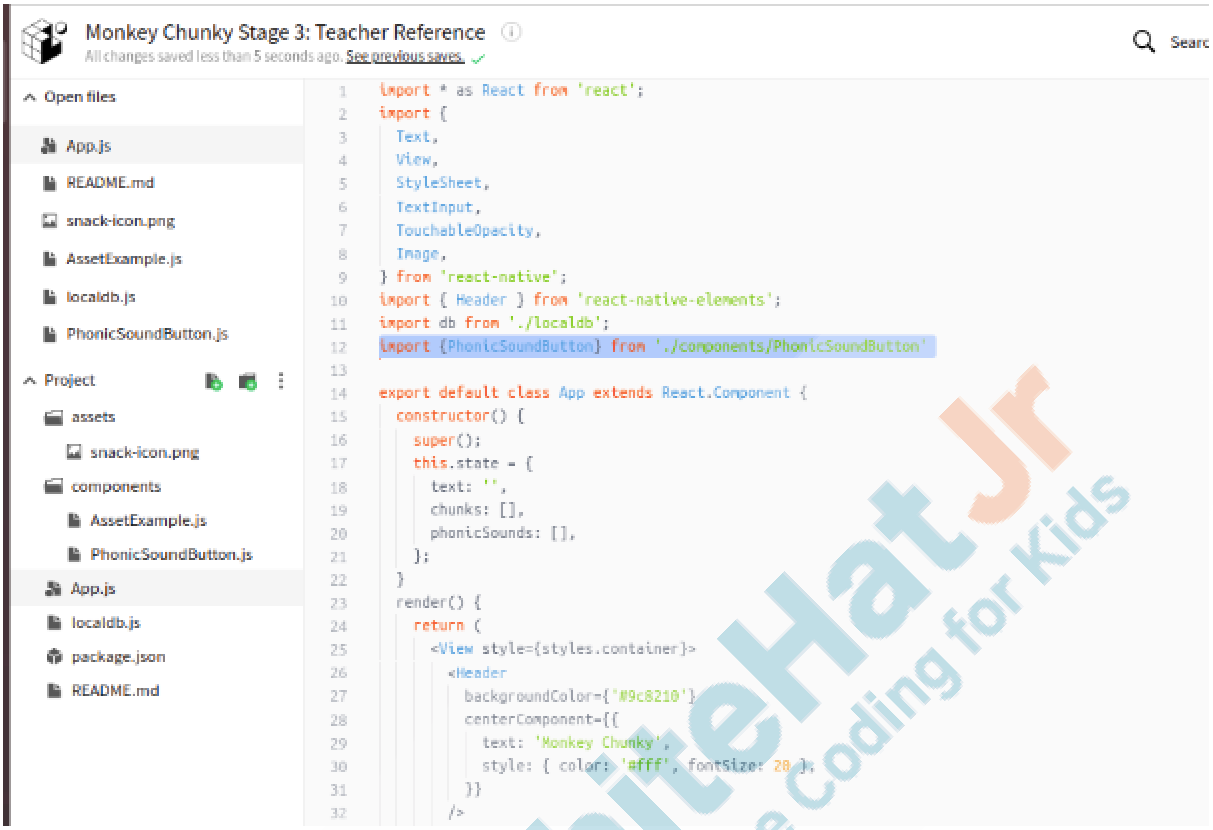
```
import * as React from 'react';
import { Text, View, TouchableOpacity, StyleSheet } from 'react-native';
import { Audio } from 'expo-av';

export default class PhonicSoundButton extends React.Component {
  playSound = async (soundChunk) => {
    console.log(soundChunk);
    var soundLink =
      'https://s3-whitehatjrcontent.whjr.online/phones/' + soundChunk + '.mp3';
    await Audio.Sound.createAsync(
      {
        uri: soundLink,
      },
      { shouldPlay: true }
    );
  };
  render() {
    return (
      <TouchableOpacity
        style={styles.chunkButton}
        onPress={() => {
          this.playSound(this.props.soundChunk);
        }}>
        <Text style={styles.displayText}>{this.props.wordChunk}</Text>
      </TouchableOpacity>
    );
  }
}

const styles = StyleSheet.create({
  displayText: {
    textAlign: 'center',
    fontSize: 30,
    color: 'white',
```

	Let's add proper styling to our ‘TouchableOpacity’ and Text we have used here.	The student adds styling using StyleSheet.


ky Stage 3: Teacher Reference
minutes ago. See previous saves.
Search
Ru
.js
ton.js

```
      { shouldPlay: true }
    );
  };
  render() {
    return (
      <TouchableOpacity
        style={styles.chunkButton}
        onPress={() => {
          this.playSound(this.props.soundChunk);
        }}>
        <Text style={styles.displayText}>{this.props.wordChunk}</Text>
      </TouchableOpacity>
    );
  }
}

const styles = StyleSheet.create({
  displayText: {
    textAlign: 'center',
    fontSize: 30,
    color: 'white'
  },
  chunkButton:{
    width: '60%',
    height: 50,
    justifyContent: 'center',
    alignItems: 'center',
    alignSelf: 'center',
    borderRadius: 10,
    margin: 5,
    backgroundColor: 'red'
  }
```

	Great! We have the ‘phonicSound’ component ready now. You can import this in your App.js file.	The student imports ‘PhonicSoundButton’ in App.js.



```
import * as React from 'react';
import {
  Text,
  View,
  StyleSheet,
  TextInput,
  TouchableOpacity,
  Image,
} from 'react-native';
import { Header } from 'react-native-elements';
import db from './localdb';
import {PhonicSoundButton} from './components/PhonicSoundButton'

export default class App extends React.Component {
  constructor() {
    super();
    this.state = {
      text: '',
      chunks: [],
      phonicSounds: [],
    };
  }
  render() {
    return (
      <View style={styles.container}>
        <Header
          backgroundColor={'#9c8210'}
          centerComponent={{
            text: 'Monkey Chunky',
            style: { color: '#fff', fontSize: 20 },
          }}
        />
```

	Now you can run and test your app to see if it works.	The student runs and tests the app on their phone using the expo app.

Teacher Guides Student to Stop Screen Share

FEEDBACK

- **Encourage the student to review React Native using the reference material provided in the links.**

Step 4: **Wrap-Up** **(5 min)**	The monkey-chunky app is working and ready to be published. But wait, we have only a few words in our app. Also, when a user presses a word which is not there in the database, the	The student checks for these errors.

	program throws an error - even if the word is in capital letters.	
	In the next class, we will learn how we can add more words and fix these small issues in our app. Also, we will be learning more about git and how we can contribute to Open Source Applications using git. It is going to be an exciting session!	
	You get a “hats off”. Till next class then. See you. Bye!	Make sure you have given at least 2 Hats Off during the class for: Creatively Solved Activities +10 Great Question +10 Strong Concentration +10
Project Pointers and Cues (5 min)	TEXT TO SPEECH CONVERTER **Goal of the Project:** Today you learned to play the corresponding sound of a phonic chunk. In this text to speech converter app you have to use expo speech library to convert a text entered by the user to a pronounced word.	

	Story: Saisha's friend Lisa is visiting her from France. She understands English, but speaks only French. Help Saisha create a Text-to-Speech app for Lisa so she can communicate with people in India. I am very excited to see your project solution and I know you both will do really well. Bye Bye!	
	Teacher Clicks ✖ End Class	
Additional Activities	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	Teacher Reference	https://snack.expo.io/@rajeevtfi/monkey-chunky-stage-3:-teacher-reference
Student Activity 1	Class Activity	https://snack.expo.io/@rajeevtfi/monkeychunky-stage2--reference
Student Activity 2	Phone sounds link	https://s3-whitehatjrcontent.whjr.online/phones/V.mp3
Student Activity 3	Student Reference	https://facebook.github.io/react-native/docs/tutorial