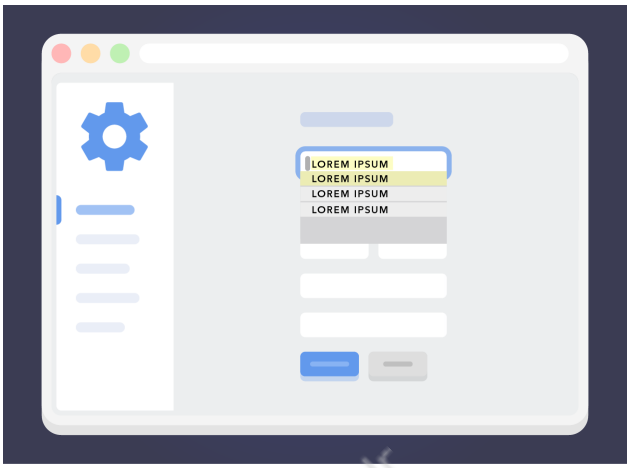
AUTOFILL TEXT INPUTS



What is our GOAL for this MODULE?

In this class, we designed the input form for the issue/return screen of the app and wrote code to autofill input information when book id and student ids are scanned. Also customize the appearance of tab navigation by adding images/icons to the tabs.

What did we ACHIEVE in the class TODAY?

- Customized the appearance of the bottom tabs by adding custom icons/images to it.
- Designed the input form for the issue/return screen of the app.
- Wrote code to automatically populate the input box when book id and student id are scanned.

Which CONCEPTS/ CODING BLOCKS did we cover today?

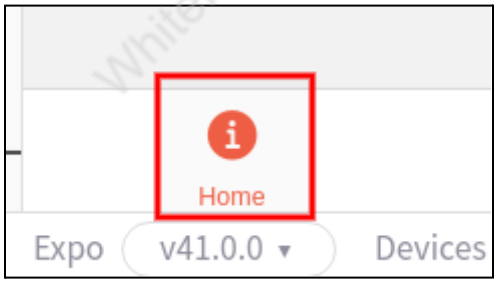
- TextInput Component
- Tab Navigation

How did we DO the activities?

1. Check the documentation for customizing the tabnavigator.
2. As seen in the documentation the **BottomTabNavigator** is created directly inside the **createAppContainer**.
3. Create the **BottomTabNavigator** separately and then insert it in the **createAppContainer**.
4. There are two additional options passed down in c**reateBottomTabNavigator** - **defaultNavigationOptions** and **tabBarOptions**.
   - **tabBarOptions** takes the colors which you want in the bottom tabs to appear when they are active or inactive.
   - Experiment with this by changing the colors.

```
tabBarOptions={{
    activeTintColor: 'tomato',
     inactiveTintColor: 'gray',
    }}
  >
    <Tab.Screen name="Home" component={HomeScreen} />
    <Tab.Screen name="Settings" component={SettingsScreen} />
  </Tab.Navigator>
 </NavigationContainer>
);
```

OUTPUT:




```
tabBarOptions={{
    activeTintColor: 'green',
     inactiveTintColor: 'gray',
    }}
  >
    <Tab.Screen name="Home" component={HomeScreen} />
```

```
      <Tab.Screen name="Settings" component={SettingsScreen} />
    </Tab.Navigator>
  </NavigationContainer>
);
```

OUTPUT

5. **screenOptions** field returns some navigation options which are used in the app.
    - One of the navigation options is **tabBarIcon** that returns components to use as an icon in your bottom tab navigation.

```
export default class BottomTabNavigator extends Component {
  render() {
    return (
      <NavigationContainer>
        <Tab.Navigator
          screenOptions={({ route }) => ({
            tabBarIcon: ({ }) => {

              }

            }
          )}

        >
          <Tab.Screen name="Transaction" component={TransactionScreen} />
          <Tab.Screen name="Search" component={SearchScreen} />
        </Tab.Navigator>
      </NavigationContainer>
    );
  }
}
```

6. Set the **tabBarIcon** field inside screenOption.
    - **focused:** Returns true when the tab is tapped. Returns false otherwise.
    - **horizontal:** Returns true when the device is in landscape mode. Returns false if the device is in portrait mode.
    - **tintColor:** Returns the active set tintColor.

```
export default class BottomTabNavigator extends Component {
  render() {
    return (
      <NavigationContainer>
        <Tab.Navigator
          screenOptions={({ route }) => ({
            tabBarIcon: ({ focused, color, size }) => {


              if (route.name === "Transaction") {

              } else if (route.name === "Search") {

              }
```

7. Add the import the Ionicons from **react-native-vector-icons/Ionicons**
    - Use the **npm install react-native vector-icons** and import it.
    - In the **tabBarOptions** sets the active and inactive color along with the styles for the label.

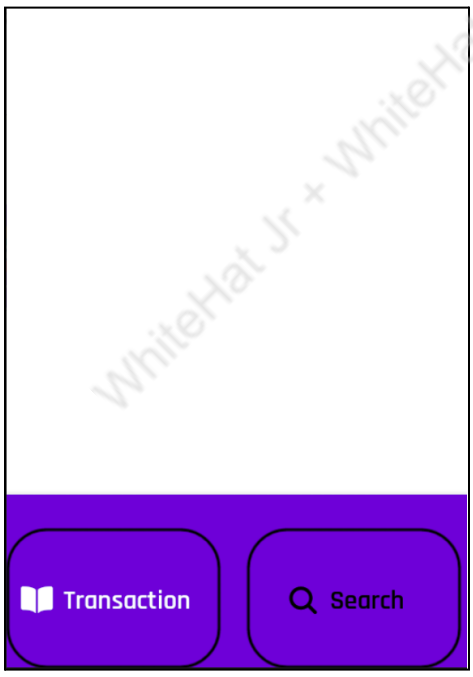
```
import React, { Component } from "react";
import { View } from "react-native";
import { NavigationContainer } from "@react-navigation/native";
import { createBottomTabNavigator } from
"@react-navigation/bottom-tabs";
import Ionicons from "react-native-vector-icons/Ionicons";

import TransactionScreen from "../screens/Transaction";
import SearchScreen from "../screens/Search";

const Tab = createBottomTabNavigator();

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

```
return (
  <NavigationContainer>
    <Tab.Navigator
      screenOptions={({ route }) => ({
        tabBarIcon: ({ focused, color, size }) => {
          let iconName;

          if (route.name === "Transaction") {
            iconName = "book";
          } else if (route.name === "Search") {
            iconName = "search";
          }

          // You can return any component that you like here!
          return (
            <Ionicons
              name={iconName}
              size={size}
              color={color}
            />
          );
        }
      })}
```

```
// You can return any component that you like here!
          return (
            <Ionicons
              name={iconName}
              size={size}
              color={color}

            />
          );
        }
      })}
      tabBarOptions={{
        activeTintColor: "#FFFFFF",
        inactiveTintColor: "black",
        style: {
          height: 130,
          borderTopWidth: 0,
          backgroundColor: "#5653d4"
```

```
        },
        labelStyle: {
          fontSize: 20,
          fontFamily: "Rajdhani_600SemiBold"
        },
        labelPosition: "beside-icon",
        tabStyle: {
          marginTop: 25,
          marginLeft: 10,
          marginRight: 10,
          borderRadius: 30,
          borderWidth: 2,
          alignItems: "center",
          justifyContent: "center",
          backgroundColor: "#5653d4"
        }      }
```

OUTPUT



8. Design the input screen used for book transactions in such a way it automatically populates the text inputs when QR code is scanned.
    - The input form should contain two **TextInput** components - which will accept book id and student id respectively.
    - There should be two scan buttons next to the text input to allow the user to scan the QR code and automatically populate the text input fields.
    - There should be a submit button to submit the information entered.

```
return (
  <View style={styles.container}>
    <View style={styles.lowerContainer}>
      <View style={styles.textinputContainer}>
        <TextInput
          style={styles.textinput}
          placeholder={"Book Id"}
          placeholderTextColor={"#FFFFFF"}
          value={bookId}
        />
        <TouchableOpacity
          style={styles.scanbutton}
        >
          <Text style={styles.scanbuttonText}>Scan</Text>
        </TouchableOpacity>
      </View>

      <View style={[styles.textinputContainer, { marginTop: 25 }]}>
        <TextInput
          style={styles.textinput}
          placeholder={"Student Id"}
          placeholderTextColor={"#FFFFFF"}
          value={studentId}
        />
        <TouchableOpacity
          style={styles.scanbutton}
        >
          <Text style={styles.scanbuttonText}>Scan</Text>
        </TouchableOpacity>
      </View>
    </View>
  </View>
);
```

```
const styles = StyleSheet.create({
 container: {
  flex: 1,
  backgroundColor: "#FFFFFF"
 },
 lowerContainer: {
  flex: 0.5,
  alignItems: "center"
 },
 textinputContainer: {
  borderWidth: 2,
  borderRadius: 10,
  flexDirection: "row",
  backgroundColor: "#9DFD24",
  borderColor: "#FFFFFF"
 },
 textinput: {
  width: "57%",
  height: 50,
  padding: 10,
  borderColor: "#FFFFFF",
  borderRadius: 10,
  borderWidth: 3,
  fontSize: 18,
  backgroundColor: "#5653D4",
  fontFamily: "Rajdhani_600SemiBold",
  color: "#FFFFFF"
 },
 scanbutton: {
  width: 100,
  height: 50,
  backgroundColor: "#9DFD24",
  borderTopRightRadius: 10,
  borderBottomRightRadius: 10,
  justifyContent: "center",
  alignItems: "center"
 },
 scanbuttonText: {
  fontSize: 24,
  color: "#0A0101",
  fontFamily: "Rajdhani_600SemiBold"
 }
```

```
});
```

9. Add a logo, background image and give our app some name.
    - To add the Background image and other images, we'll first need to import the **ImageBackground** and the image component from the react native.

```
import {
  View,
  StyleSheet,
  TextInput,
  TouchableOpacity,
  Text,
  ImageBackground,
  Image
} from "react-native";
```

```
const bgImage = require("../assets/background2.png");
const appIcon = require("../assets/appIcon.png");
const appName = require("../assets/appName.png");
```

```
return (
  <View style={styles.container}>
    <ImageBackground source={bgImage} style={styles.bgImage}>
      <View style={styles.upperContainer}>
        <Image source={appIcon} style={styles.appIcon} />
        <Image source={appName} style={styles.appName} />
      </View>

      <View style={styles.lowerContainer}>
        <View style={styles.textinputContainer}>
          <TextInput
            style={styles.textinput}
            placeholder={"Book Id"}
            placeholderTextColor={"#FFFFFF"}
            value={bookId}
          />
          <TouchableOpacity
            style={styles.scanbutton}
            onPress={() => this.getCameraPermissions("bookId")}
```

```
        >
          <Text style={styles.scanbuttonText}>Scan</Text>
        </TouchableOpacity>
      </View>
      <View style={[styles.textinputContainer, { marginTop: 25 }]}>
        <TextInput
          style={styles.textinput}
          placeholder={"Student Id"}
          placeholderTextColor={"#FFFFFF"}
          value={studentId}
        />
        <TouchableOpacity
          style={styles.scanbutton}
          onPress={() => this.getCameraPermissions("studentId")}
        >
          <Text style={styles.scanbuttonText}>Scan</Text>
        </TouchableOpacity>
      </View>
    </View>
  </ImageBackground>
 </View>
);
```

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: "#FFFFFF"
  },
  bgImage: {
    flex: 1,
    resizeMode: "cover",
    justifyContent: "center"
  },
  upperContainer: {
    flex: 0.5,
    justifyContent: "center",
    alignItems: "center"
  },
  appIcon: {
    width: 200,
    height: 200,
    resizeMode: "contain",
    marginTop: 80
  },
  appName: {
    width: 80,
    height: 80,
    resizeMode: "contain"
  },
  lowerContainer: {
    flex: 0.5,
    alignItems: "center"
  },
```

10. Add two more states:
    - **scannedBookid:** It will store the scanned book id data.
    - **scannedStudentid:** It will store the scanned student id data.

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

11. Call the **getCameraPermissions** function when any of the scan buttons are clicked and pass which button is clicked as an argument. Inside **getCameraPermissions** function, Set the **domState** depending on which button is clicked.

```
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
  });
};
```

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

    <View style={styles.lowerContainer}>
      <View style={styles.textinputContainer}>
        <TextInput
          style={styles.textinput}
          placeholder={"Book Id"}
          placeholderTextColor={"#FFFFFF"}
          value={bookId}
        />
        <TouchableOpacity
          style={styles.scanbutton}
          onPress={() => this.getCameraPermissions("bookId")}
        >
          <Text style={styles.scanbuttonText}>Scan</Text>
        </TouchableOpacity>
      </View>
      <View style={[styles.textinputContainer, { marginTop: 25 }]}>
        <TextInput
          style={styles.textinput}
          placeholder={"Student Id"}
          placeholderTextColor={"#FFFFFF"}
          value={studentId}
        />
        <TouchableOpacity
          style={styles.scanbutton}
          onPress={() => this.getCameraPermissions("studentId")}
        >
          <Text style={styles.scanbuttonText}>Scan</Text>
        </TouchableOpacity>
      </View>
    </View>

  </View>
);
```

12. Set the value for text input to these states which we have scanned.

```
handleBarCodeScanned = async ({ type, data }) => {
  const { domState } = this.state;

  if (domState === "bookId") {
    this.setState({
      bookId: data,
      domState: "normal",
      scanned: true
    });
  } else if (domState === "studentId") {
    this.setState({
      studentId: data,
      domState: "normal",
      scanned: true
    });
  }
};
```

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

    <View style={styles.lowerContainer}>
      <View style={styles.textinputContainer}>
        <TextInput
          style={styles.textinput}
          placeholder={"Book Id"}
          placeholderTextColor={"#FFFFFF"}
          value={bookId}
        />
        <TouchableOpacity
          style={styles.scanbutton}
          onPress={() => this.getCameraPermissions("bookId")}
        >
          <Text style={styles.scanbuttonText}>Scan</Text>
        </TouchableOpacity>
      </View>
      <View style={[styles.textinputContainer, { marginTop: 25 }]}>
        <TextInput
          style={styles.textinput}
          placeholder={"Student Id"}
          placeholderTextColor={"#FFFFFF"}
          value={studentId}
        />
        <TouchableOpacity
          style={styles.scanbutton}
          onPress={() => this.getCameraPermissions("studentId")}
        >
          <Text style={styles.scanbuttonText}>Scan</Text>
        </TouchableOpacity>
      </View>
    </View>

  </View>
);
```

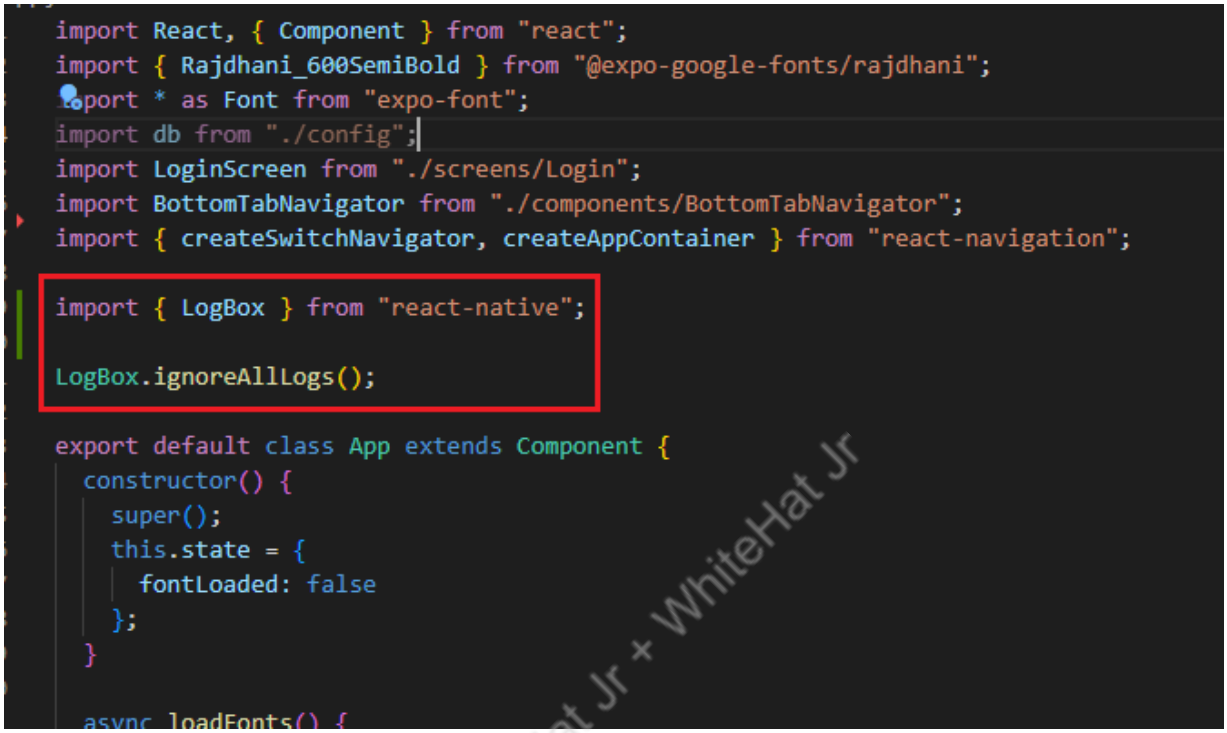
13. **domState** does not have "clicked" as a value ever. Display the barcode scanner as soon as the button changes from "normal". Change the condition to render **BarCodeScanner** component accordingly.

```
render() {
  const { bookId, studentId, domState, scanned } = this.state;
  if (domState !== "normal") {
    return (
      <BarCodeScanner
        onBarCodeScanned={scanned ? undefined : this.handleBarCodeScanned}
        style={StyleSheet.absoluteFillObject}
      />
    );
  }
```

14. Run and test the code.

15. We can see a few warnings while testing our app. We can hide these warnings by adding the LogBox Component to our project.

These warnings can be hidden by using **LogBox.ignoreAllLogs()**.



```
import React, { Component } from "react";
import { Rajdhani_600SemiBold } from "@expo-google-fonts/rajdhani";
import * as Font from "expo-font";
import db from "./config";
import LoginScreen from "./screens/Login";
import BottomTabNavigator from "./components/BottomTabNavigator";
import { createSwitchNavigator, createAppContainer } from "react-navigation";

import { LogBox } from "react-native";

LogBox.ignoreAllLogs();

export default class App extends Component {
  constructor() {
    super();
    this.state = {
      fontLoaded: false
    };
  }

  async loadFonts() {
```


What's NEXT?

In the next class, we will be learning what to do with the data that we have just collected. We will also learn more about databases and how to use them in our app.

EXTEND YOUR KNOWLEDGE

1. Tab Navigation: https://reactnavigation.org/docs/tab-based-navigation/