PRO-C66

Ternary Operators

What is our GOAL for this MODULE?

We learned about ternary operators and how to use them in a program in place of conditional statements.

What did we ACHIEVE in the class TODAY?

- Learned about ternary operators.
- Used ternary operators to conditionally render different styles to the user.
- Fixed case issue in the application.

Which CONCEPTS/ CODING BLOCKS did we cover today?

- Ternary Operators

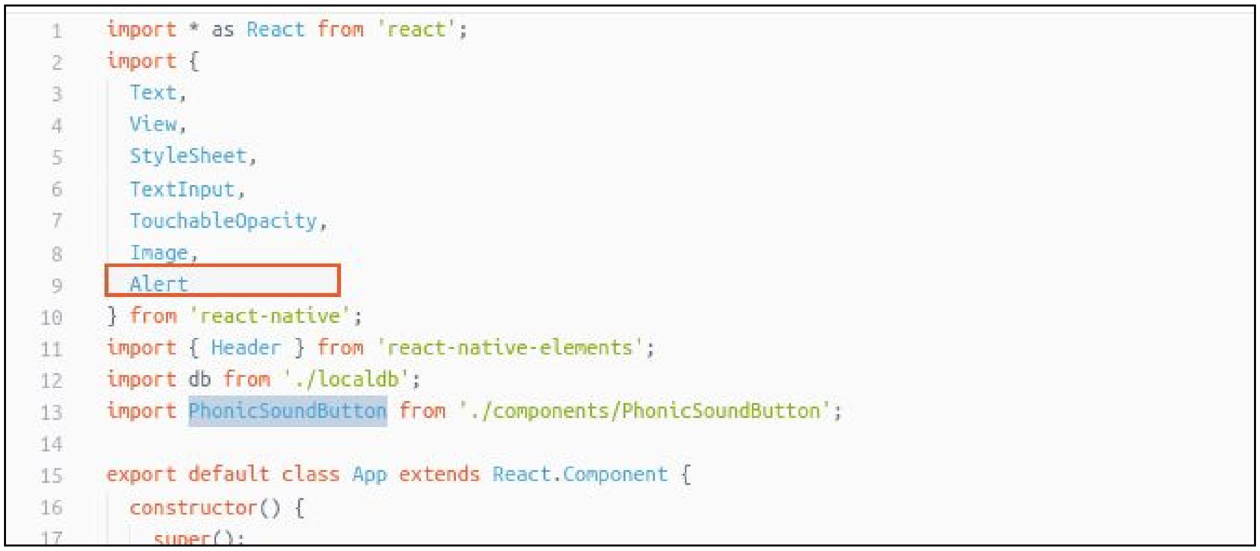
How did we DO the activities?

Ternary operator is a simple operator in programming using which you can write if-else statements in a single line. Programmers use ternary operators to write conditional statements in their code keeping their program small.

1. Let's use the ternary operator to create an alert box when the typed word does not exist in the database.

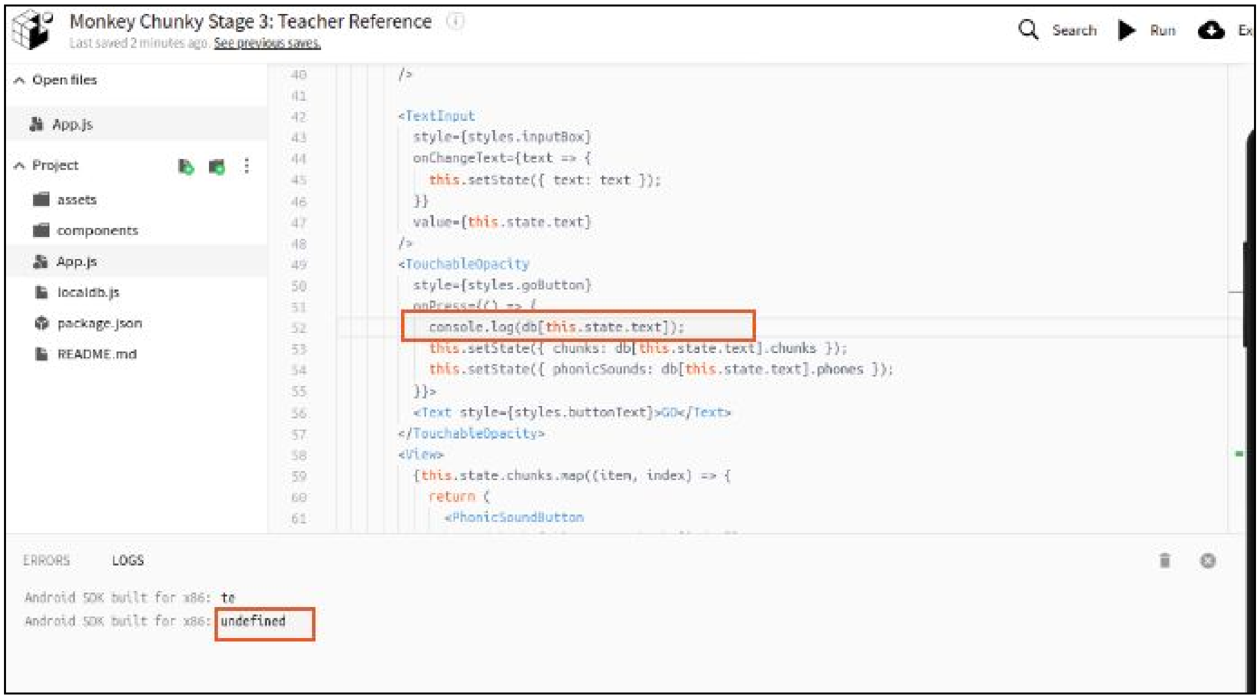
2. When the 'Go' button is pressed, we should check if the word entered in the text is in the database or not.
    - If the word is there in the database, we store the chunks and the phones in the state using 'this.state', else we display an alert Box.

3. React Native also has an Alert component which can be used to display alert. Let's import it.



```
import * as React from 'react';
import {
  Text,
  View,
  StyleSheet,
  TextInput,
  TouchableOpacity,
  Image,
  Alert
} 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();
```

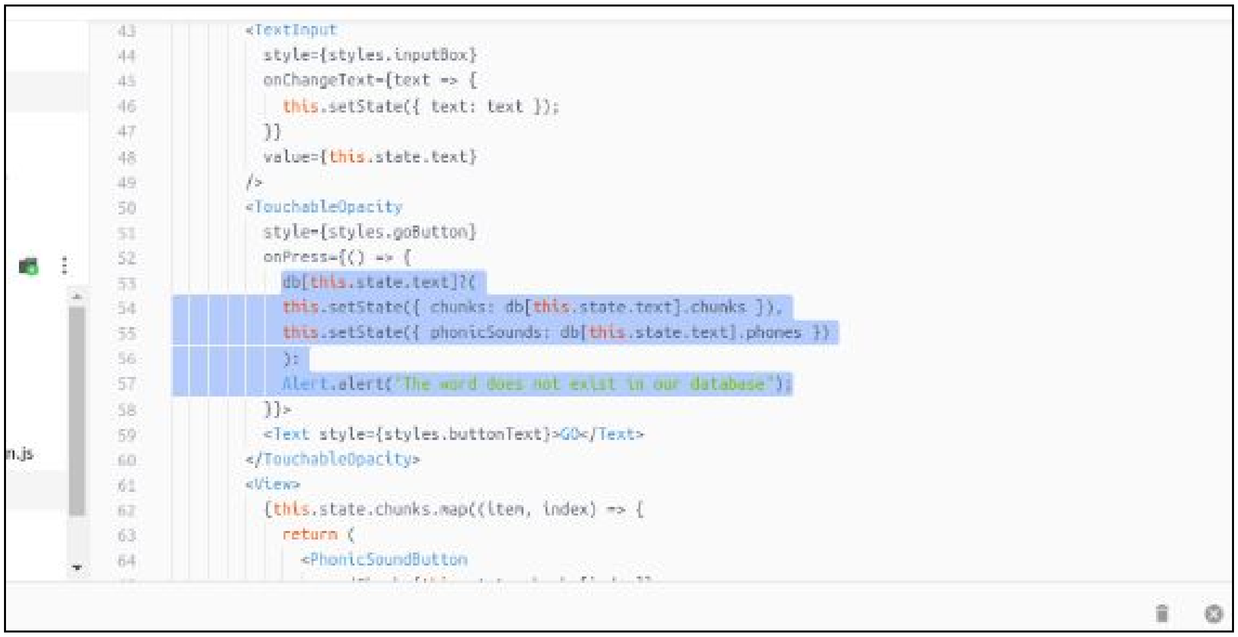
4. Enter a word which does not exist in our database and just console what we get if we try to access it from our database.



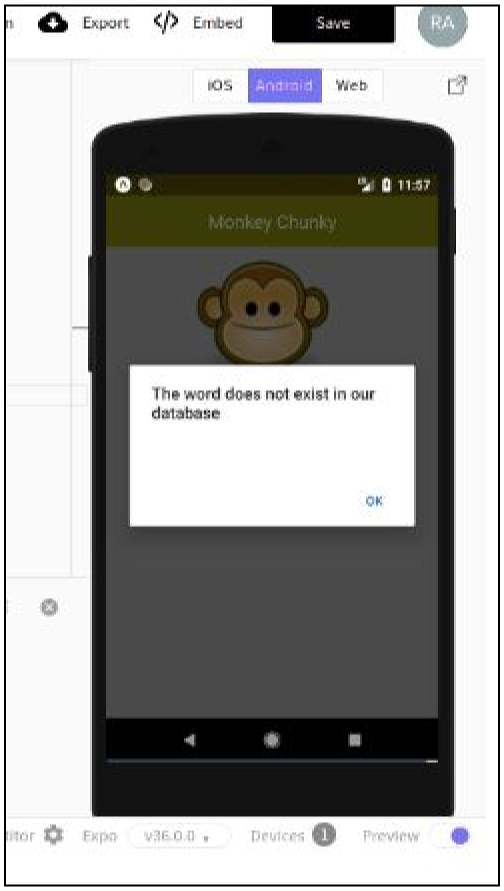
```
        />

        <TextInput
          style={styles.inputBox}
          onChangeText={text => {
            this.setState({ text: text });
          }}
          value={this.state.text}
        />
        <TouchableOpacity
          style={styles.goButton}
          onPress={() => {
            console.log(db[this.state.text]);
            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
```

```
Android SDK built for x86: te
Android SDK built for x86: undefined
```

5. We can use this as our condition for our ternary operator.
    - If the word exists, the state for chunks and ‘phonicSounds’ are set. Else, an alert box is displayed.



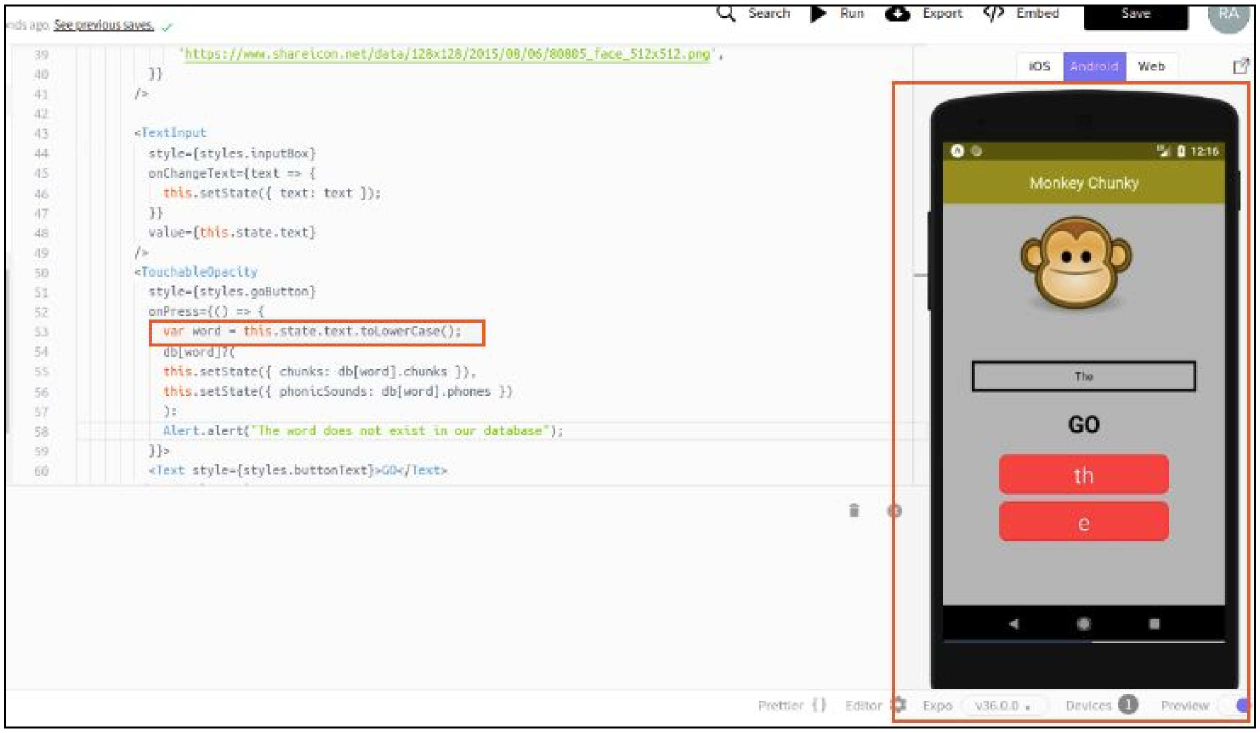
```
        <TextInput
          style={styles.inputBox}
          onChangeText={text => {
            this.setState({ text: text });
          }}
          value={this.state.text}
        />
        <TouchableOpacity
          style={styles.goButton}
          onPress={() => {
            db[this.state.text]?(
            this.setState({ chunks: db[this.state.text].chunks }),
            this.setState({ phonicSounds: db[this.state.text].phones })
            ):
            Alert.alert('The word does not exist in our database');
          }}>
          <Text style={styles.buttonText}>GO</Text>
        </TouchableOpacity>
        <View>
          {this.state.chunks.map((item, index) => {
            return (
              <PhonicSoundButton
```



The word does not exist in our database

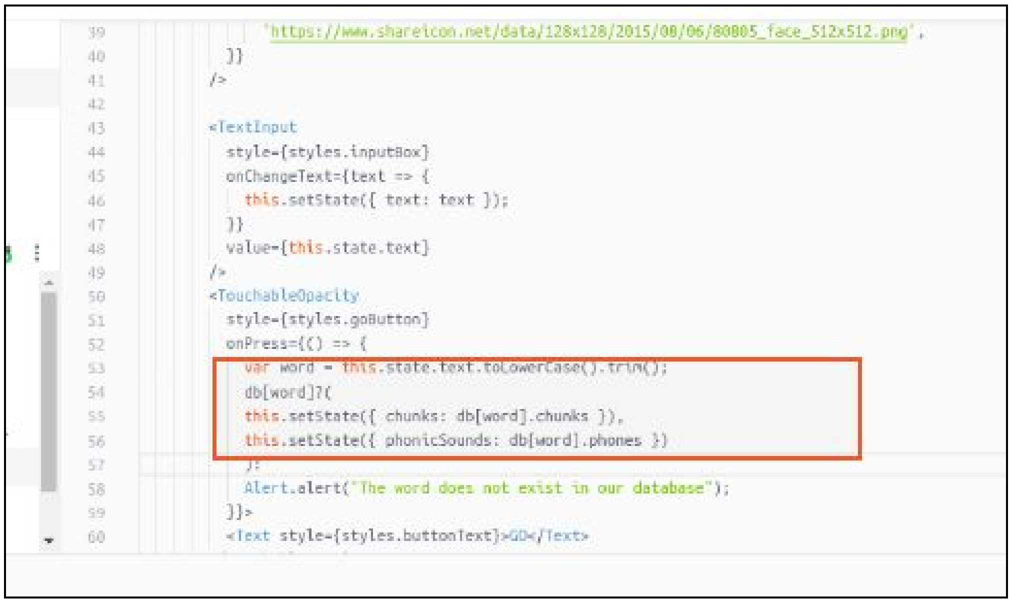
6. Our database contains all the letters in lowercase. So we can convert the text from our 'TextInput' to lowercase before searching for the word in our database.

7. Also strings already have an in-built function called '.toLowerCase()'. It converts any string to lowercase. You can use it to convert the text to lowercase.



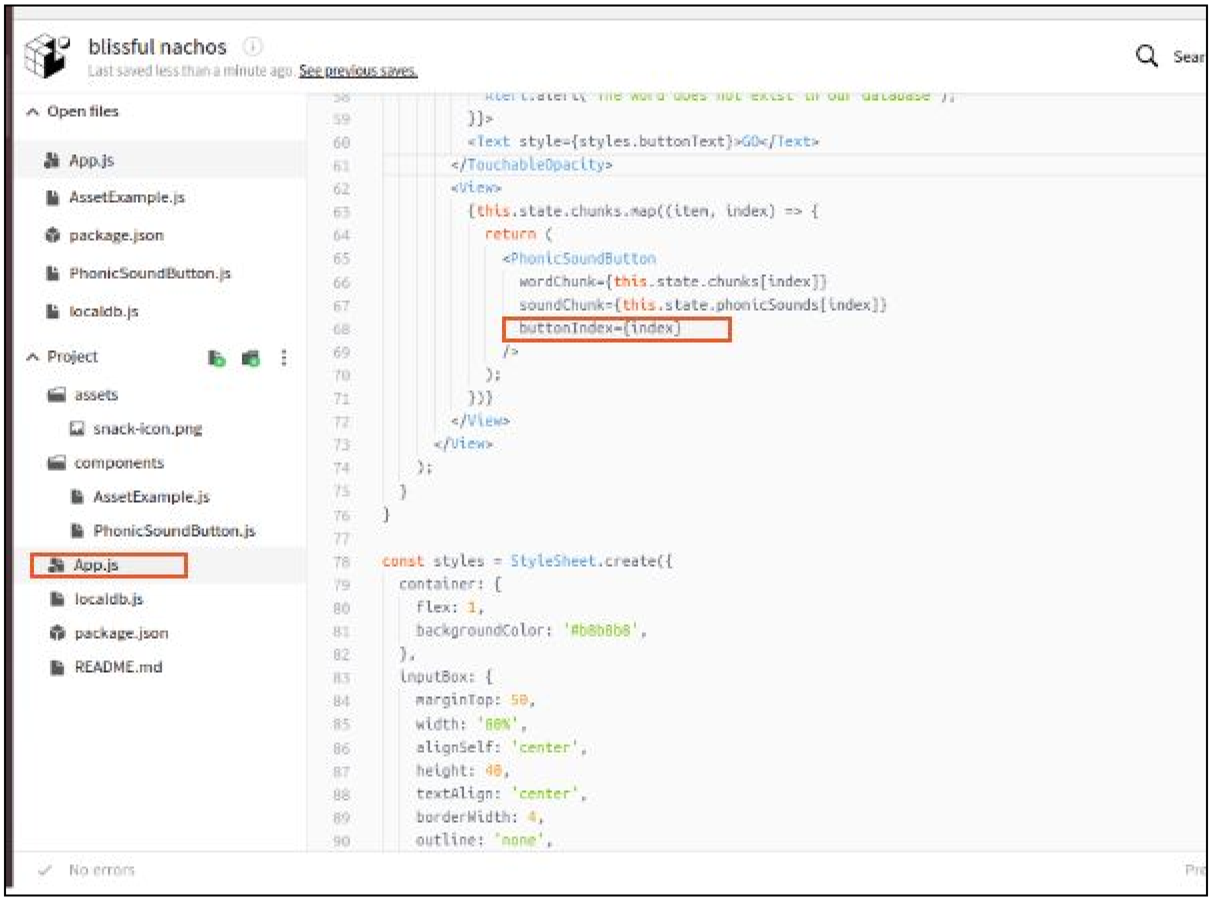
```
          'https://www.shareicon.net/data/128x128/2015/08/06/80805_face_512x512.png',
      }}
    />

    <TextInput
      style={styles.inputBox}
      onChangeText={text => {
        this.setState({ text: text });
      }}
      value={this.state.text}
    />
    <TouchableOpacity
      style={styles.goButton}
      onPress={() => {
        var word = this.state.text.toLowerCase();
        db[word]?(
        this.setState({ chunks: db[word].chunks }),
        this.setState({ phonicSounds: db[word].phones })
        ):
        Alert.alert("The word does not exist in our database");
      }}>
      <Text style={styles.buttonText}>GO</Text>
```

8. We can do more.
    - We can remove the spaces before and after the typed word by the user so that if the user has accidentally pressed spaces before or after the word, we can still find the word in the database.
    - String has a function called 'trim()' which does exactly that.
    - 'toLowerCase()' returns a string so you can combine '.trim()' in the same statement to strip the word of any spaces.

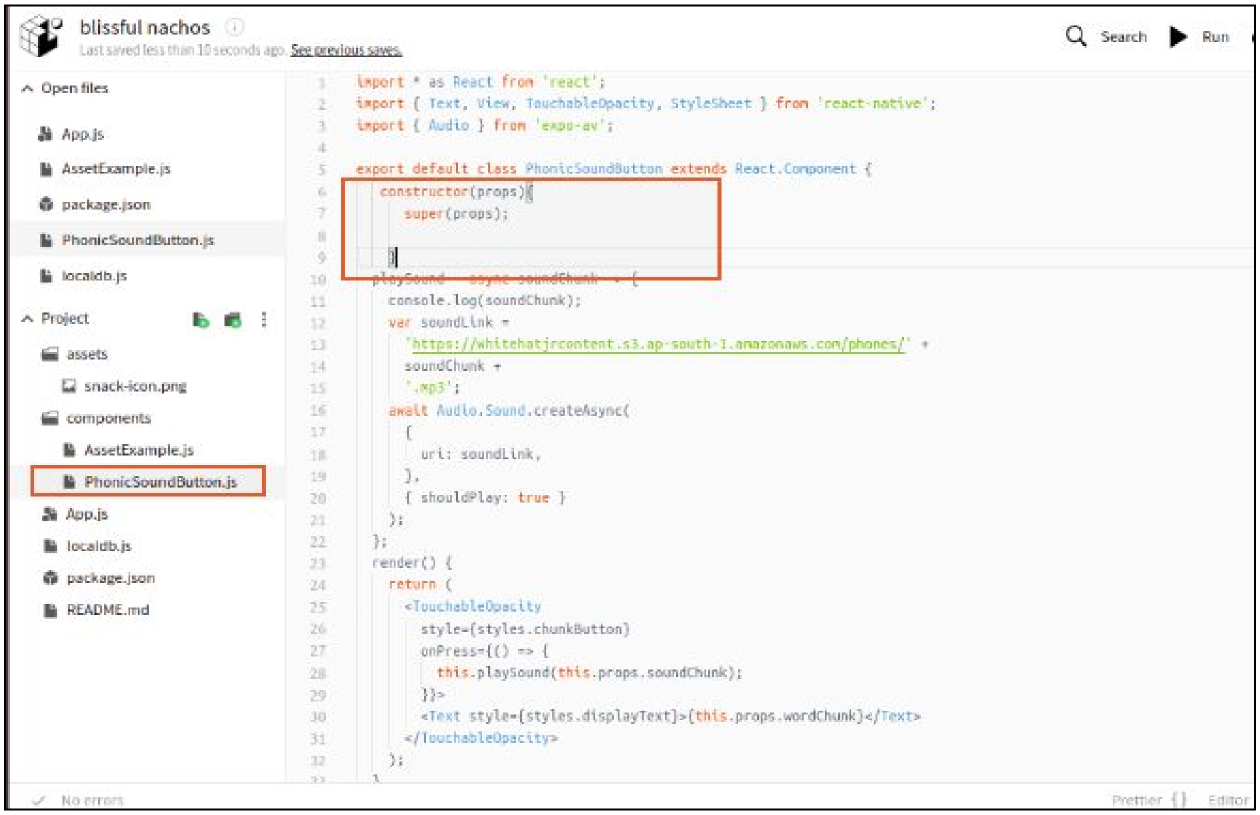


```
          'https://www.shareicon.net/data/128x128/2015/08/06/80805_face_512x512.png',
      }}
    />

    <TextInput
      style={styles.inputBox}
      onChangeText={text => {
        this.setState({ text: text });
      }}
      value={this.state.text}
    />
    <TouchableOpacity
      style={styles.goButton}
      onPress={() => {
        var word = this.state.text.toLowerCase().trim();
        db[word]?(
        this.setState({ chunks: db[word].chunks }),
        this.setState({ phonicSounds: db[word].phones })
        ):
        Alert.alert("The word does not exist in our database");
      }}>
      <Text style={styles.buttonText}>GO</Text>
```

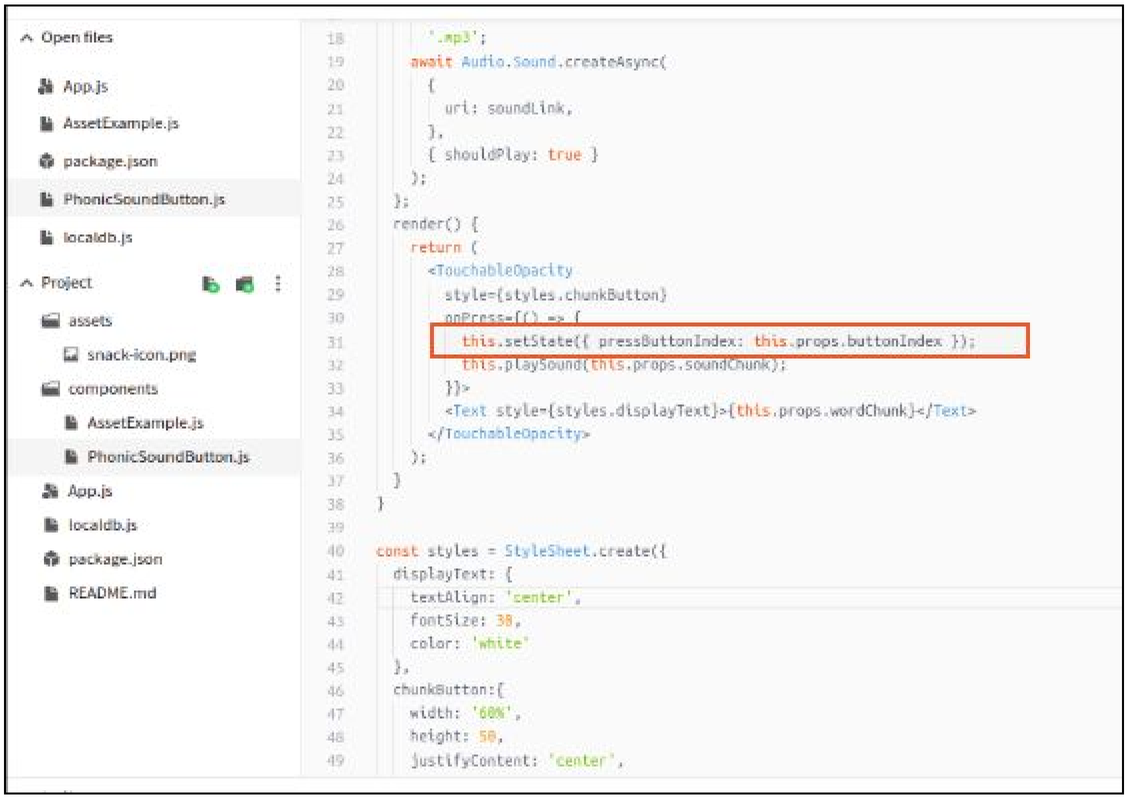
9. Now let's write code to change the style of the last phonic button which was pressed.
    - We can create a new prop called 'buttonIndex' for 'phonicSoundButton' which passes the index of the current button.
    - Inside the 'PhonicSoundButton' component, we can have a state called 'pressedButtonIndex'.
    - This will contain the index number of the word chunk button in the array which is currently pressed.
    - If the 'pressedButtonIndex' is the same as the word chunk index, we will give it one kind of style, else we will give a different style.



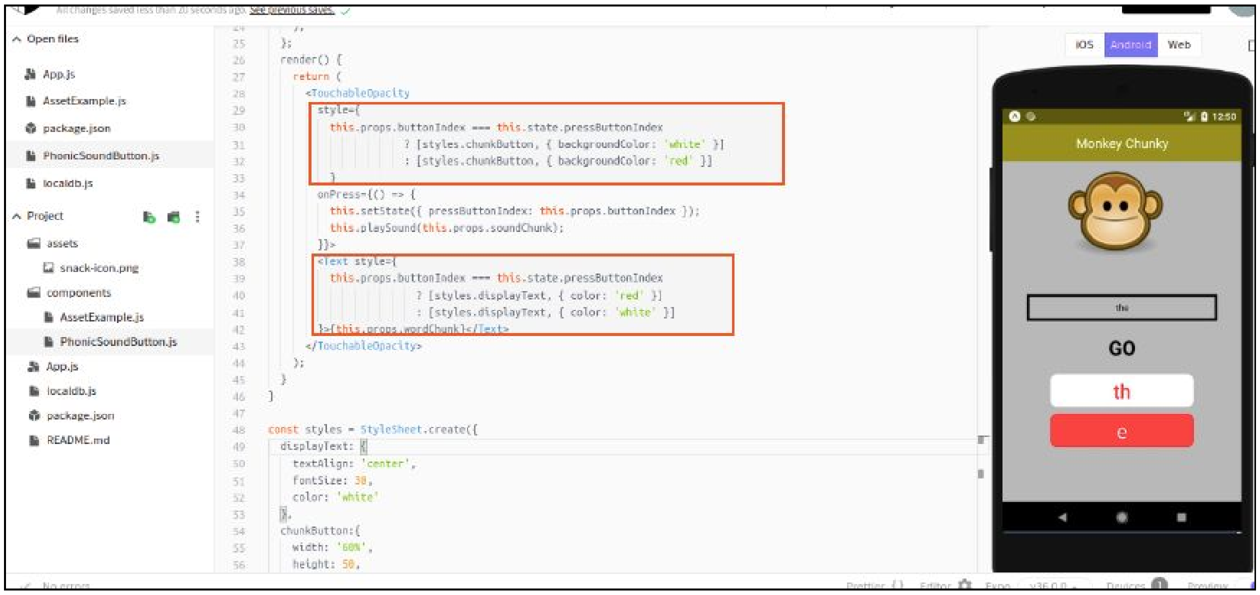
```
          Alert.alert('The word does not exist in our database');
        }}>
        <Text style={styles.buttonText}>GO</Text>
      </TouchableOpacity>
      <View>
        {this.state.chunks.map((item, index) => {
          return (
            <PhonicSoundButton
              wordChunk={this.state.chunks[index]}
              soundChunk={this.state.phonicSounds[index]}
              buttonIndex={index}
            />
          );
        })}
      </View>
    </View>
  );
  }
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#b8b8b8',
  },
  inputBox: {
    marginTop: 50,
    width: '80%',
    alignSelf: 'center',
    height: 40,
    textAlign: 'center',
    borderWidth: 4,
    outline: 'none',
```



```
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 {
  constructor(props){
    super(props);

  }
  playSound = async soundChunk => {
    console.log(soundChunk);
    var soundLink =
      'https://whitehatjrcontent.s3.ap-south-1.amazonaws.com/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>
    );
  }
```



```
      '.mp3';
    await Audio.Sound.createAsync(
      {
        uri: soundLink,
      },
      { shouldPlay: true }
    );
  };
  render() {
    return (
      <TouchableOpacity
        style={styles.chunkButton}
        onPress={() => {
          this.setState({ pressButtonIndex: this.props.buttonIndex });
          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',
```



```
  };
  render() {
    return (
      <TouchableOpacity
        style={
          this.props.buttonIndex === this.state.pressButtonIndex
                  ? [styles.chunkButton, { backgroundColor: 'white' }]
                  : [styles.chunkButton, { backgroundColor: 'red' }]
          }
        onPress={() => {
          this.setState({ pressButtonIndex: this.props.buttonIndex });
          this.playSound(this.props.soundChunk);
        }}>
        <Text style={
          this.props.buttonIndex === this.state.pressButtonIndex
                    ? [styles.displayText, { color: 'red' }]
                    : [styles.displayText, { color: 'white' }]
        }>{this.props.wordChunk}</Text>
      </TouchableOpacity>
    );
  }
}

const styles = StyleSheet.create({
  displayText: {
    textAlign: 'center',
    fontSize: 30,
    color: 'white'
  },
  chunkButton:{
    width: '60%',
    height: 50,
```

What's NEXT?

In the next class, we will be learning about practical implementation of advanced git commands.

EXTEND YOUR KNOWLEDGE

1. Ternary operators:
   https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Conditional_Operator