Topic	Accessing data from local files	
Class Description	Students learn how to access data from local files. Students will use the data from a local file to break a given word into smaller chunks associated with a phoneme sound.	
Class	C64	
Class time	45 mins	
Goal	- Export and import data from a local file. - Get smaller chunks of a word and display it using map method.	
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

WARM-UP SESSION - 5 mins

CONTEXT

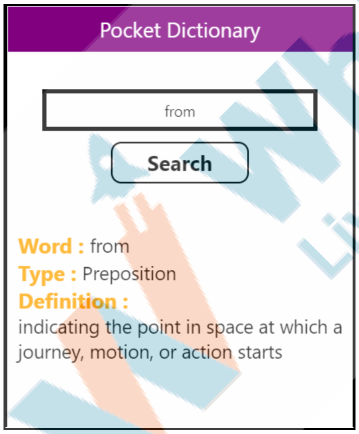
- **Review code from the last class.**
- **Introduce the problem of accessing data from local files.**

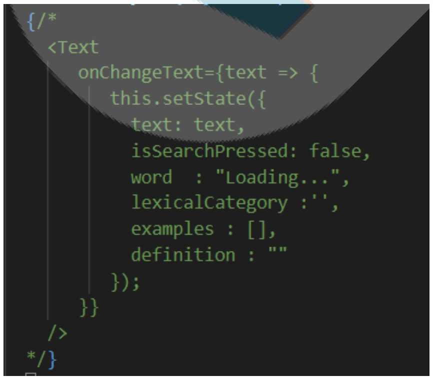
Teacher starts slideshow from slides 1 to 15

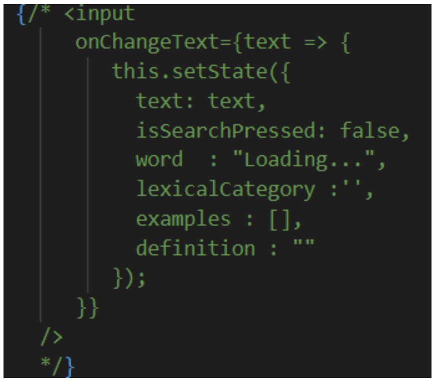
<table>
<tr><td colspan="2">Refer to speaker notes and follow the instructions on each slide.</td></tr>
<tr><th>Activity details</th><th>Solution/Guidelines</th></tr>
<tr><td>Hi, how have you been? Are you excited to learn something new?

<b><i>Run the presentation from slide 1 to slide 5.</i></b>

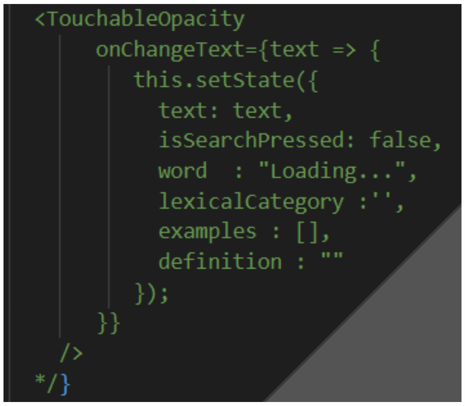
<b>The following are the warm-up session deliverables:</b>
● Reconnect with previous class topics.
● Warm-Up quiz session.</td><td><b>ESR</b>: Varied Response.

Click on the slide show tab and present the slides.</td></tr>
<tr><td colspan="2"><b>QnA Session</b></td></tr>
<tr><th>Question</th><th>Answer</th></tr>
<tr><td>Which of the following will create an input box in which the word to be searched can be typed as shown below.

Pocket Dictionary
from
Search
Word : from
Type : Preposition
Definition :
indicating the point in space at which a journey, motion, or action starts

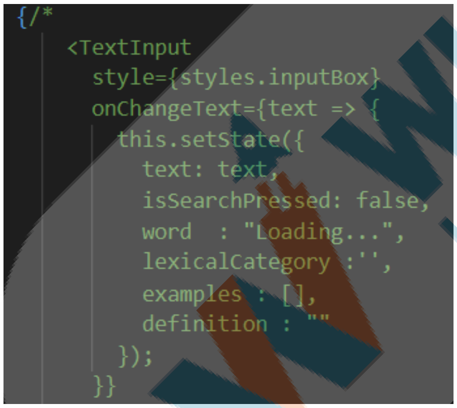
A. 
<pre>
{/*
  <Text
    onChangeText={text => {
      this.setState({
        text: text,
        isSearchPressed: false,
        word : "Loading...",
        lexicalCategory :'',
        examples : [],
        definition : ""
      });
    }}
  /&gt;
*/}
</pre></td><td>D</td></tr>
</table>

B. 

```
{/* <input
    onChangeText={text => {
      this.setState({
        text: text,
        isSearchPressed: false,
        word  : "Loading...",
        lexicalCategory :'',
        examples : [],
        definition : ""
      });
    }}
  />
  */}
```

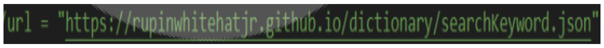
C. 

```
<TouchableOpacity
     onChangeText={text => {
       this.setState({
         text: text,
         isSearchPressed: false,
         word  : "Loading...",
         lexicalCategory :'',
         examples : [],
         definition : ""
       });
     }}
   />
 */}
```

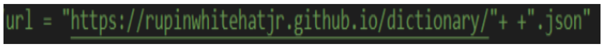
D. 

```
{/*
    <TextInput
      style={styles.inputBox}
      onChangeText={text => {
        this.setState({
          text: text,
          isSearchPressed: false,
          word  : "Loading...",
          lexicalCategory :'',
          examples : [],
          definition : ""
        });
      }}
```

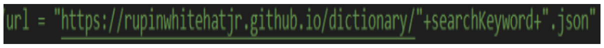
Which of the following is the correct URL to be sent to the API?

A. 

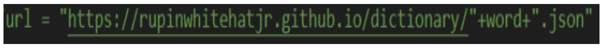
```
url = "https://rupinwhitehatjr.github.io/dictionary/searchKeyword.json"
```

B. 

```
url = "https://rupinwhitehatjr.github.io/dictionary/"+ +".json"
```

C. 

```
url = "https://rupinwhitehatjr.github.io/dictionary/"+searchKeyword+".json"
```

D. 

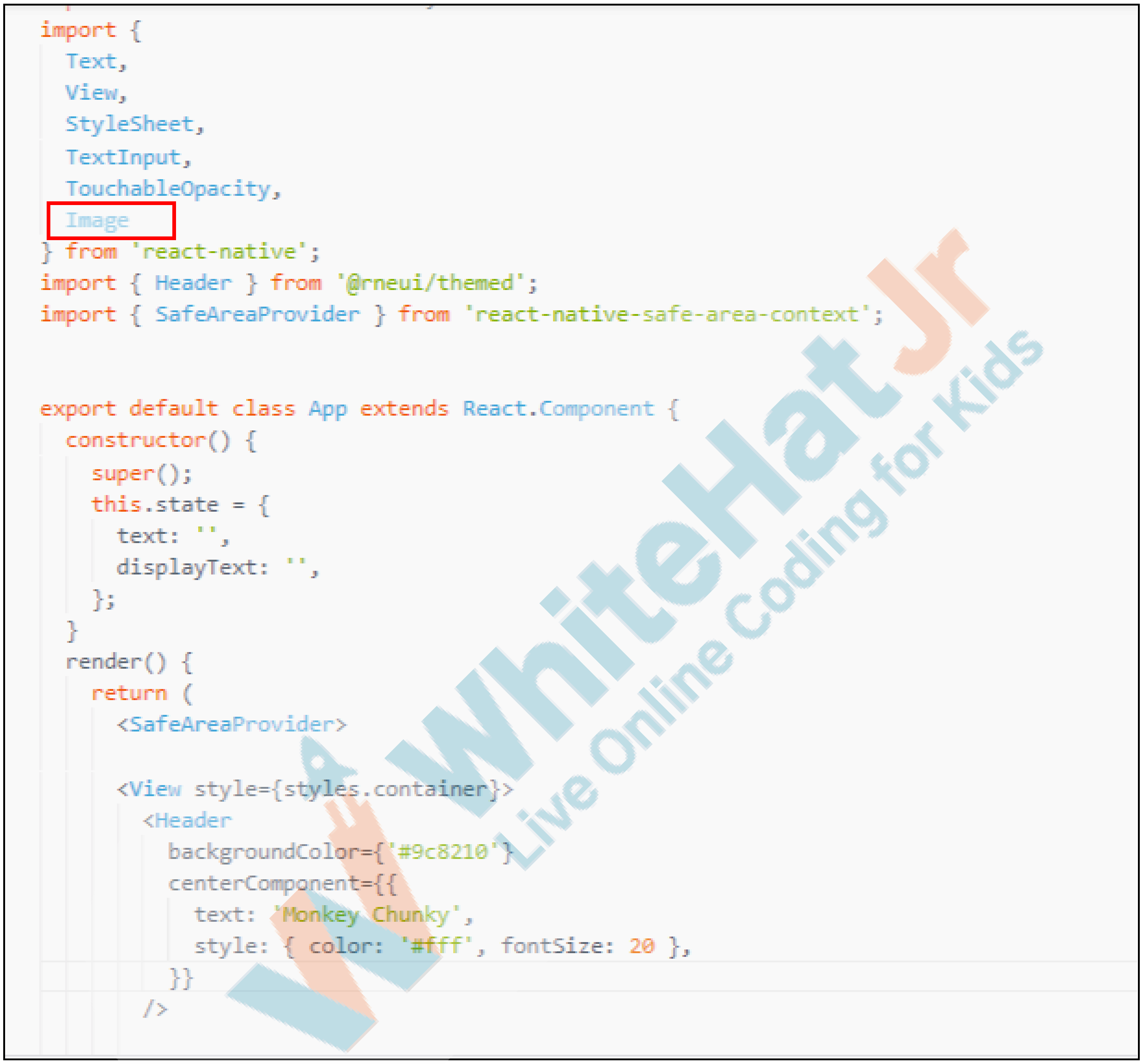
```
url = "https://rupinwhitehatjr.github.io/dictionary/"+word+".json"
```

C

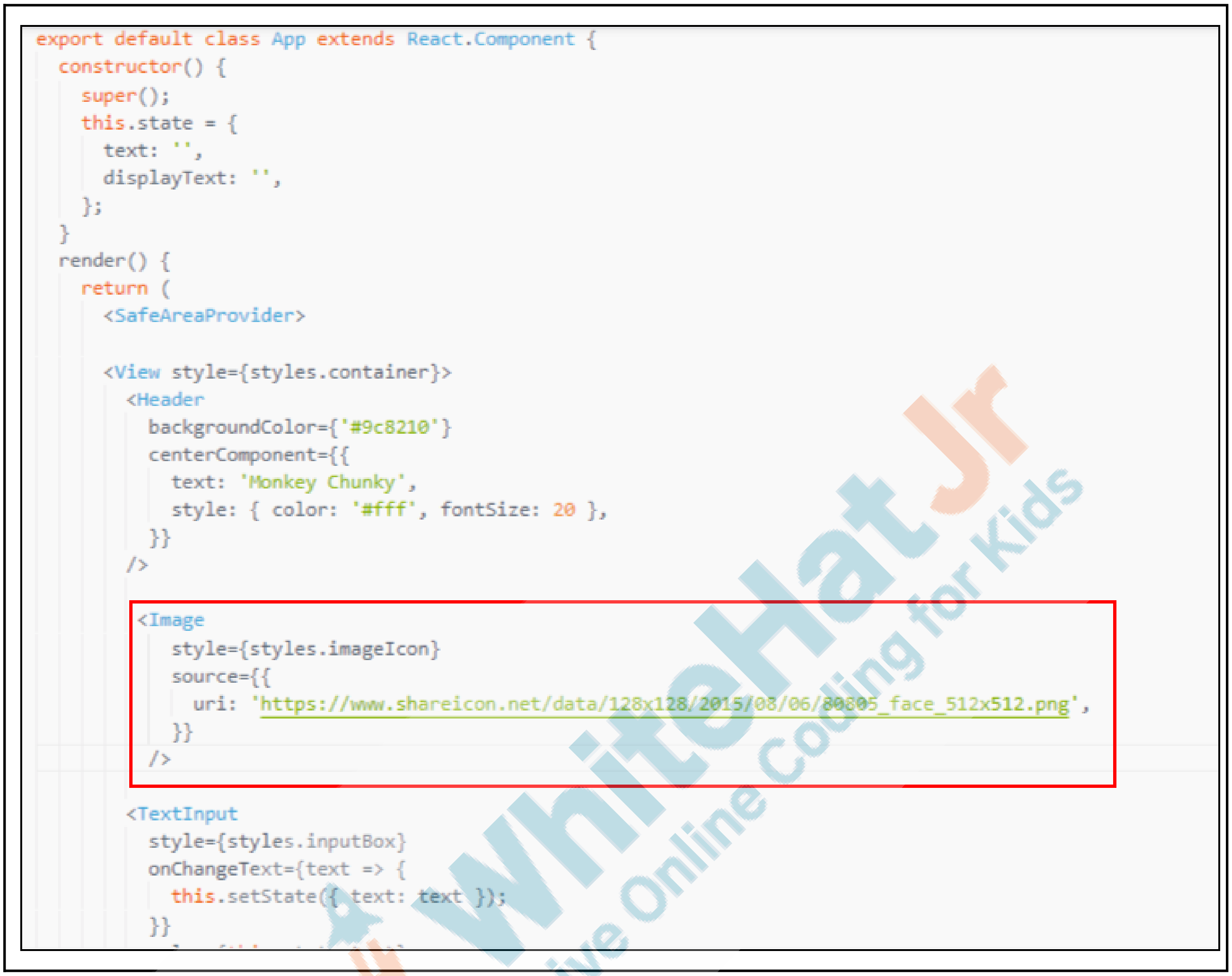
Continue the warm-up session

<table>
<tr><th colspan="2">Activity details</th><th>Solution/Guidelines</th></tr>
<tr><td colspan="2"><b><i>Run the presentation from slide 6 to slide 15 to set the problem statement.</i></b>
<b>The following are the warm-up session deliverables:</b>
● <b>Review code from the last class.</b>
● <b>Introduce the problem of accessing data from local files.</b></td><td></td></tr>
<tr><td colspan="3"><b>Teacher ends slideshow</b></td></tr>
<tr><td colspan="3"><b>TEACHER-LED ACTIVITY - 15 mins</b></td></tr>
<tr><td colspan="3"><b>Teacher Initiates Screen Share</b></td></tr>
<tr><td colspan="3"><b><u>CHALLENGE</u></b>
● <b>Export and import data from a local file.</b></td></tr>
<tr><td><b>Step 2:
Teacher-led Activity
(15 min)</b></td><td>Before we start, let's quickly open our code from the last class.
Teacher opens <b><u>Teacher Activity 1</u></b>.
Can you quickly go through the code and explain it?</td><td>The student goes through the code and explains what is happening.
He/She also explains how text input is getting collected.</td></tr>
<tr><td></td><td>Let's first add some image to our Monkey-chunky app to give it some branding.
Do you know how to add an image in React Native?</td><td>ESR:
Using 'Image' Component.</td></tr>
<tr><td></td><td>Can you guide me on how to add an image above the input box?</td><td>The student guides on how to use the image component</td></tr>
</table>

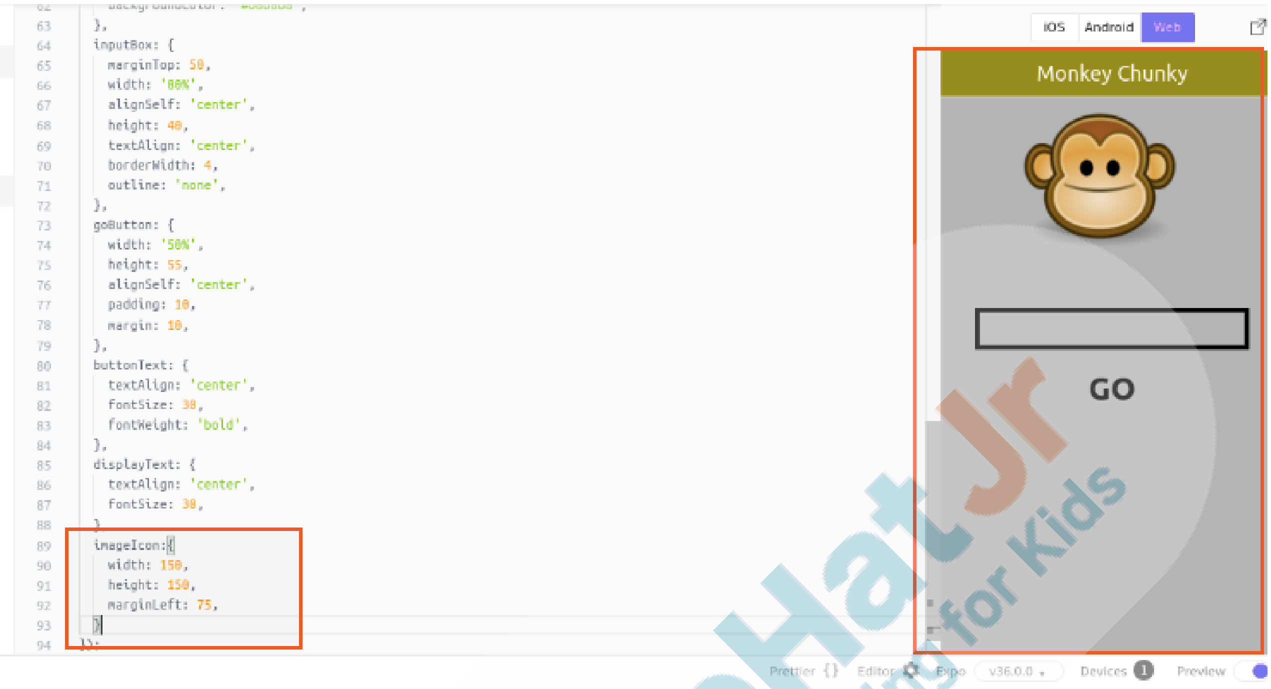
	Teacher codes to add the ‘Image’ Component.	to add a monkey image above the input box.



```
import {
  Text,
  View,
  StyleSheet,
  TextInput,
  TouchableOpacity,
  Image
} from 'react-native';
import { Header } from '@rneui/themed';
import { SafeAreaProvider } from 'react-native-safe-area-context';


export default class App extends React.Component {
  constructor() {
    super();
    this.state = {
      text: '',
      displayText: '',
    };
  }
  render() {
    return (
      <SafeAreaProvider>

      <View style={styles.container}>
        <Header
          backgroundColor={'#9c8210'}
          centerComponent={{
            text: 'Monkey Chunky',
            style: { color: '#fff', fontSize: 20 },
          }}
        />
```

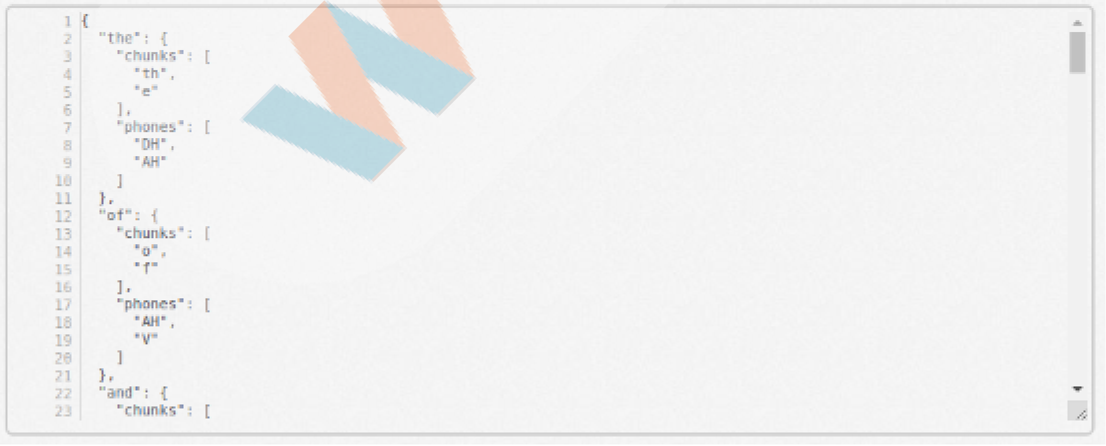


```
export default class App extends React.Component {
  constructor() {
    super();
    this.state = {
      text: '',
      displayText: '',
    };
  }
  render() {
    return (
      <SafeAreaProvider>

      <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',
          }}
        />

        <TextInput
          style={styles.inputBox}
          onChangeText={text => {
            this.setState({ text: text });
          }}
```



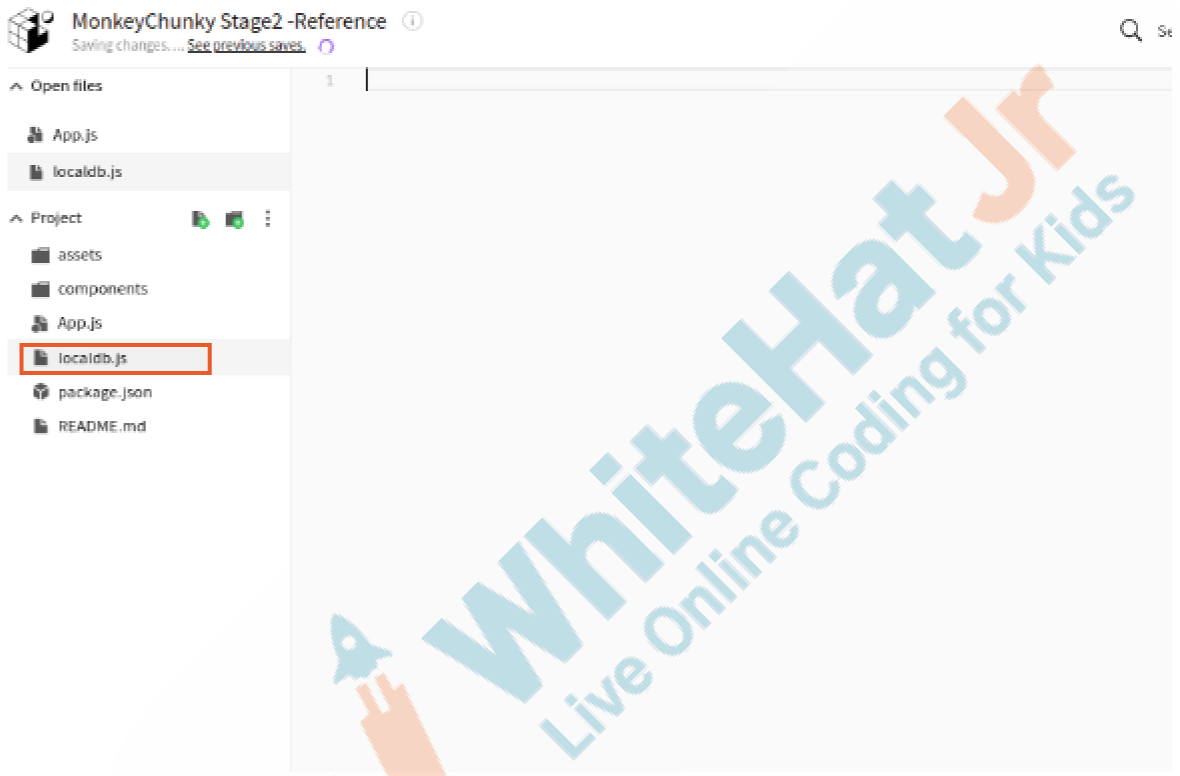
```
  },
  inputBox: {
    marginTop: 50,
    width: '80%',
    alignSelf: 'center',
    height: 40,
    textAlign: 'center',
    borderWidth: 4,
    outline: 'none',
  },
  goButton: {
    width: '50%',
    height: 55,
    alignSelf: 'center',
    padding: 10,
    margin: 10,
  },
  buttonText: {
    textAlign: 'center',
    fontSize: 30,
    fontWeight: 'bold',
  },
  displayText: {
    textAlign: 'center',
    fontSize: 30,
  }
  imageIcon:{
    width: 150,
    height: 150,
    marginLeft: 75,
  }
});
```

	How does the app look now?	ESR: varied
	Awesome! Right now our app has an input box where we type text and it displays the same word below. Instead of the same word, we need to chunk the words. To do that we need some sort of database where the chunks of each word is stored. What kind of database do you already know?	The student listens and asks questions' ESR: Firebase Realtime Database.

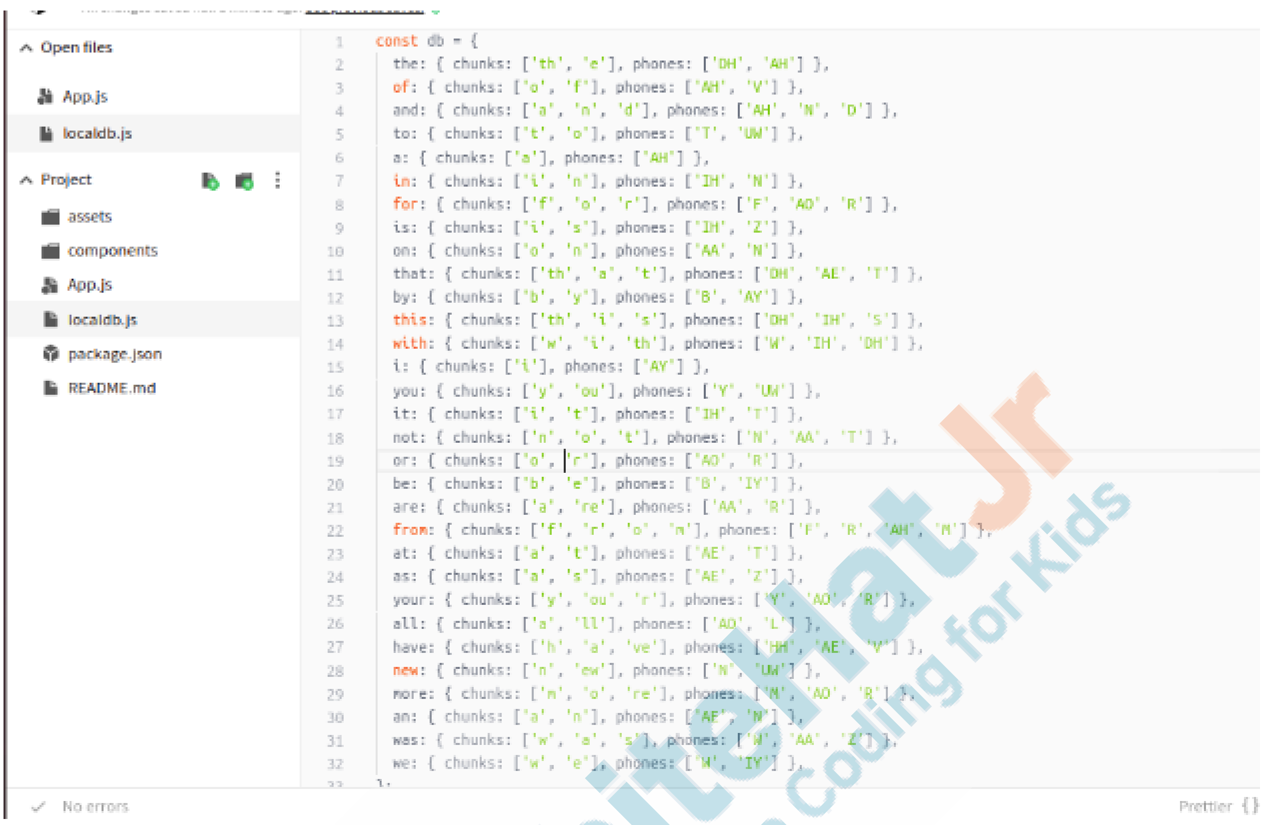
	Yes, that's an online database where we stored data in JSON format. To use Firebase Database, our users have to stay connected to the internet. There is another way we can store and use data - in a local file. We can store 'json' objects in a local file and use it to access the data we need. I will show you how.	The student is curious and asks questions.
	We have JSON data here which contains chunks of a few words in an array. It also contains the associated phonemes which we will use later. Show the student that "chunks" and "phones" of each word are stored inside the word keyname. For example: For the word "the", the chunks are stored in the array ["th", "e"]	The student looks at the JSON data.
		

```
{
  "the": {
    "chunks": [
      "th",
      "e"
    ],
    "phones": [
      "DH",
      "AH"
    ]
  },
  "of": {
    "chunks": [
      "o",
      "f"
    ],
    "phones": [
      "AH",
      "V"
    ]
  },
  "and": {
    "chunks": [
```

	Let's create a file called ‘localdb’, where we are going to store these words. Teacher creates a file called ‘localdb.js’.	



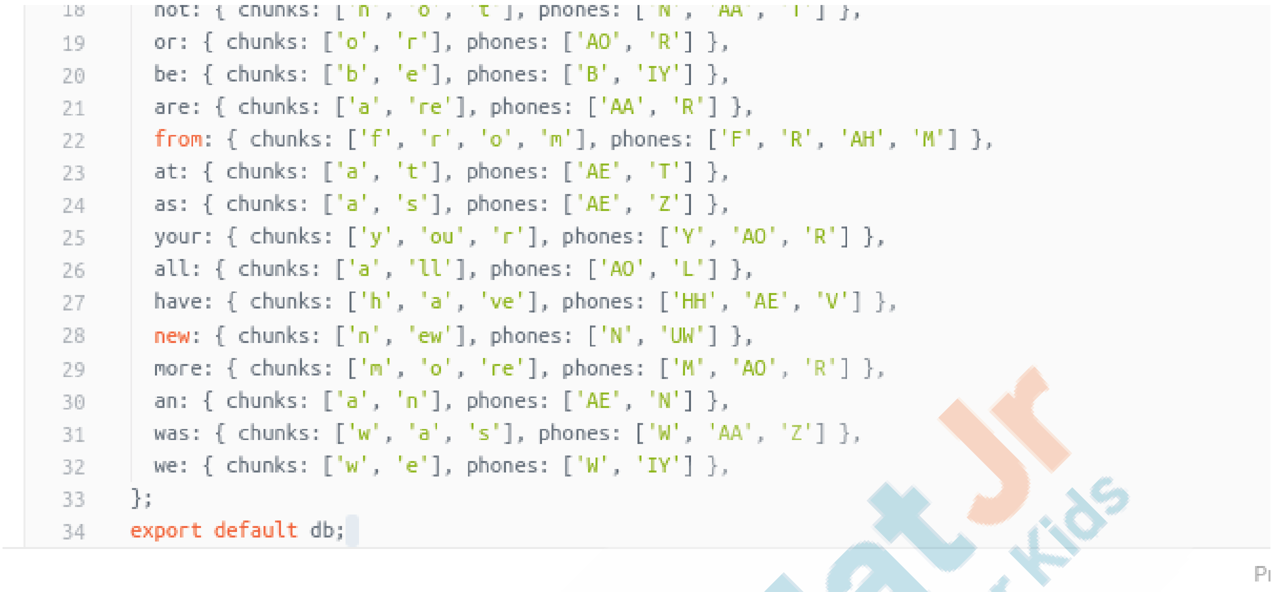
	We can create a variable called db which will hold this JSON object. Since this is not going to change in the program, we will make it a constant using the "const" keyword.	The student observes and asks questions.



```
const db = {
  the: { chunks: ['th', 'e'], phones: ['DH', 'AH'] },
  of: { chunks: ['o', 'f'], phones: ['AH', 'V'] },
  and: { chunks: ['a', 'n', 'd'], phones: ['AH', 'N', 'D'] },
  to: { chunks: ['t', 'o'], phones: ['T', 'UW'] },
  a: { chunks: ['a'], phones: ['AH'] },
  in: { chunks: ['i', 'n'], phones: ['IH', 'N'] },
  for: { chunks: ['f', 'o', 'r'], phones: ['F', 'AO', 'R'] },
  is: { chunks: ['i', 's'], phones: ['IH', 'Z'] },
  on: { chunks: ['o', 'n'], phones: ['AA', 'N'] },
  that: { chunks: ['th', 'a', 't'], phones: ['DH', 'AE', 'T'] },
  by: { chunks: ['b', 'y'], phones: ['B', 'AY'] },
  this: { chunks: ['th', 'i', 's'], phones: ['DH', 'IH', 'S'] },
  with: { chunks: ['w', 'i', 'th'], phones: ['W', 'IH', 'DH'] },
  i: { chunks: ['i'], phones: ['AY'] },
  you: { chunks: ['y', 'ou'], phones: ['Y', 'UW'] },
  it: { chunks: ['i', 't'], phones: ['IH', 'T'] },
  not: { chunks: ['n', 'o', 't'], phones: ['N', 'AA', 'T'] },
  or: { chunks: ['o', 'r'], phones: ['AO', 'R'] },
  be: { chunks: ['b', 'e'], phones: ['B', 'IY'] },
  are: { chunks: ['a', 're'], phones: ['AA', 'R'] },
  from: { chunks: ['f', 'r', 'o', 'm'], phones: ['F', 'R', 'AH', 'M'] },
  at: { chunks: ['a', 't'], phones: ['AE', 'T'] },
  as: { chunks: ['a', 's'], phones: ['AE', 'Z'] },
  your: { chunks: ['y', 'ou', 'r'], phones: ['Y', 'AO', 'R'] },
  all: { chunks: ['a', 'll'], phones: ['AO', 'L'] },
  have: { chunks: ['h', 'a', 've'], phones: ['HH', 'AE', 'V'] },
  new: { chunks: ['n', 'ew'], phones: ['N', 'UW'] },
  more: { chunks: ['m', 'o', 're'], phones: ['M', 'AO', 'R'] },
  an: { chunks: ['a', 'n'], phones: ['AE', 'N'] },
  was: { chunks: ['w', 'a', 's'], phones: ['W', 'AA', 'Z'] },
  we: { chunks: ['w', 'e'], phones: ['W', 'IY'] },
};
```

	Now, we need to export this variable db so that we can use it in our app wherever we need it. You already know how to do that. Can you help me? It is similar to how we export a class object.	The student helps the teacher with the export statement.

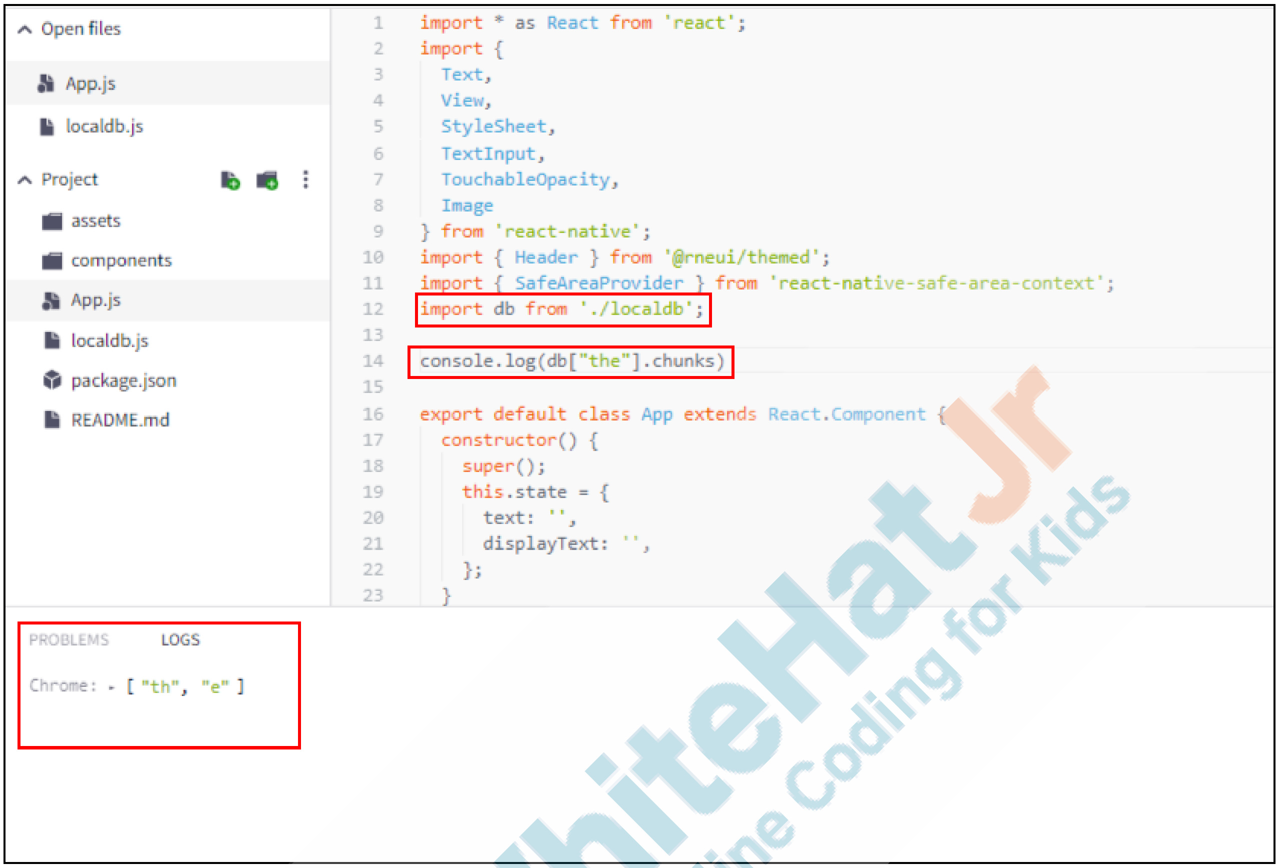
<table>
<tr><td colspan="3">



```
  not: { chunks: ['n', 'o', 't'], phones: ['N', 'AA', 'T'] },
  or: { chunks: ['o', 'r'], phones: ['AO', 'R'] },
  be: { chunks: ['b', 'e'], phones: ['B', 'IY'] },
  are: { chunks: ['a', 're'], phones: ['AA', 'R'] },
  from: { chunks: ['f', 'r', 'o', 'm'], phones: ['F', 'R', 'AH', 'M'] },
  at: { chunks: ['a', 't'], phones: ['AE', 'T'] },
  as: { chunks: ['a', 's'], phones: ['AE', 'Z'] },
  your: { chunks: ['y', 'ou', 'r'], phones: ['Y', 'AO', 'R'] },
  all: { chunks: ['a', 'll'], phones: ['AO', 'L'] },
  have: { chunks: ['h', 'a', 've'], phones: ['HH', 'AE', 'V'] },
  new: { chunks: ['n', 'ew'], phones: ['N', 'UW'] },
  more: { chunks: ['m', 'o', 're'], phones: ['M', 'AO', 'R'] },
  an: { chunks: ['a', 'n'], phones: ['AE', 'N'] },
  was: { chunks: ['w', 'a', 's'], phones: ['W', 'AA', 'Z'] },
  we: { chunks: ['w', 'e'], phones: ['W', 'IY'] },
};
export default db;
```

</td></tr>
<tr><td></td><td>Now we can simply import the variable wherever we need it and use it in our app.

Teacher shows how to import db. Teacher can also console log the chunks to show how to get the chunks for any word from the database.</td><td>The student looks at how to access the JSON object.</td></tr>
</table>



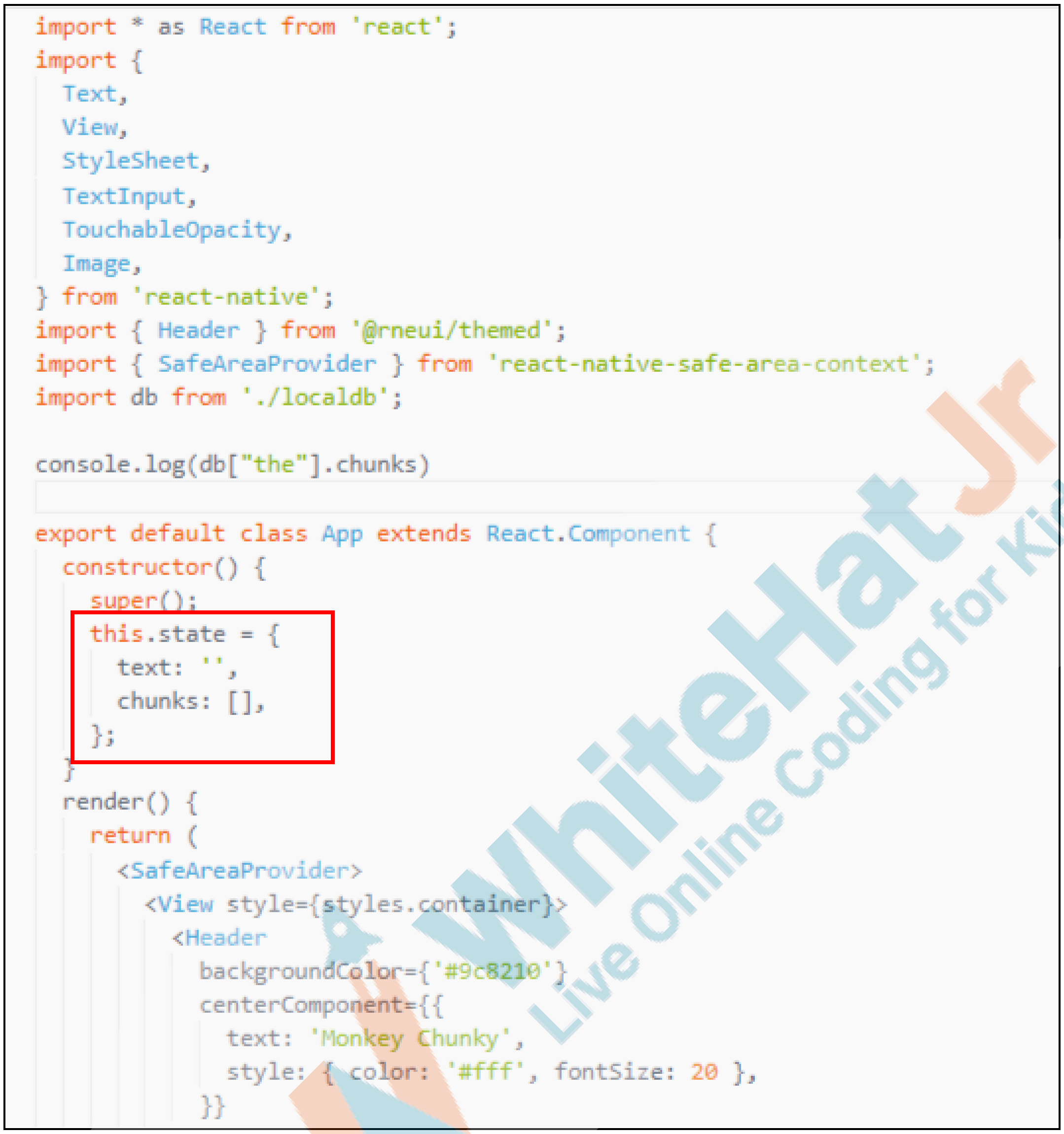
```
import * as React from 'react';
import {
  Text,
  View,
  StyleSheet,
  TextInput,
  TouchableOpacity,
  Image
} from 'react-native';
import { Header } from '@rneui/themed';
import { SafeAreaProvider } from 'react-native-safe-area-context';
import db from './localdb';

console.log(db["the"].chunks)

export default class App extends React.Component {
  constructor() {
    super();
    this.state = {
      text: '',
      displayText: '',
    };
  }
```

```
Chrome: ▸ [ "th", "e" ]
```

	Alright. Now here is a challenge for you. Can you use the data from db to display the chunks of the words typed in the input box below it. The chunks should be separate and in different lines. Hint: You will have to use the map method to render separate Text Components	The student takes up the challenge.

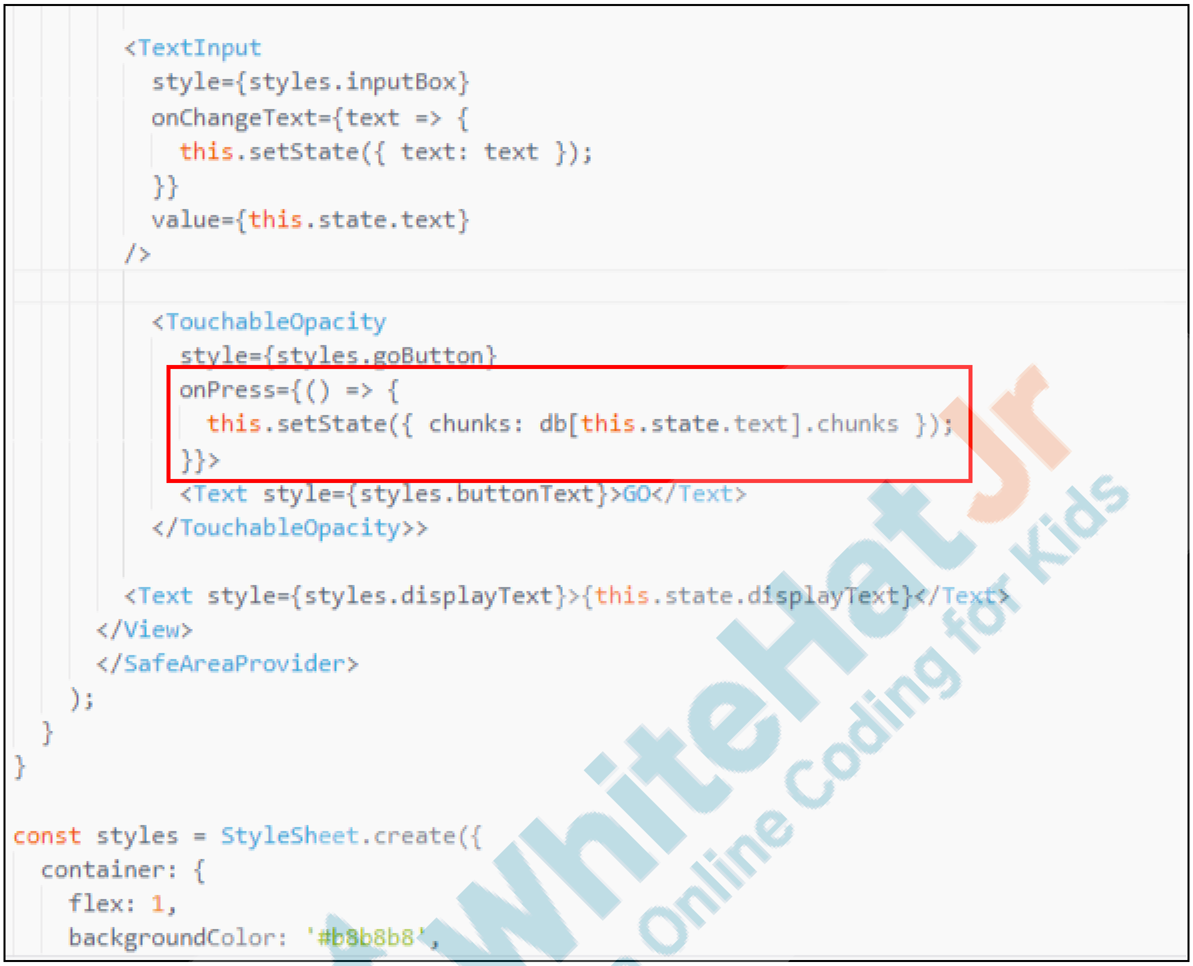
Teacher Stops Screen Share

STUDENT-LED ACTIVITY - 15 mins		
● Ask Student to press ESC key to come back to panel **● Guide Student to start Screen Share** **● Teacher gets into Fullscreen**		
ACTIVITY **● Access the chunks of any word and display them using a map method.**		
Teacher starts slideshow for slide 16 and 17.		
	Now it's your turn. Please share your screen with me.	
Teacher ends slideshow		
Step 3: **Student-Led Activity** **(15 min)**	Instead of 'displayText', let's create a state called 'chunks'. 'chunks' will be an array that will hold the parts of the word typed in the input box. For now it can be an empty array.	The student creates an empty array called 'chunks' inside the App state.



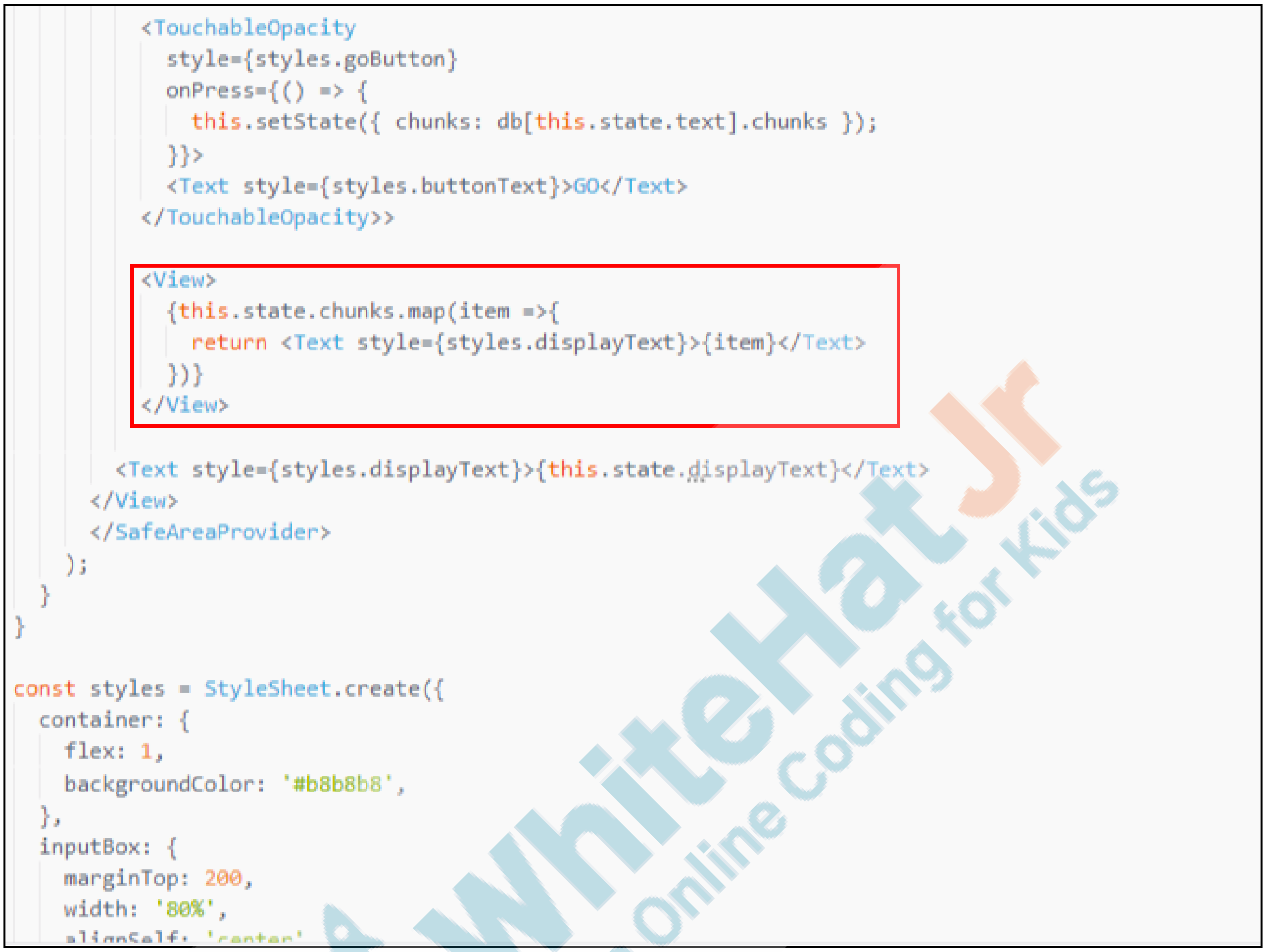
```
import * as React from 'react';
import {
  Text,
  View,
  StyleSheet,
  TextInput,
  TouchableOpacity,
  Image,
} from 'react-native';
import { Header } from '@rneui/themed';
import { SafeAreaProvider } from 'react-native-safe-area-context';
import db from './localdb';

console.log(db["the"].chunks)

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

	When "Go" Button is pressed, update the chunks.	The student updates the chunk when the “Go” button is pressed.

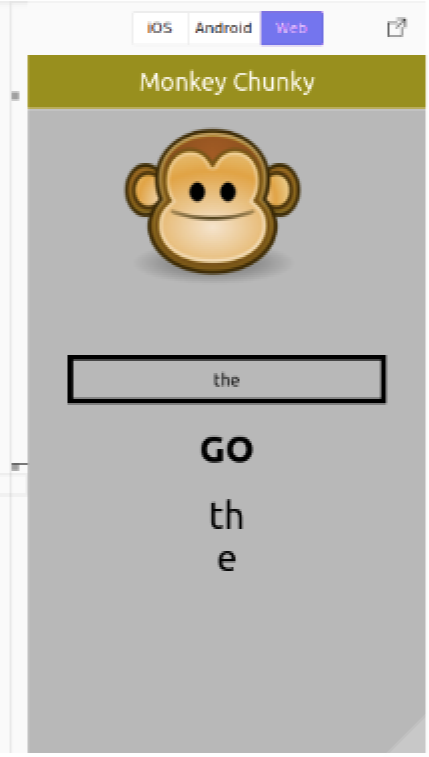


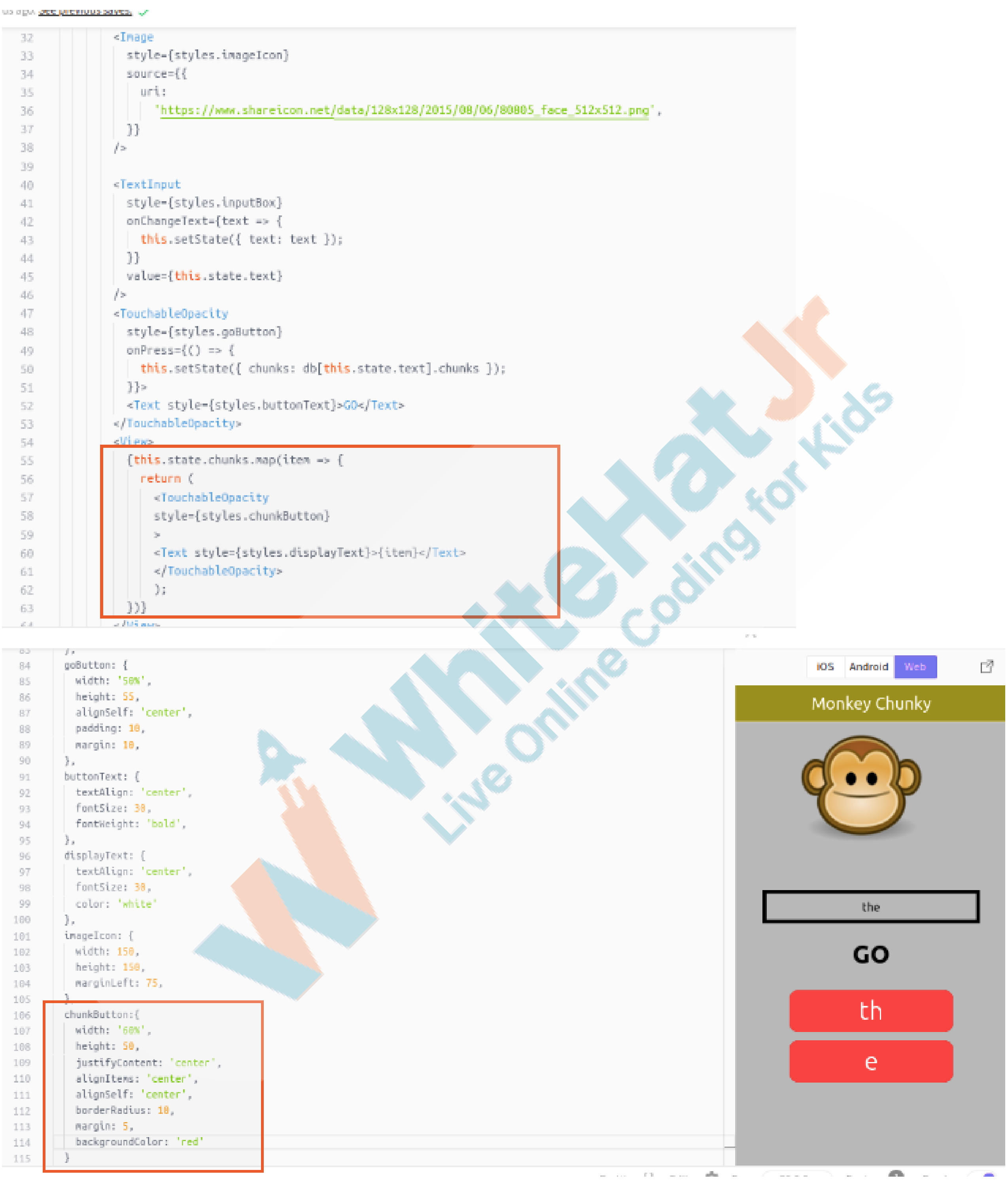
```
        <TextInput
          style={styles.inputBox}
          onChangeText={text => {
            this.setState({ text: text });
          }}
          value={this.state.text}
        />

          <TouchableOpacity
            style={styles.goButton}
            onPress={() => {
              this.setState({ chunks: db[this.state.text].chunks });
            }}>
            <Text style={styles.buttonText}>GO</Text>
          </TouchableOpacity>>

        <Text style={styles.displayText}>{this.state.displayText}</Text>
      </View>
      </SafeAreaProvider>
    );
  }
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#b8b8b8',
```

	In the render() function, inside a View Component iterate over all the elements inside the ‘chunks’ state and render a text for each chunk. Ask the student to recall the map method and how it is used. - when is map method used - what gets passed to the callback function for the map method	The student uses map method over the chunk array to render a separate text for each chunk. He/She adds ‘displayText’ style to each chunk. The student runs the code to see the output.



```
          <TouchableOpacity
            style={styles.goButton}
            onPress={() => {
              this.setState({ chunks: db[this.state.text].chunks });
            }}>
            <Text style={styles.buttonText}>GO</Text>
          </TouchableOpacity>>

          <View>
            {this.state.chunks.map(item =>{
              return <Text style={styles.displayText}>{item}</Text>
            })}
          </View>

        <Text style={styles.displayText}>{this.state.displayText}</Text>
      </View>
      </SafeAreaProvider>
    );
  }
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#b8b8b8',
  },
  inputBox: {
    marginTop: 200,
    width: '80%',
    alignSelf: 'center'
```

 iOS Android Web Monkey Chunky the GO th e		
	Now in our App, the user will be able to press on each chunk to listen to the sound of the associated phoneme. To allow the user to press the chunk, each chunk should look like a button. Can you style each text as a clickable button? Hint: Each text will now be inside a TouchableOpacity component.	The student adds the text inside a 'TouchableOpacity' component. He/She also adds some styling to the 'TouchableOpacity' component.



See previous saves.

```
        <Image
          style={styles.imageIcon}
          source={{
            uri:
              '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={() => {
            this.setState({ chunks: db[this.state.text].chunks });
          }}>
          <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>
```

```
  },
  goButton: {
    width: '50%',
    height: 55,
    alignSelf: 'center',
    padding: 10,
    margin: 10,
  },
  buttonText: {
    textAlign: 'center',
    fontSize: 30,
    fontWeight: 'bold',
  },
  displayText: {
    textAlign: 'center',
    fontSize: 30,
    color: 'white'
  },
  imageIcon: {
    width: 150,
    height: 150,
    marginLeft: 75,
  },
  chunkButton:{
    width: '60%',
    height: 50,
    justifyContent: 'center',
    alignItems: 'center',
    alignSelf: 'center',
    borderRadius: 10,
    margin: 5,
    backgroundColor: 'red'
  }
```

	Awesome! You have truly become a PRO at this. Now, all we need to do is add sounds of the respective phonemes to the buttons. You already know how sounds are used in react native app. We are going to do this in the next class.	-

Teacher Guides Student to Stop Screen Share

WRAP-UP SESSION - 5 Mins

Teacher starts slideshow from slide 18 to slide 28

Activity details	**Solution/Guidelines**
Run the presentation from slide 18 to slide 28 **Following are the wrap-up session deliverables:** ● **Explain the facts and trivias** ● **Next class challenge** ● **Project for the day** ● **Additional Activity**	Guide the student to develop the project and share with us.

Quiz time - Click on in-class quiz

Question	**Answer**
________ is an online database where we store data in JSON format. A. SQL B. Firebase C. Oracle D. DBMS	**B**

<table>
<tr><td colspan="2">Using which of the following can we add an image in React Native?
A. Image Component
B. Header Component
C. Picture Component
D. Text Component</td><td><b>C</b></td></tr>
<tr><td colspan="2">Which of the following methods is used to render separate text components?
A. Map method
B. Gap method
C. Explain method
D. Solve method</td><td><b>A</b></td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3"><b><u>FEEDBACK</u></b>
● <b>Review the learning from the class and talk about future features in the app.</b></td></tr>
<tr><td></td><td>Amazing!
In the next class our app will be properly ready and functional!</td><td></td></tr>
<tr><td></td><td>You get a “hats off”.
Till next class then. See you. Bye!</td><td>Make sure you have given at least 2 Hats Off during the class for:
Creatively Solved Activities +10
Great Question +10
Strong Concentration +10</td></tr>
<tr><td><b>Project Pointers and Cues (5 min)</b></td><td><b>* This Project will take only 30 mins to complete. Motivate students to</b></td><td><b>Note: You can assign the project to the student in class itself by clicking on</b></td></tr>
</table>

	try and finish it immediately after the class. DICTIONARY APP - OFFLINE VERSION **Goal of the Project:** In Class 64, you have learnt how to access data from local files. You have used the data from a local file to break a given word into smaller chunks associated with a phoneme sound. You will be using this concept to modify the online dictionary created in the previous project into an offline dictionary. This way users can search words even when they are offline. **Story:** Sara and Josh are participating in a treasure hunt where the hints are hidden in the meanings of different words. You have created an online Dictionary App to help Josh and Sara find the meanings of the words. Now you have to create a dictionary app which will work without an internet connection. I am very excited to see your project solution and I know you both will do really well. Bye Bye!	**the Assign Project button which is available under the projects tab.**

Teacher Clicks ✖ End Class		
Teacher ends slideshow		
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 as a reflection journal.

Activity	Activity Name	Links
Teacher Activity 1	Class Activity	https://snack.expo.dev/@procodingclass/monkey-chunky-stage-1:-reference
Teacher Activity 2	Teacher Reference	https://snack.expo.dev/@procodingclass/monkeychunky-stage2--reference
Student Activity 1	Class Activity	https://snack.expo.dev/@procodingc

		lass/monkey-chunky-stage-1:-reference
Student Activity 2	JSON data	https://github.com/rajeevrjha/monkey-chunky/blob/master/db/db_1.json
Project Solution	Dictionary App-Offline Version	https://github.com/priyapandey2020/35ae9f7b26a91613b02956474e7894a2
Teacher Reference visual aid link	Visual aid link	https://curriculum.whitehatjr.com/Visual+Project+Asset/PRO_VD/BJFC_PRO_V3_C64_withcues.html
Teacher Reference In-class quiz	In-class quiz	https://s3-whjr-curriculum-uploads.whjr.online/494325fa-1bc2-4a5e-bc87-40bcdeec3653.pdf