
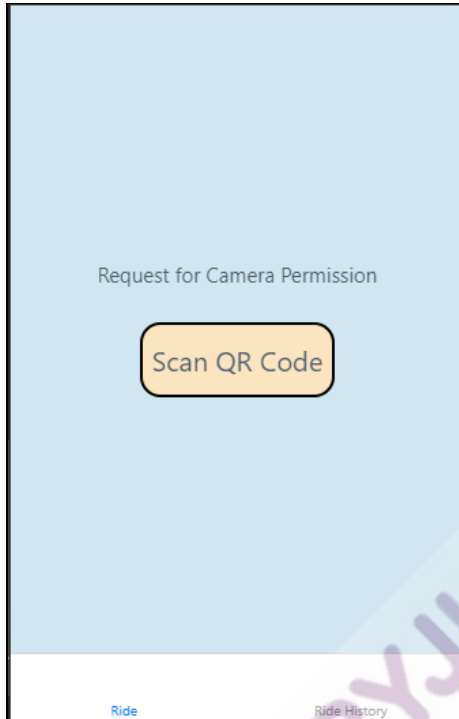


Topic	QR CODE SCANNER	
Class Description	Students learn how to perform QR Code/Barcode scanning in a React Native application. Students write to scan QR code and display the data inside a Text component.	
Class	C69	
Class time	50 min	
Goal	<ul style="list-style-type: none"> • Get permissions to use the camera in application. • Use the BarCodeScanner component to scan a QR code. • Display data from QR code inside a Text component. 	
Resources Required	<ul style="list-style-type: none"> • Teacher Resources <ul style="list-style-type: none"> ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Android/iOS Smartphone with Expo App installed • Student Resources <ul style="list-style-type: none"> ○ 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 20 mins 20 mins 5 mins
Credits	<p>Image of QR Code by Nicolas1981 is licensed under the CC-BY- SA 3.0 license from Wikimedia Commons.</p> <p>Image of Barcode by porges is licensed under the Public Domain of Wikimedia Commons.</p>	

Permissions	Expo permissions require the camera permission to access the camera to scan the QR code.	
WARM-UP SESSION - 5 mins		
<p align="center">CONTEXT</p> <ul style="list-style-type: none"> Compare how books and id cards are scanned by barcode scanning devices. 		
<p align="center">  Teacher starts slideshow from slides 1 to 13 Refer to speaker notes and follow the instructions on each slide. </p>		
Activity details		Solution/Guidelines
<p><i>Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today?</i></p> <p>Run the presentation from slide 1 to slide 5.</p> <p>Following are the WARM-UP session deliverables:</p> <ul style="list-style-type: none"> Greet the student. Revision of previous class activities. Quizzes 		<p>ESR: Hi, thanks, Yes I am excited about it!</p> <p>Click on the slide show tab and present the slides</p>
QnA Session		
Question		Answer
<p>Choose the correct block of code to determine if the app has been granted permissions or not.</p> <p>A. <code>const { status } = await Permissions.askAsync(Permissions.CAMERA);</code></p> <p>B. <code>const { status } = Permissions.askAsync(Permissions.CAMERA);</code></p> <p>C. <code>const { status } = await Permissions.askAsync(Permissions);</code></p> <p>D. <code>const { status } = await Permissions.askAsync(CAMERA);</code></p>		A
Choose the correct block of code which can be used to		B

create a button, which when clicked gets the camera permissions and scans the barcode.



A.

```

/*
<TouchableOpacity
  style={[styles.button, { marginTop: 25 }]}
  onPress= this.getCameraPermissions("scanner")>
    <Text style={styles.buttonText}>Scan QR Code</Text>
  </TouchableOpacity>
*/

```

B.

```

/*
<TouchableOpacity
  style={[styles.button, { marginTop: 25 }]}
  onPress={() => this.getCameraPermissions("scanner")}>
    <Text style={styles.buttonText}>Scan QR Code</Text>
  </TouchableOpacity>
*/

```

C.

```

/*
<TouchableOpacity
  style={[styles.button, { marginTop: 25 }]}
  onPress={() => this.getCameraPermissions()}>
    <Text style={styles.buttonText}>Scan QR Code</Text>
  </TouchableOpacity>
*/


```

D.

```

/*
<TouchableOpacity
  style={[styles.button, { marginTop: 25 }]}
  onPress={() => this.getCameraPermissions("scanner")}
  />
*/

```

Continue the WARM-UP session	
Activity details	Solution/Guidelines
<p>Run the presentation from slide 6 to slide 13 to set the problem statement.</p> <p>Following are the WARM-UP session deliverables:</p> <ul style="list-style-type: none"> • Appreciate the student. • Explain how to use QR code feature in e-library app 	<p>Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students.</p>
<p>Teacher ends slideshow </p>	
TEACHER-LED ACTIVITY - 20 mins	
Teacher Initiates Screen Share	
CHALLENGE	
<ul style="list-style-type: none"> • Use camera permissions and expo-barcode-scanner package to scan a QR code. 	
Teacher Action	Student Action
<p>Before we start, let's try to understand what is a Barcode and QR code and what is the difference between them.</p> <p>Have you ever read about them?</p>	<p>ESR: Varied.</p>
<p>You might have seen a Barcode on all products that you purchase - books, packed food items, etc.</p> <p>These codes have information embedded in them. They can contain strings / text which convey information like - name of the product, price, etc.</p> <p>The advantage of QR code and Barcode is that they are machine readable and machines can read information</p>	

<p>printed on them very quickly.</p> <p>You can learn more about QR code and Barcode through the reference links provided in the lesson.</p> <p><i>Teacher explains about QR code & Barcode. Teacher can also ask student to use any school book to check QRCode and barcode.</i></p> <p>QRCode: A QR code (short for "quick response" code) is a type of Barcode that contains a matrix of dots. All QR codes have a square shape. It can be scanned using a QR scanner or a smartphone with a built-in camera. Once scanned, software on the device converts the dots within the code into numbers or a string of characters.</p> <p>Barcode is a printed series of parallel bars or lines of varying width used to enter data into a computer system—mainly used for stock keeping in shops. The bars are typically black on a white background, and their width and quantity vary according to application. The numbers represented by a barcode are also printed out at its base.</p>	<p><i>Student scans through the Wikipedia references provided in the Student Activity 1 and Student Activity 2.</i></p>
<p>Alright. Let's get started in building our own Barcode/QR code scanner.</p> <p>Let's open our project from last class.</p> <p><i>Teacher can clone the project from the last class using Teacher Activity 1.</i></p> <p><i>Teacher also installs all the dependencies from package.json using <code>npm install</code> / <code>yarn install</code>.</i></p> <p>steps to clone the project:- <code>git clone <projectURL></code></p>	

<pre>cd <projectFolder> npm install</pre>	
<p><i>Teacher opens the cloned directory in VS Code.</i></p> <p>Can you quickly go over the code from the previous class and explain what we were doing?</p>	<p><i>The student goes through the code and explains how tab navigation works.</i></p>
<p>Currently our Transaction screen only has text inside the View component.</p> <p>Let's create a Button (using TouchableOpacity) which will trigger the QR code scanner and display the scanned output as text.</p> <p>Can you guide me on how to do that?</p> <p><i>Teacher modifies the code in the Transaction Screen to display a Button and a text with student input.</i></p>	<p><i>The student guides the teacher to display a Dummy Text output and a Touchable Opacity Button.</i></p>
<pre>import React, { Component } from "react"; import { View, Text, StyleSheet, TouchableOpacity } from "react-native"; export default class TransactionScreen extends Component { render() { return (<View style={styles.container}> <TouchableOpacity> <Text>Scan QR Code</Text> </TouchableOpacity> </View>); } }</pre>	

```
}
```

Let's add some styling to our TouchableOpacity and Text using StyleSheet.

Teacher can copy some styling to the button and text while the student gives his/her inputs.

The student gives his/her input to style the items.

```
export default class TransactionScreen extends Component {
  render() {

    return (
      <View style={styles.container}>

        <TouchableOpacity
          style={styles.button}
        >
          <Text style={styles.buttonText}>Scan QR Code</Text>
        </TouchableOpacity>
      </View>
    );
  }
}
```

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: "center",
    alignItems: "center",
    backgroundColor: "#5653D4"
  },
  text: {
    color: "#ffff",
    fontSize: 15
  },
  button: {
```



```
width: "43%",
height: 55,
justifyContent: "center",
alignItems: "center",
backgroundColor: "#F48D20",
borderRadius: 15
},
buttonText: {
fontSize: 24,
color: "#FFFFFF"
}
});
```

We will be using an expo package (library) which will help us build the QR code scanner in our application.

Let's install this package in the folder where we are working. Do you remember how we install packages using npm?

Teacher runs the command and waits for the installation to finish.

ESR:

```
expo install
expo-barcode-scanner
```

```
C:\Users\ADMIN\e-library>expo install expo-barcode-scanner
```

```
There is a new version of expo-cli available (4.6.0).
You are currently using expo-cli 4.4.3
Install expo-cli globally using the package manager of your choice;
for example: `npm install -g expo-cli` to get the latest version
```

```
Installing 1 SDK 41.0.0 compatible native module using npm.
```

Now, let's quickly go over the expo-barcode-scanner documentation to see how to use this package.

The student and teacher go through the

<p><i>Teacher opens the document from Teacher Activity 2. And checks the usage part</i></p>	<p><i>expo-barcode-scanner documentation.</i></p>
<p>The code might look confusing but let's start writing the code and we will soon understand how to use the barcode scanner library.</p> <p>You might have noticed that our app needs to ask for Camera permissions first before using it to scan QR codes.</p> <p>Let's import the Barcode scanner component and permissions.</p> <p>*Note: Teacher has to also install expo-permissions using the following command :- expo install expo-permissions</p>	<p><i>The student observes and asks questions.</i></p>
<div data-bbox="159 1066 1425 1297"> <pre>C:\Users\ADMIN\e-library>expo install expo-permissions There is a new version of expo-cli available (4.6.0). You are currently using expo-cli 4.4.3 Install expo-cli globally using the package manager of your choice;</pre> </div> <div data-bbox="159 1339 1425 1864"> <pre>import React, { Component } from "react"; import { View, Text, StyleSheet, TouchableOpacity } from "react-native"; import * as Permissions from "expo-permissions"; import { BarCodeScanner } from "expo-barcode-scanner"; export default class TransactionScreen extends Component { render() { return (<View style={styles.container}> <TouchableOpacity</pre> </div>	

```

    style={styles.button}
  >
    <Text style={styles.buttonText}>Scan QR Code</Text>
  </TouchableOpacity>
</View>
);
}
}

```

Let's define three states in our application:

- **domState** : " => This will tell if the app is in scanner mode or scanned mode.
- **hasCameraPermissions**: null => This will tell if the user has granted camera permission to the application.
- **scanned**: false => This will tell if scanning has been completed or not.
- **scannedData**: " => This will hold the scanned data that we get after scanning.

Now we know the states that we require. Can you help me create these states?

The student guides the teacher on how to create these states with default values.

```

import React, { Component } from "react";
import { View, Text, StyleSheet, TouchableOpacity } from "react-native";
import * as Permissions from "expo-permissions";
import { BarCodeScanner } from "expo-barcode-scanner";

export default class TransactionScreen extends Component {
  constructor(props){
    super(props);
    this.state = {
      domState : "normal",
      hasCameraPermissions: null,

```

```
scanned: false,
```

```
scannedData: ""
```

```
}
```

```
}
```

When we press our Scan QR Code Button in our application, we need to request for camera permission.

Let's write a function - **getCameraPermission** - which can request for camera permission.

Note that this function needs to be asynchronous because it takes time for the user to give camera permission to the application.

The **Permission component** which we imported has a predefined function called **.askAsync()** which can request for various permissions.

We will use this to request for camera permission inside our function and change the state of **hasCameraPermissions**.

Note that **.askAsync()** returns an object with a 'status' key containing the status of the permission granted by the user. If the user grants permission, status changes to 'granted'

*Note: **{status}** automatically extracts the value from the object with key 'status'.

*Teacher writes the code for **getCameraPermission** while explaining to the student.*

The student observes and asks questions.

```
import React, { Component } from "react";
import { View, Text, StyleSheet, TouchableOpacity } from "react-native";
import * as Permissions from "expo-permissions";
import { BarCodeScanner } from "expo-barcode-scanner";

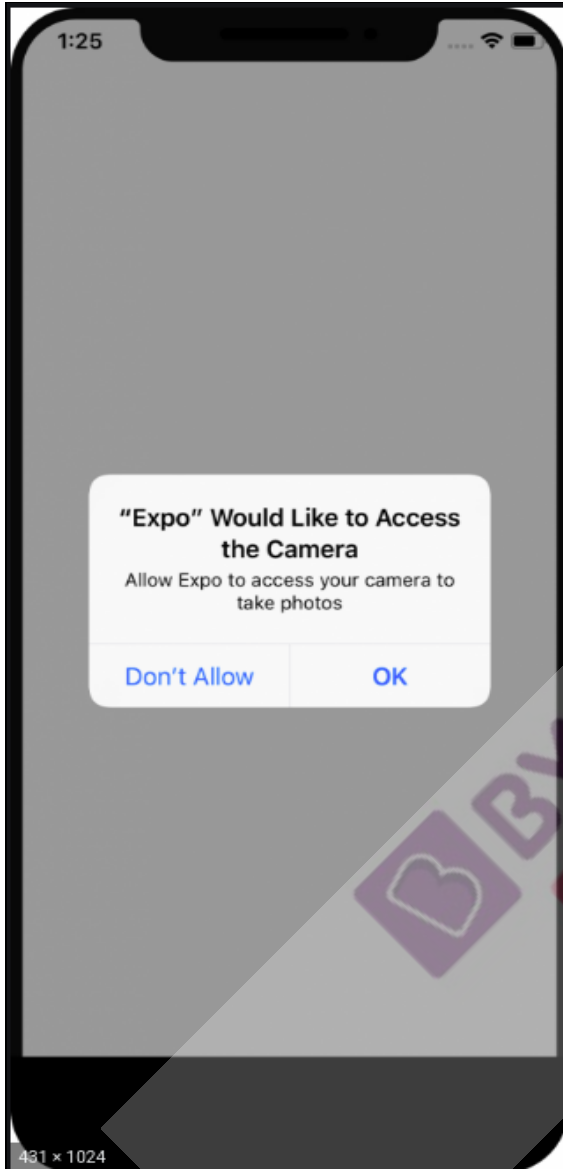
export default class TransactionScreen extends Component {
  constructor(props){
    super(props);
    this.state = {
      domState : "normal",
      hasCameraPermissions: null,
      scanned: false,
      scannedData: ""
    }
  }

  getCameraPermissions = async domState => {
    const { status } = await Permissions.askAsync(Permissions.CAMERA);

    this.setState({
      /*status === "granted" is true when user has granted permission
      status === "granted" is false when user has not granted the permission
      */
      hasCameraPermissions: status === "granted",
      domState: domState,
      scanned: false
    });
  };
};
```

Let's call the function **getCameraPermissions** when the TouchableOpacity Button is pressed using its onPress prop.

Teacher writes the code and tests it.



What do you see?

The camera permission is asked only once. If you grant the permission, the application will remember it.

ESR:

Pressing the button makes the application ask for camera permission.

```
render() {
```

```
  return (
```

```
<View style={styles.container}>

  <TouchableOpacity
    style={[styles.button, { marginTop: 25 }]}
    onPress={() => this.getCameraPermissions("scanner")}
  >
    <Text style={styles.buttonText}>Scan QR Code</Text>
  </TouchableOpacity>
</View>
);
}
```

We need to tell our application what to do if, **hasCameraPermissions** is 'null' or 'false'.

Let's say that we want to display a text "Request for camera permission" if **hasCameraPermissions** is false or null.

If **hasCameraPermissions** is 'true', we will display whatever text is inside the **scannedData** state. Currently **scannedData** is an empty string.

Let us write code for this in our **render()** function.

Note that we can write JSX only in the return function. If we are writing JavaScript code, we do it inside the curly {} brackets.

Teacher writes code using a ternary operator.

The student explains how the ternary operator works and what the current line of code is doing.

The student and teacher test the working of the code.

```
render() {
  const { domState, hasCameraPermissions, scannedData, scanned } = this.state;

  return (
    <View style={styles.container}>
      <Text style={styles.text}>
        {hasCameraPermissions ? scannedData : "Request for Camera Permission"}
      </Text>
      <TouchableOpacity
        style={[styles.button, { marginTop: 25 }]}
        onPress={() => this.getCameraPermissions("scanner")}
      >
        <Text style={styles.buttonText}>Scan QR Code</Text>
      </TouchableOpacity>
    </View>
  );
}
```

Now, we want to display our BarCodeScanner when our Scan Button is clicked.

How would we know that our Scan Button has been clicked?

We have already created a **domState** that keeps track if the button has been clicked.

ESR: Varied.

- domState will be 'normal' when the application starts.
- When the button is clicked to get camera permissions, domState should change to 'scanner'.

Teacher writes code with student inputs.

The student helps the teacher and asks questions.


```
constructor(props){
  super(props);
  this.state = {
    domState : "normal",
    hasCameraPermissions: null,
    scanned: false,
    scannedData: ""
  }
}
```

Now, we want to return a `BarCodeScanner` component when the button is clicked and the user has given camera permissions.

The **BarCodeScanner** component automatically starts scanning using the Camera. It has a prop called **onBarCodeScanned** which can call a function to handle data received after scanning. We want to call this function only when the **scanned** is false.

Let's write a function called **handleBarCodeScanned** which is called when the scan is completed.

- This function automatically receives the type of barcode scanned and the data inside the barcode. We can set the **scannedData** here to be equal to the data received after scanning.
- Once the scan has been completed, we also want to set the **scanned state** to **true**. We also want to change the state for the **domState** to make it back to normal when the scan is completed.

Teacher writes the code to display BarCodeScanner when hasCameraPermissions is true and domState is clicked.

The student observes and asks questions around the code.

```

getCameraPermissions = async domState => {
  const { status } = await Permissions.askAsync(Permissions.CAMERA);

  this.setState({
    /*status === "granted" is true when user has granted permission
      status === "granted" is false when user has not granted the permission
    */
    hasCameraPermissions: status === "granted",
    domState: domState,
    scanned: false
  });
};

handleBarCodeScanned = async ({ type, data }) => {
  this.setState({
    scannedData: data,
    domState: "normal",
    scanned: true
  });
};

render() {
  const { domState, hasCameraPermissions, scannedData, scanned } = this.state;
  if (domState === "scanner") {
    return (
      <BarcodeScanner
        onBarCodeScanned={scanned ? undefined : this.handleBarCodeScanned}
        style={StyleSheet.absoluteFillObject}
      />
    );
  }
}

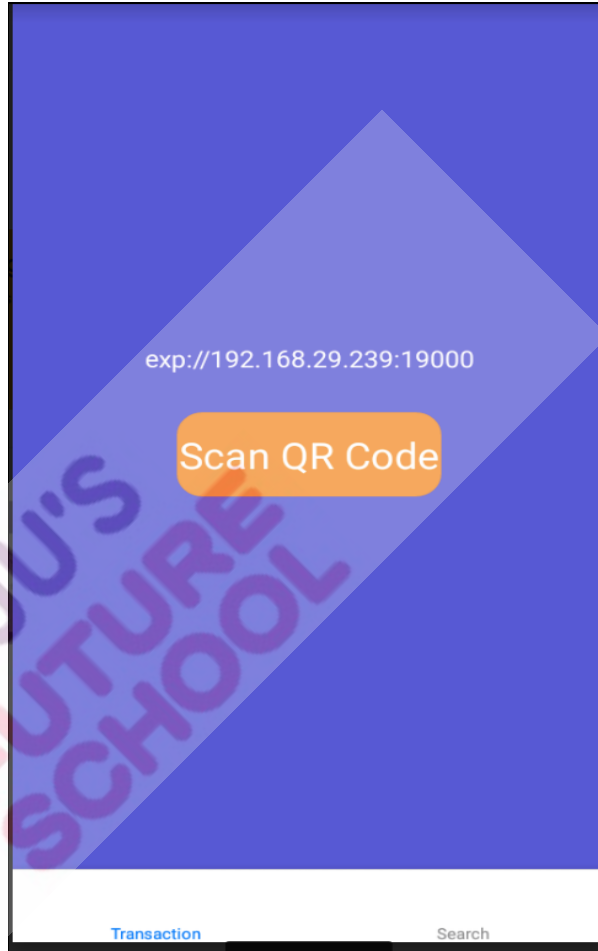
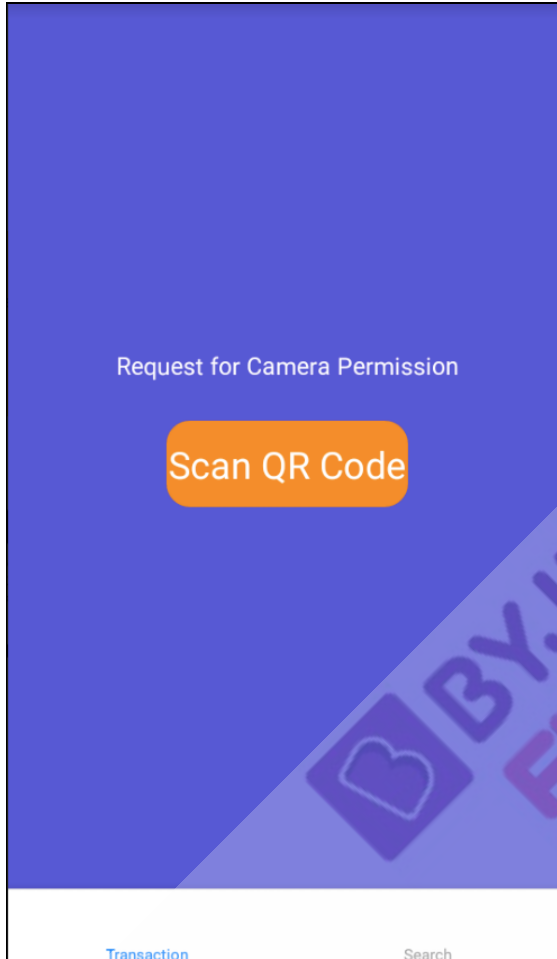
```

Let's execute our application and see how it works.

The teacher can use the QR code generator link ([Teacher](#)

Teacher and student test the application to see if it works as expected.

Activity 3) to generate QR codes with random data for testing.




Great! Did you enjoy working on this QR code scanner?
Can you make one for your own application now?

Students take up the challenge to embed the QR code scanner in their application.

Teacher Stops Screen Share

Now it's your turn. Please share your screen with me.

STUDENT-LED ACTIVITY - 20 mins	
<ul style="list-style-type: none"> Ask the student to press the ESC key to come back to the panel. Guide the student to start Screen Share. The teacher gets into Fullscreen. 	
<p align="center">ACTIVITY</p> <ul style="list-style-type: none"> Display data from QR code inside a Text Component. 	
<div>  <p>Teacher starts slideshow :Slide 14 to 15</p> <p>Refer to speaker notes and follow the instructions on each slide.</p> </div>	
Teacher Action	Student Action
<p><i>Teacher guides the student to open the project.</i></p>	<p><i>The student opens the code from the previous class.</i></p> <p>Student Activity 5</p>
<p>So what's the first thing we did in the app?</p> <p><i>Teacher guides the student to create the states needed in the application.</i></p>	<p>ESR:</p> <p>We need to create the states .</p> <p><i>In the BookTransaction screen, student creates the states:</i></p> <ul style="list-style-type: none"> <i>hasCameraPermissions</i> <i>scanned</i> <i>scannedData</i> <i>domState</i>

```
import React, { Component } from "react";
import { View, Text, StyleSheet, TouchableOpacity } from "react-native";
import * as Permissions from "expo-permissions";
import { BarCodeScanner } from "expo-barcode-scanner";

export default class TransactionScreen extends Component {
  constructor(props){
    super(props);
    this.state = {
      domState : "normal",
      hasCameraPermissions: null,
      scanned: false,
      scannedData: ""
    }
  }
}
```

What do we have to do next?

Yes! and why do we need the button?

Guide the student to create the Text and Button needed on the screen.

ESR:

We need to add a button.

ESR:

We need the button as when this button is pressed we'll open the scanner to scan the QR code.

The student creates a Text and TouchableOpacity components on the screen.

```
import React, { Component } from "react";
import { View, Text, StyleSheet, TouchableOpacity } from "react-native";

export default class TransactionScreen extends Component {
  render() {

    return (
```

```

<View style={styles.container}>
  <TouchableOpacity>
    <Text>Scan QR Code</Text>
  </TouchableOpacity>
</View>
);
}
}

```

Now when the button is clicked we need to get permission from the device to use the camera, and then update the state when the user gives the permission.

Guide the student to ask for camera permissions when the button is clicked.

The student writes code to request camera permissions when the button is clicked. He/She changes the state for hasCameraPermissions and buttonState.

```

render() {
  const { domState, hasCameraPermissions, scannedData, scanned } = this.state;

  return (
    <View style={styles.container}>
      <Text style={styles.text}>
        {hasCameraPermissions ? scannedData : "Request for Camera Permission"}
      </Text>
      <TouchableOpacity
        style={[styles.button, { marginTop: 25 }]}
        onPress={() => this.getCameraPermissions("scanner")}
      >
        <Text style={styles.buttonText}>Scan QR Code</Text>
      </TouchableOpacity>
    </View>
  );
}
}

```

What do we want to do when the user gives us the permissions?

Guide the student to display BarCodeScanner when the button is clicked and the user has granted camera permissions.

ESR:

When the user gives us the permission we want to open the barcode scanner to scan the QR code.

The student writes code to display the BarCodeScanner component when domState is 'scanner' and hasCameraPermissions is true.

When BarCode is scanned, the student calls a function called 'handleBarCodeScanned' inside onBarCodeScanned prop.

```
getCameraPermissions = async domState => {
  const { status } = await Permissions.askAsync(Permissions.CAMERA);

  this.setState({
    /*status === "granted" is true when user has granted permission
    status === "granted" is false when user has not granted the permission
    */
    hasCameraPermissions: status === "granted",
    domState: domState,
    scanned: false
  });
};

handleBarCodeScanned = async ({ type, data }) => {
  this.setState({
    scannedData: data,
```



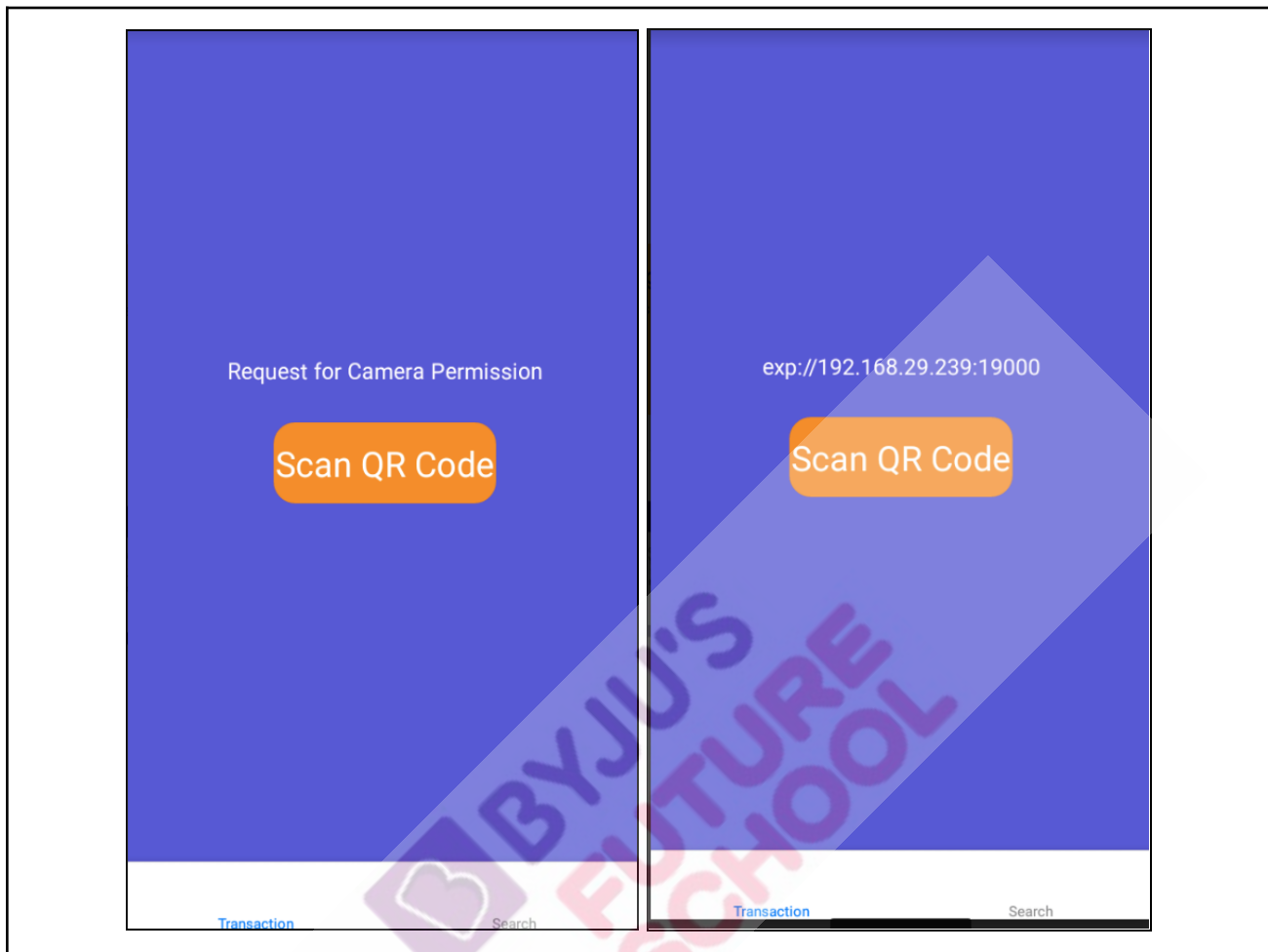
```
domState: "normal",
scanned: true
});
};

render() {
  const { domState, hasCameraPermissions, scannedData, scanned } = this.state;
  if (domState === "scanner") {
    return (
      <BarcodeScanner
        onBarCodeScanned={scanned ? undefined : this.handleBarCodeScanned}
        style={StyleSheet.absoluteFillObject}
      />
    );
  }
}
```

Now you have all the code in position we just need to let's test and see if we are able to scan the QR code.

Guide the student to test the application

The student runs the code using expo start and tests on their mobile phone.



Awesome job! today you wrote code to scan the QR code and automatically input information when book id and student ids are scanned.

Teacher Guides Student to Stop Screen Share

WRAP-UP SESSION - 5 Mins

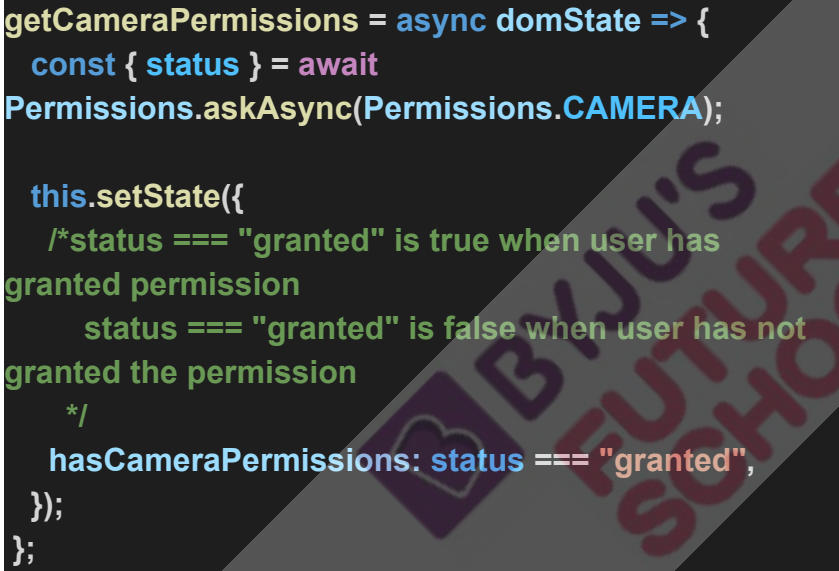
Teacher starts slideshow






from slide 16 to slide 26


Activity details

Solution/Guidelines

<p>Run the presentation from slide 16 to slide 26</p> <p>Following are the WRAP-UP session deliverables:</p> <ul style="list-style-type: none"> • Appreciate the student. • Revise the current class activities. • Discuss the quizzes. 	<p>Discuss with the student the current class activities and Student will ask doubts related to the activities.</p>
<p>Quiz time - Click on in-class quiz</p>	
Question	Answer
 <pre> getCameraPermissions = async domState => { const { status } = await Permissions.askAsync(Permissions.CAMERA); this.setState({ /*status === "granted" is true when user has granted permission status === "granted" is false when user has not granted the permission */ hasCameraPermissions: status === "granted", }); }; </pre> <p>After running the code, if we run console.log(hasCameraPermissions), what would be output of the same?</p> <p>A. "granted" B. status=== "granted" C. True or False D. Yes or No</p>	<p>C</p>
<p>Which prop of the Barcode Scanner component calls a function to handle data received after scanning?</p> <p>A. onCodeScanned B. onBarScanned</p>	<p>D</p>

<p>C. BarCodeScanned D. onBarCodeScanned</p>	
<p>Which predefined function of the Permission component can help us to request for various permissions?</p> <p>A. .askPermission() B. .Async() C. .askAsync() D. .Permission()</p>	<p>C</p>
<p>Teacher Action</p>	<p>Student Action</p>
<p>In today's class, we learned how to create a BarCodeScanner using Camera Permissions.</p> <p>Can you think about how we can use this in issuing or returning books?</p>	<p>ESR: We can get the data after scanning the book and student id cards and autofill the issue or return forms.</p>
<p>Exactly! That is what we are going to do in our next class.</p> <p>Hope you enjoyed today's class and working on this application</p>	
<p>You get a "hats off". In the next class, we will write code to automatically input information when book id and student ids are scanned. See you. Bye!</p>	<p><i>Make sure you have given at least 2 Hats Off during the class for:</i></p> <div> <p>Creatively Solved Activities  +10</p> <p>Great Question  +10</p> </div>

	
<p>* This Project will take only 30 mins to complete. Motivate students to try and finish it immediately after the class.</p> <p>Project Overview E-RIDER STAGE 2</p> <p>Goal of the Project:</p> <p>In class 69, you have learned about QR code Scanner and to get permission from a user to use Mobile Camera. In this project, you have to practice applying what you have done in the class and creating a Barcode/QR code Scanner application.</p> <p>* This is a continuation of Project-68; make sure you have completed and submitted that before attempting this one.</p> <p>Story:</p> <p>Your friend Vihaan is very happy with your idea; he has suggested that each bicycle will have a QR code assigned to it which the app should be able to scan. You need to include the QR Code scanner in your app to receive further details.</p> <p>I am very excited to see your project solution and I know you both will do really well.</p> <p>Bye Bye!</p>	<p>Note: You can assign the project to the student in class itself by clicking on the Assign Project button which is available under the projects tab.</p>



Teacher ends slideshow

Teacher Clicks

✕ End Class

ADDITIONAL ACTIVITIES

<p>Additional Activities</p> <p><i>Encourage the student to write reflection notes in their reflection journal using markdown.</i></p> <p>Use these as guiding questions:</p> <ul style="list-style-type: none"> What happened today? <ul style="list-style-type: none"> 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? 	<p><i>The student uses the markdown editor to write her/his reflection as a reflection journal.</i></p>
---	---

Activity	Activity Name	Links
Student Activity 1	QR code image	https://commons.wikimedia.org/wiki/File:Japan-qr-code-billboard.jpg
Student Activity 2	Barcode image	https://commons.wikimedia.org/wiki/File:Code93.png
Student Activity 3	BarCodeScanner Documentation	https://docs.expo.io/versions/latest/sdk/bar-code-scanner/

Student Activity 4	QR-Code Generator	https://www.the-qrcode-generator.com/
Student Activity 5	Previous Class Code	https://github.com/procodingclass/e-library-PRO-C68
Teacher Activity 1	Previous Class Code	https://github.com/procodingclass/e-library-PRO-C68
Teacher Activity 2	BarCodeScanner Documentation	https://docs.expo.io/versions/latest/sdk/bar-code-scanner/
Teacher Activity 3	QR-Code Generator	https://www.the-qrcode-generator.com/
Teacher Activity 4	Reference	https://github.com/procodingclass/e-library-v2-PRO-C69
Teacher Reference visual aid link	Visual aid link	https://curriculum.whitehatjr.com/Visual+Project+Asset/PRO_VD/BJFC-PRO-V3-C69-withcues.html
Teacher Reference In-class quiz	In-class quiz	https://s3-whjr-curriculum-uploads.whjr.online/6c621114-511b-4ba4-8ba3-b268dc691071.pdf
Project Solution	E-Ride Stage-2	https://github.com/procodingclass/PRO-C69-PROJECT