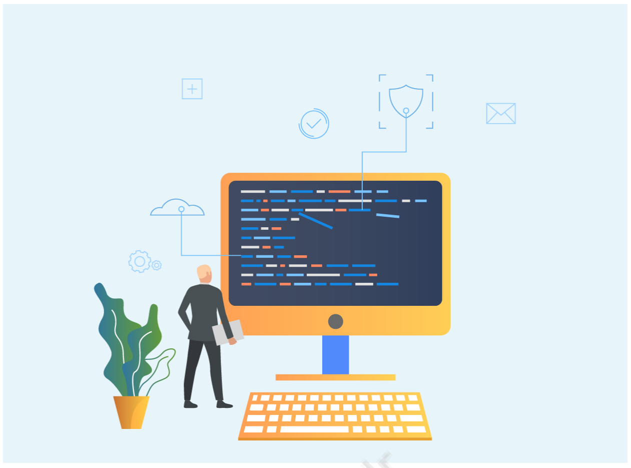
CS-PRO-C72(V3)

AVOIDING KEYBOARD OVERLAP AND TOASTS



What is our GOAL for this MODULE?

In this class, we learned to explore scenarios where how manually entering the text in a text box is beneficial. Also, we solve the issue of the overlapping keyboard on text. And finally, display a transaction message to the users using toasts or alerts.

What did we ACHIEVE in the class TODAY?

- Made the text box editable.
- Avoided keyboard layout overlap with the text box.
- Displayed a Transaction message when a transaction is completed.

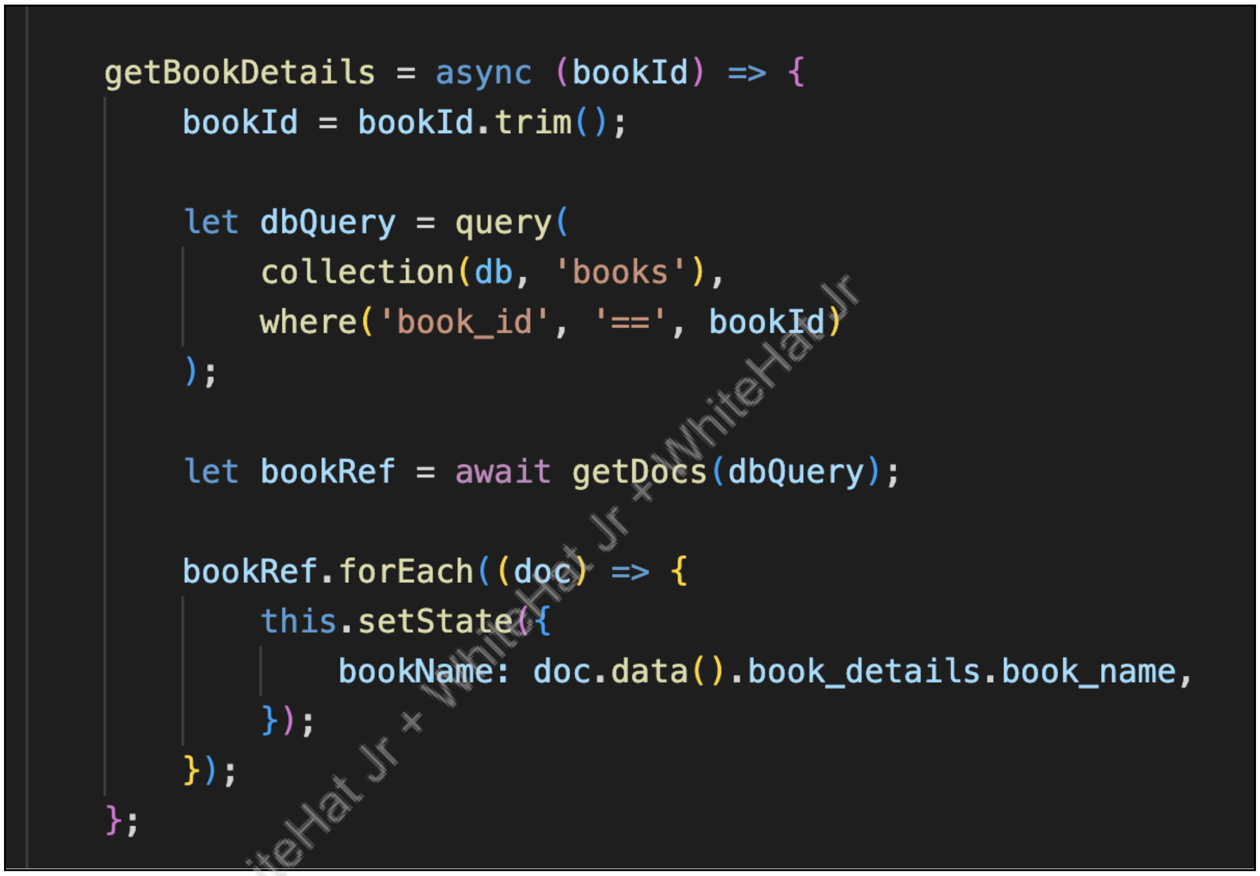
Which CONCEPTS/CODING BLOCKS did we cover today?

- Avoiding Keyboard Overlap and Toasts.
- To make the Text box editable.

How did we DO the activities?

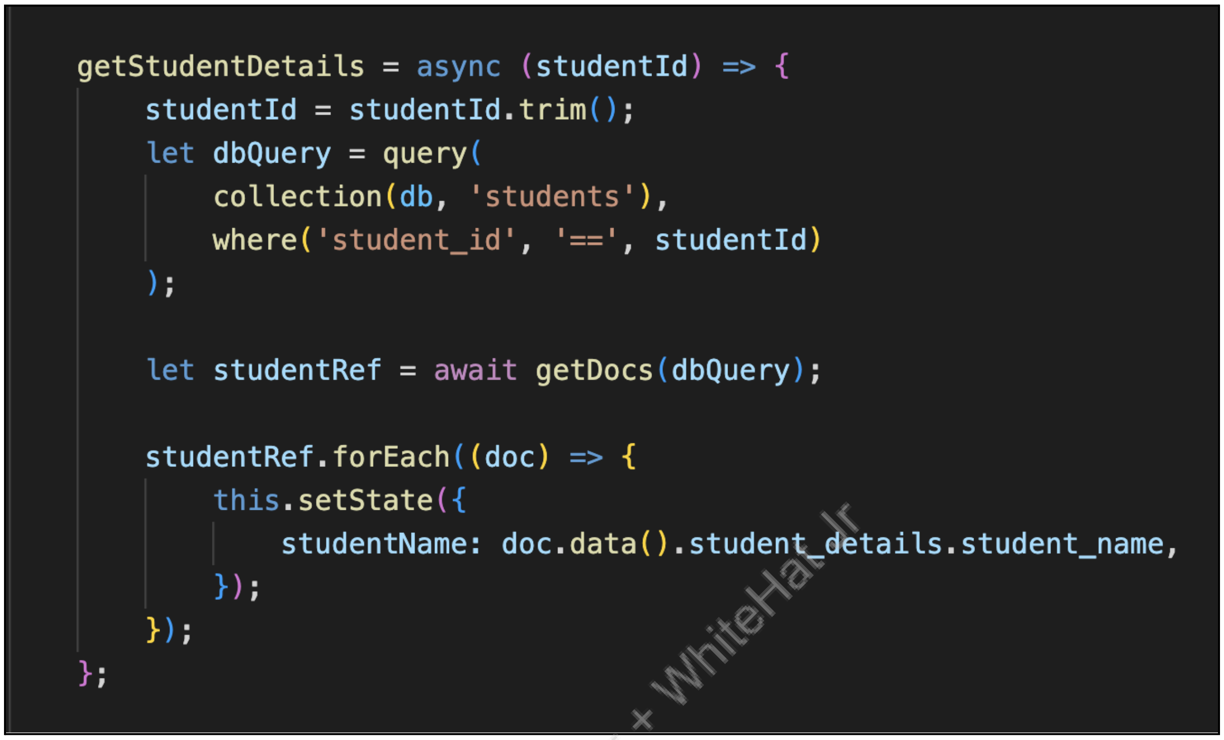
To issue a book to a student, you first needed to get the details of the student and the book from the database, after which you issue or return the book. Also, towards the end, you show the Alert/Toast message when the transaction is done.

1. Write the **getBookDetails()** function to get the book details.



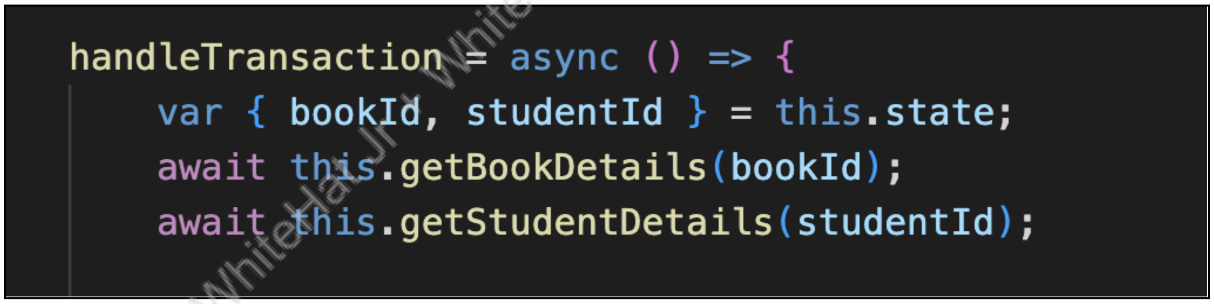
```
getBookDetails = async (bookId) => {
    bookId = bookId.trim();

    let dbQuery = query(
        collection(db, 'books'),
        where('book_id', '==', bookId)
    );

    let bookRef = await getDocs(dbQuery);

    bookRef.forEach((doc) => {
        this.setState({
            bookName: doc.data().book_details.book_name,
        });
    });
};
```

2. Write the **getStudentDetails()** to get the student details.



```
getStudentDetails = async (studentId) => {
    studentId = studentId.trim();
    let dbQuery = query(
        collection(db, 'students'),
        where('student_id', '==', studentId)
    );

    let studentRef = await getDocs(dbQuery);

    studentRef.forEach((doc) => {
        this.setState({
            studentName: doc.data().student_details.student_name,
        });
    });
};
```

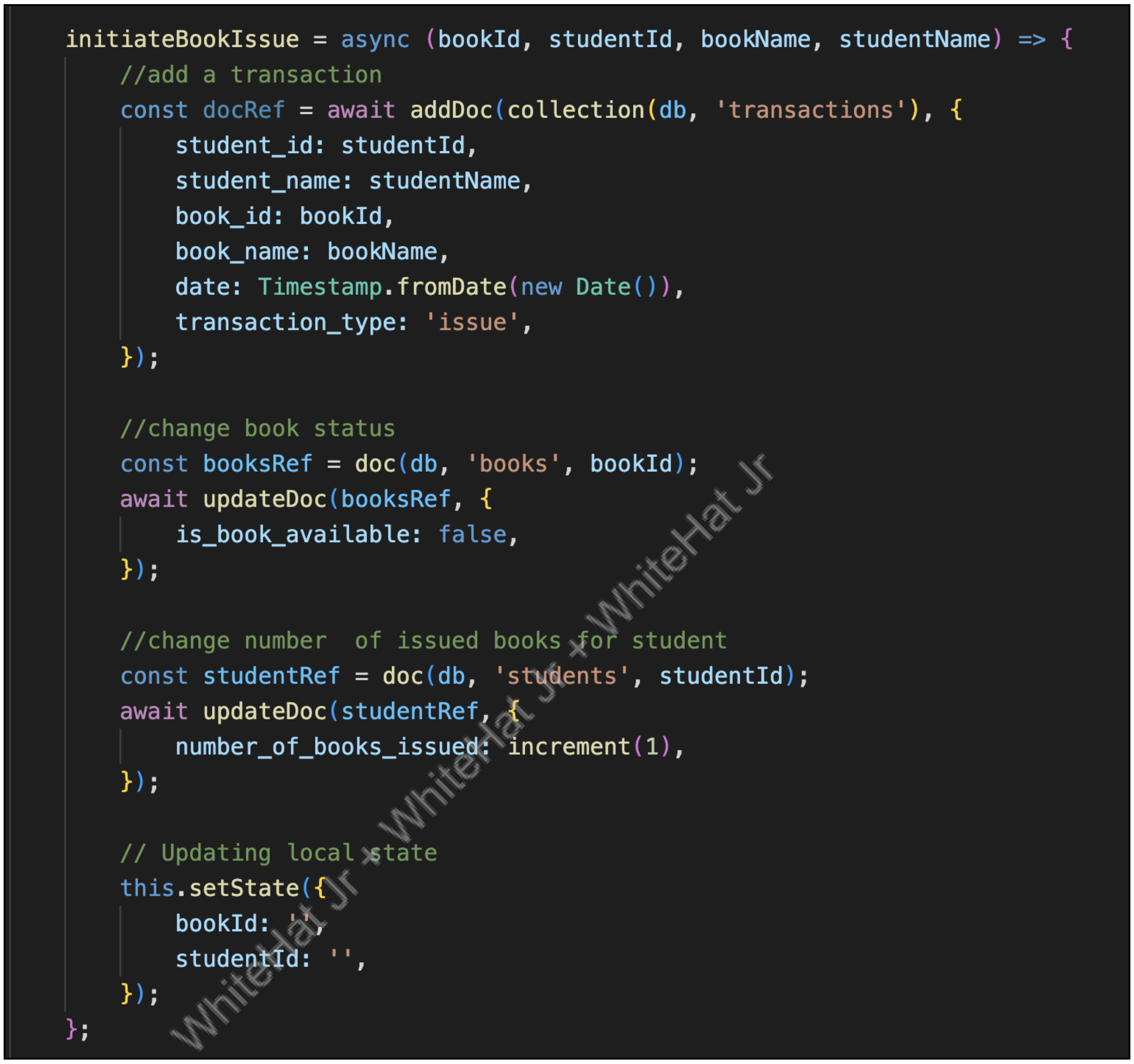
3. Call these functions inside the **handleTransaction()** function.



```
handleTransaction = async () => {
    var { bookId, studentId } = this.state;
    await this.getBookDetails(bookId);
    await this.getStudentDetails(studentId);
```

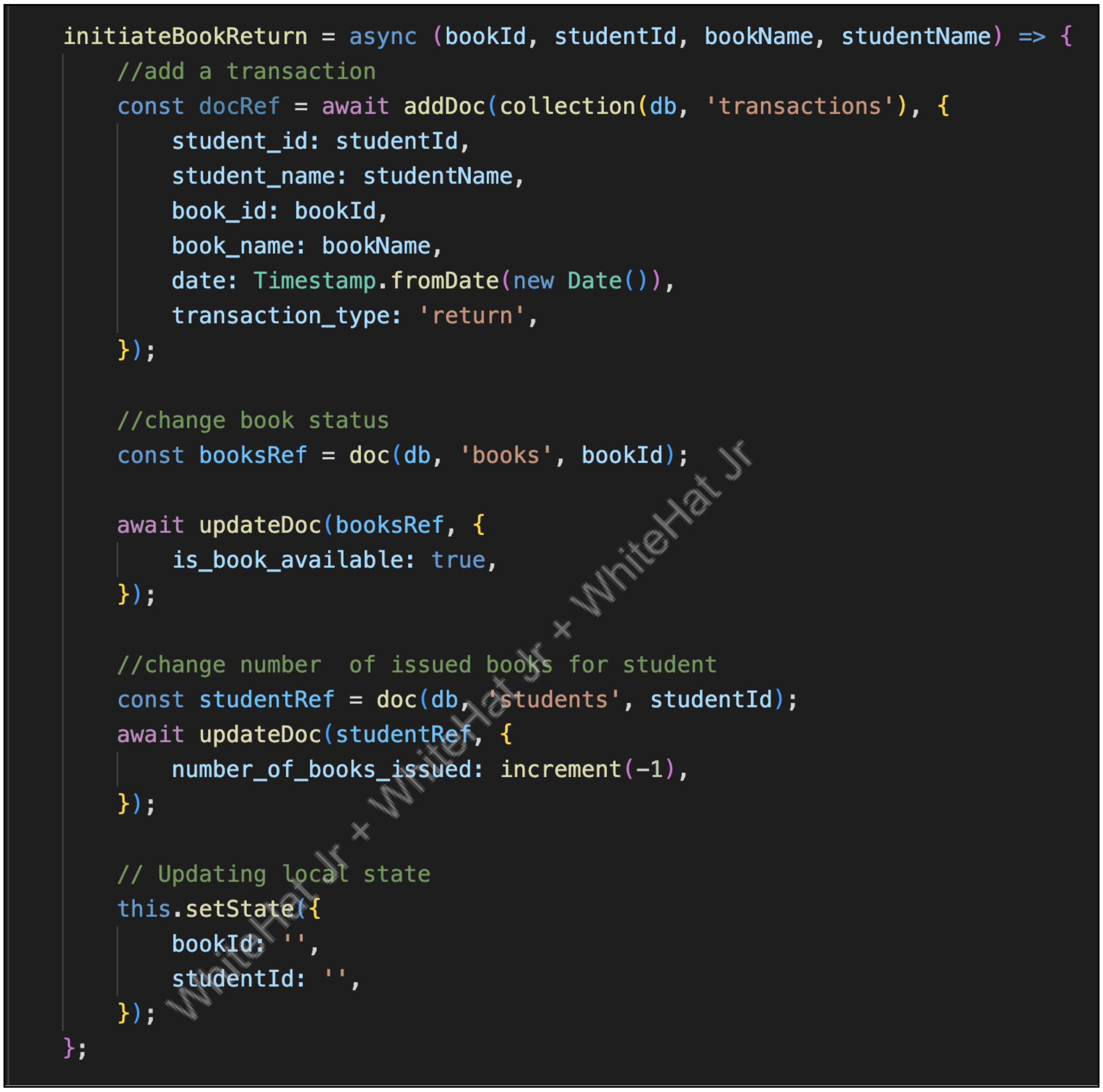
4. Write the **initiateBookIssue()** function**.** This function will take four parameters as follows:

    - **StudentId**,
    - **Student Name**,
    - **bookId,**
    - and **bookName.**

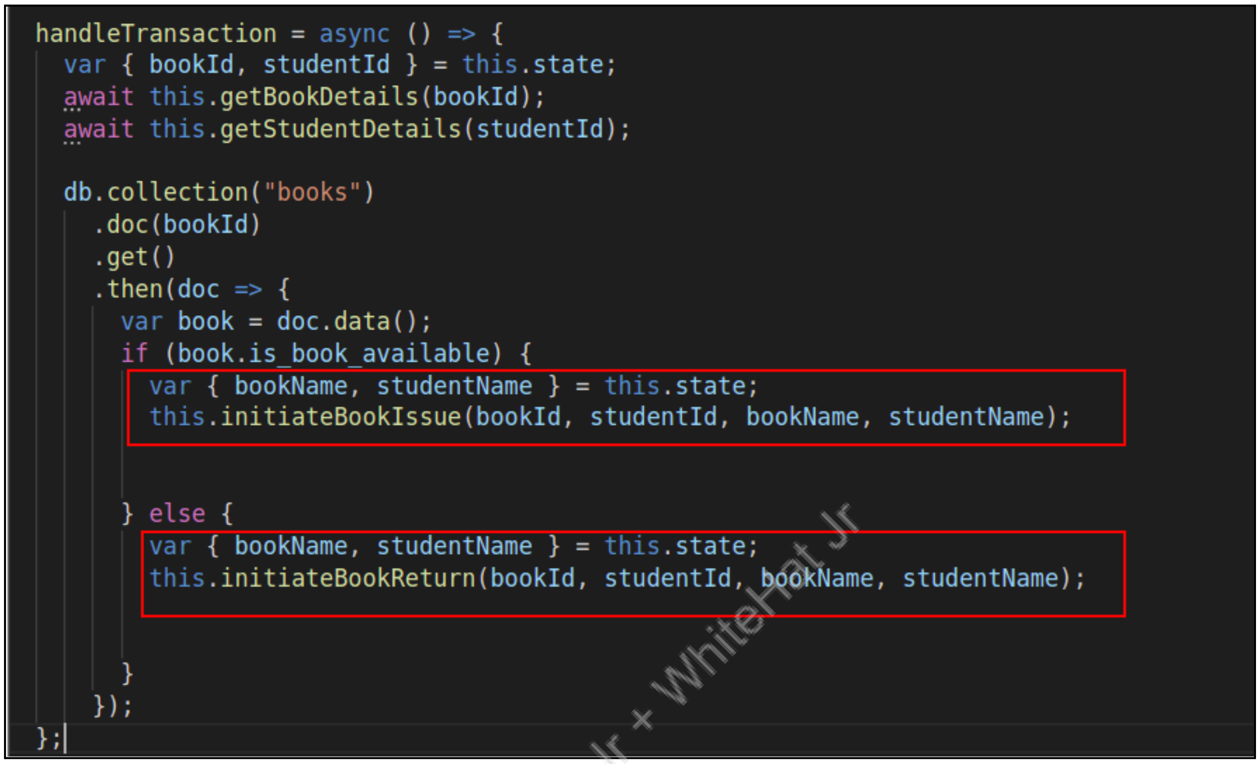


```
initiateBookIssue = async (bookId, studentId, bookName, studentName) => {
    //add a transaction
    const docRef = await addDoc(collection(db, 'transactions'), {
        student_id: studentId,
        student_name: studentName,
        book_id: bookId,
        book_name: bookName,
        date: Timestamp.fromDate(new Date()),
        transaction_type: 'issue',
    });

    //change book status
    const booksRef = doc(db, 'books', bookId);
    await updateDoc(booksRef, {
        is_book_available: false,
    });

    //change number  of issued books for student
    const studentRef = doc(db, 'students', studentId);
    await updateDoc(studentRef, {
        number_of_books_issued: increment(1),
    });

    // Updating local state
    this.setState({
        bookId: '',
        studentId: '',
    });
};
```

5. Write the **initiateBookReturn()** function**.** Here, the **transaction_type** would be set as **return**. And **is_book_available** would be set as **true.** Update the number of books issued for a student, and update the **bookId** and **studentId.**



```
initiateBookReturn = async (bookId, studentId, bookName, studentName) => {
    //add a transaction
    const docRef = await addDoc(collection(db, 'transactions'), {
        student_id: studentId,
        student_name: studentName,
        book_id: bookId,
        book_name: bookName,
        date: Timestamp.fromDate(new Date()),
        transaction_type: 'return',
    });

    //change book status
    const booksRef = doc(db, 'books', bookId);

    await updateDoc(booksRef, {
        is_book_available: true,
    });

    //change number  of issued books for student
    const studentRef = doc(db, 'students', studentId);
    await updateDoc(studentRef, {
        number_of_books_issued: increment(-1),
    });

    // Updating local state
    this.setState({
        bookId: '',
        studentId: '',
    });
};
```

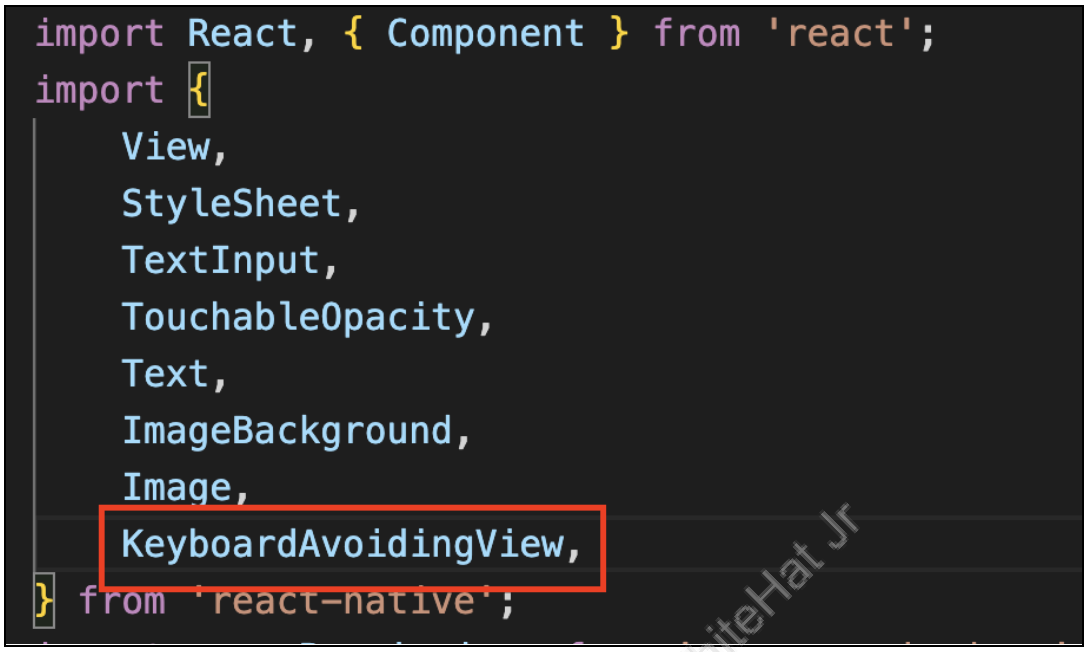


```
handleTransaction = async () => {
  var { bookId, studentId } = this.state;
  await this.getBookDetails(bookId);
  await this.getStudentDetails(studentId);

  db.collection("books")
    .doc(bookId)
    .get()
    .then(doc => {
      var book = doc.data();
      if (book.is_book_available) {
        var { bookName, studentName } = this.state;
        this.initiateBookIssue(bookId, studentId, bookName, studentName);

      } else {
        var { bookName, studentName } = this.state;
        this.initiateBookReturn(bookId, studentId, bookName, studentName);

      }
    });
};
```

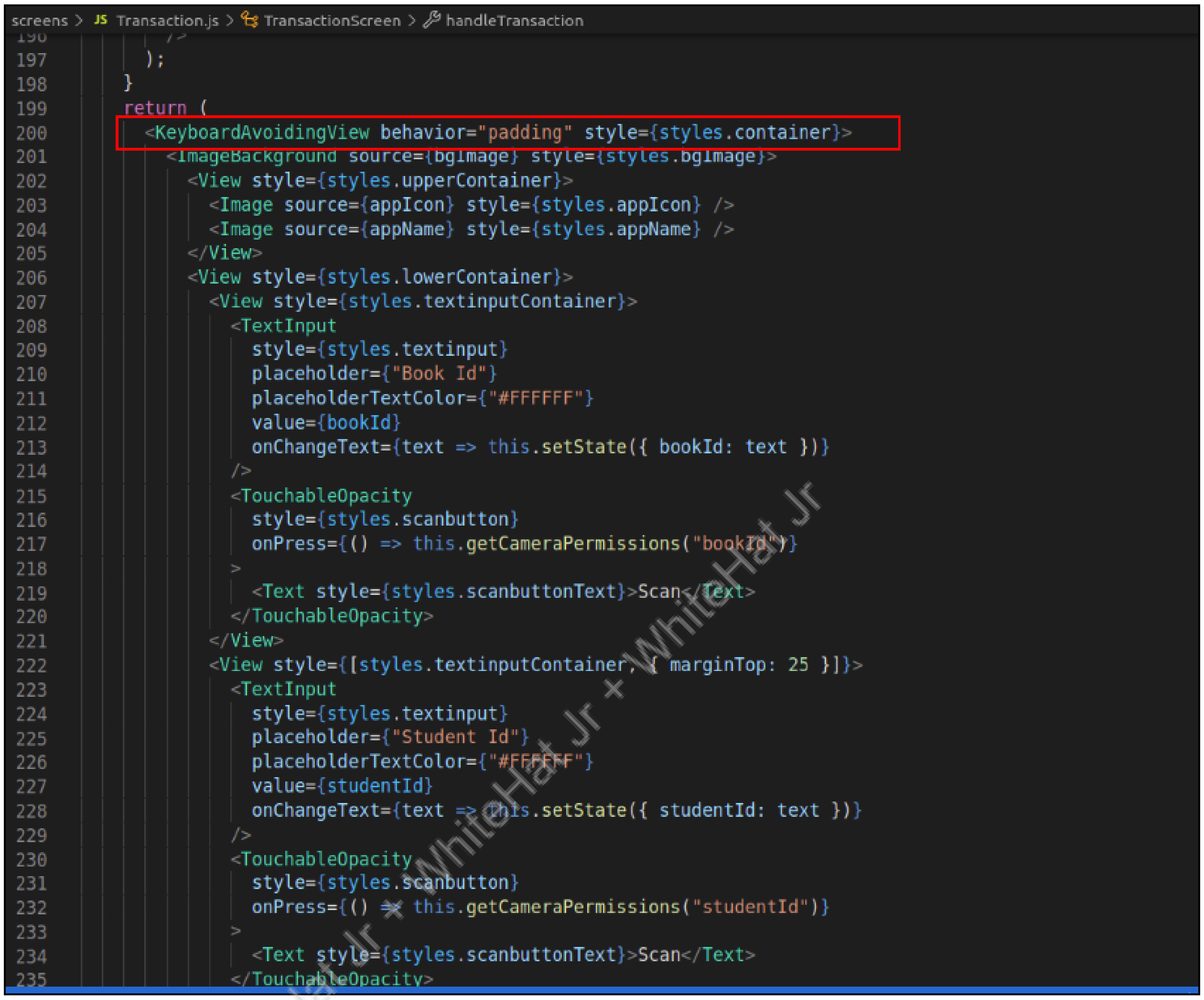
We have now written the code to get the details of the book and the student to issue or return the book to the student.

From here on ahead, we fix the issue of the overlapping keyboard on the text input boxes.

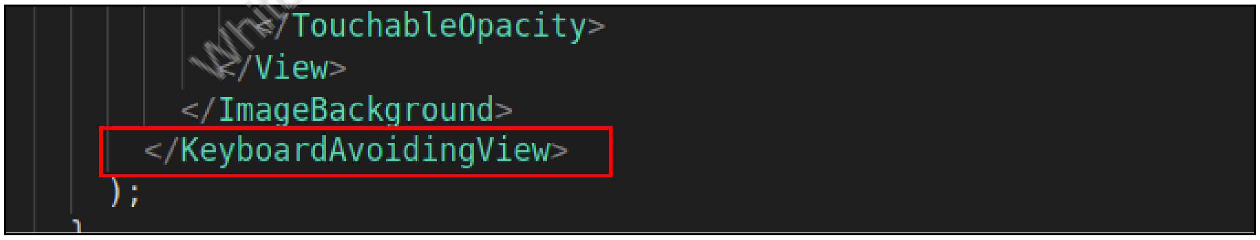
6. Use the **KeyboardAvoidingView** to solve the problem of the keyboard overlapping the **TextInput** boxes.



```
import React, { Component } from 'react';
import {
    View,
    StyleSheet,
    TextInput,
    TouchableOpacity,
    Text,
    ImageBackground,
    Image,
    KeyboardAvoidingView,
} from 'react-native';
```

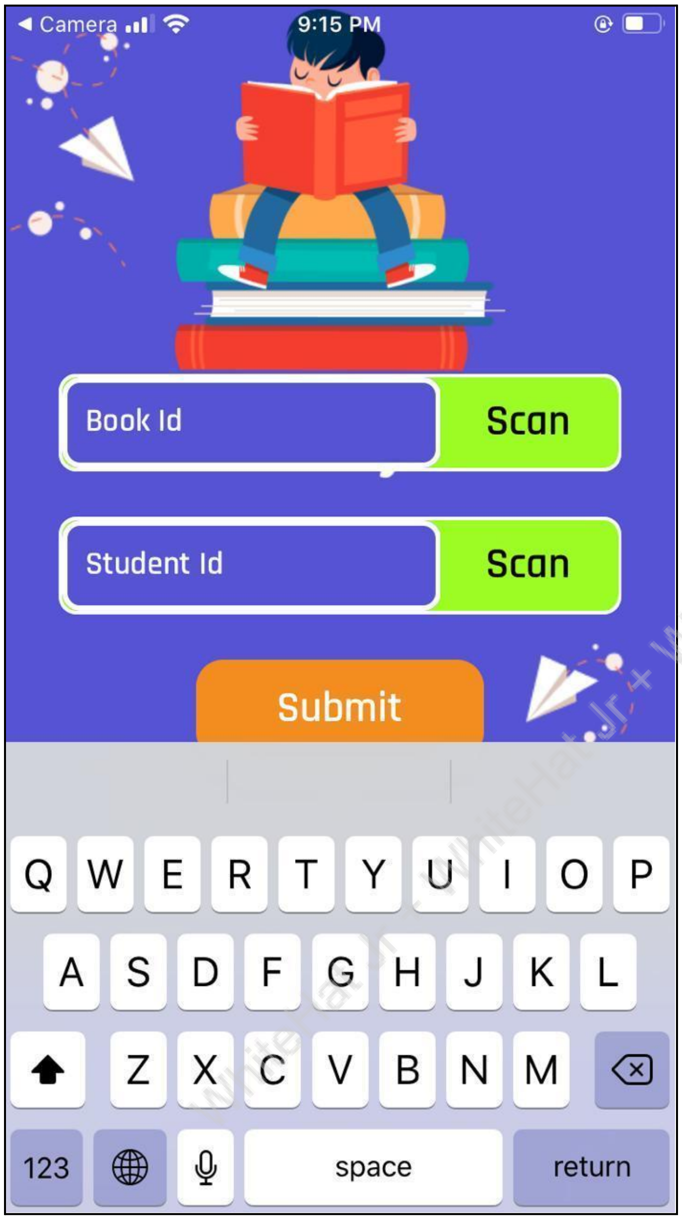


```
screens > JS Transaction.js > TransactionScreen > handleTransaction
          );
        }
        return (
          <KeyboardAvoidingView behavior="padding" style={styles.container}>
            <ImageBackground source={bgImage} style={styles.bgImage}>
              <View style={styles.upperContainer}>
                <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}
                    onChangeText={text => this.setState({ bookId: text })}
                  />
                  <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}
                    onChangeText={text => this.setState({ studentId: text })}
                  />
                  <TouchableOpacity
                    style={styles.scanbutton}
                    onPress={() => this.getCameraPermissions("studentId")}
                  >
                    <Text style={styles.scanbuttonText}>Scan</Text>
                  </TouchableOpacity>
```

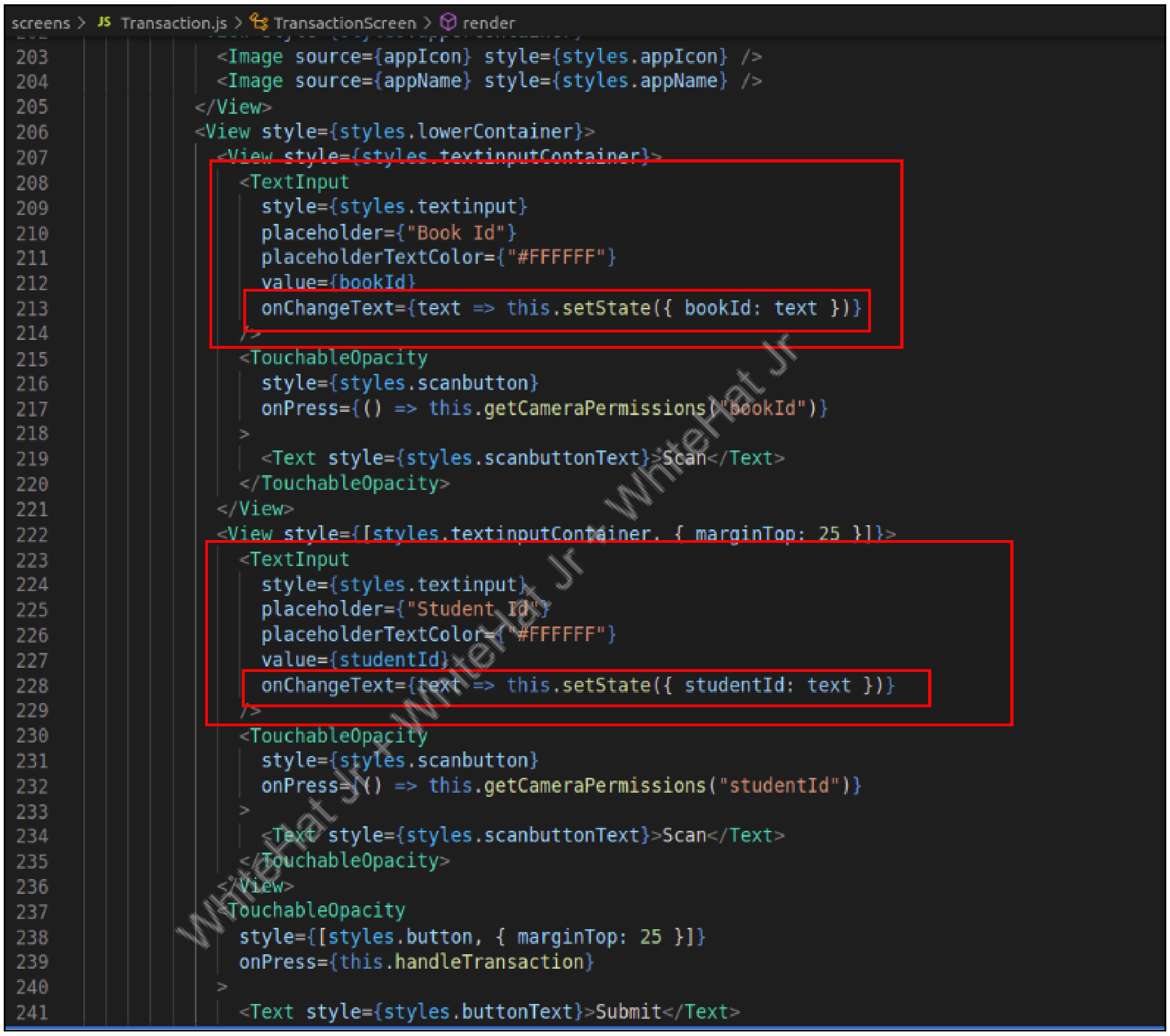


```
                  </TouchableOpacity>
                </View>
              </ImageBackground>
            </KeyboardAvoidingView>
          );
        }
```

Output:


Camera
9:15 PM
Book Id
Scan
Student Id
Scan
Submit
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
123
space
return

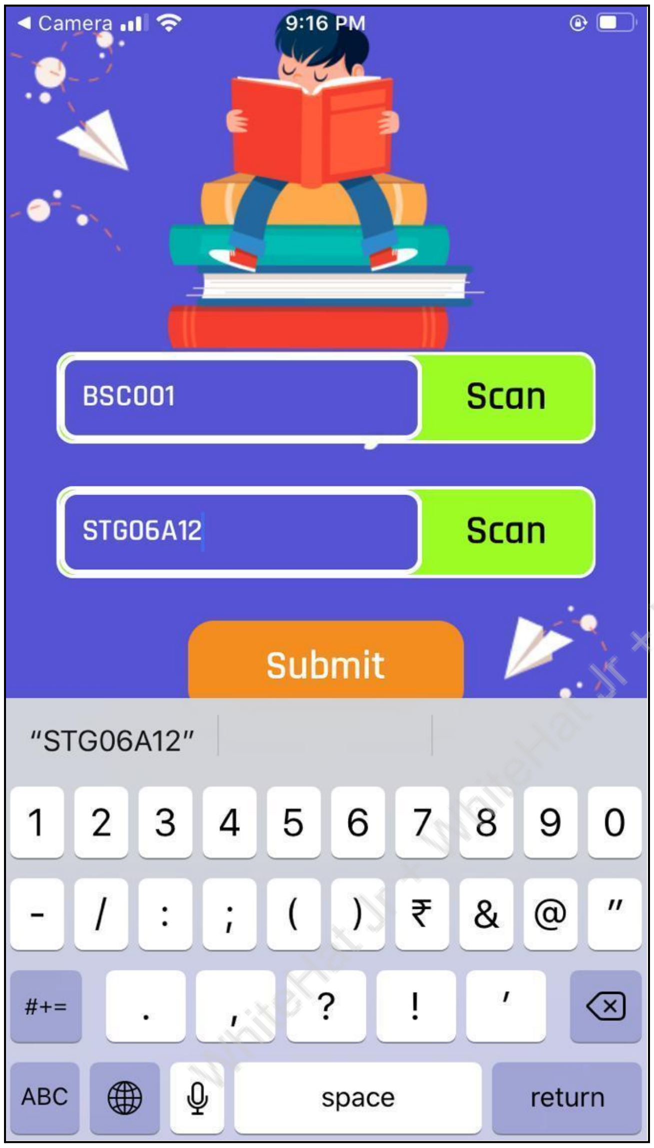
7. Get the text typed by the user in **TextInput** as the default argument, and use the **onChange** prop to set the values for student ID and book ID.



screens > Transaction.js > TransactionScreen > render

```
              <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}
                  onChangeText={text => this.setState({ bookId: text })}
                />
                <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}
                  onChangeText={text => this.setState({ studentId: text })}
                />
                <TouchableOpacity
                  style={styles.scanbutton}
                  onPress={() => this.getCameraPermissions("studentId")}
                >
                  <Text style={styles.scanbuttonText}>Scan</Text>
                </TouchableOpacity>
              </View>
              <TouchableOpacity
                style={[styles.button, { marginTop: 25 }]}
                onPress={this.handleTransaction}
              >
                <Text style={styles.buttonText}>Submit</Text>
```

Output:

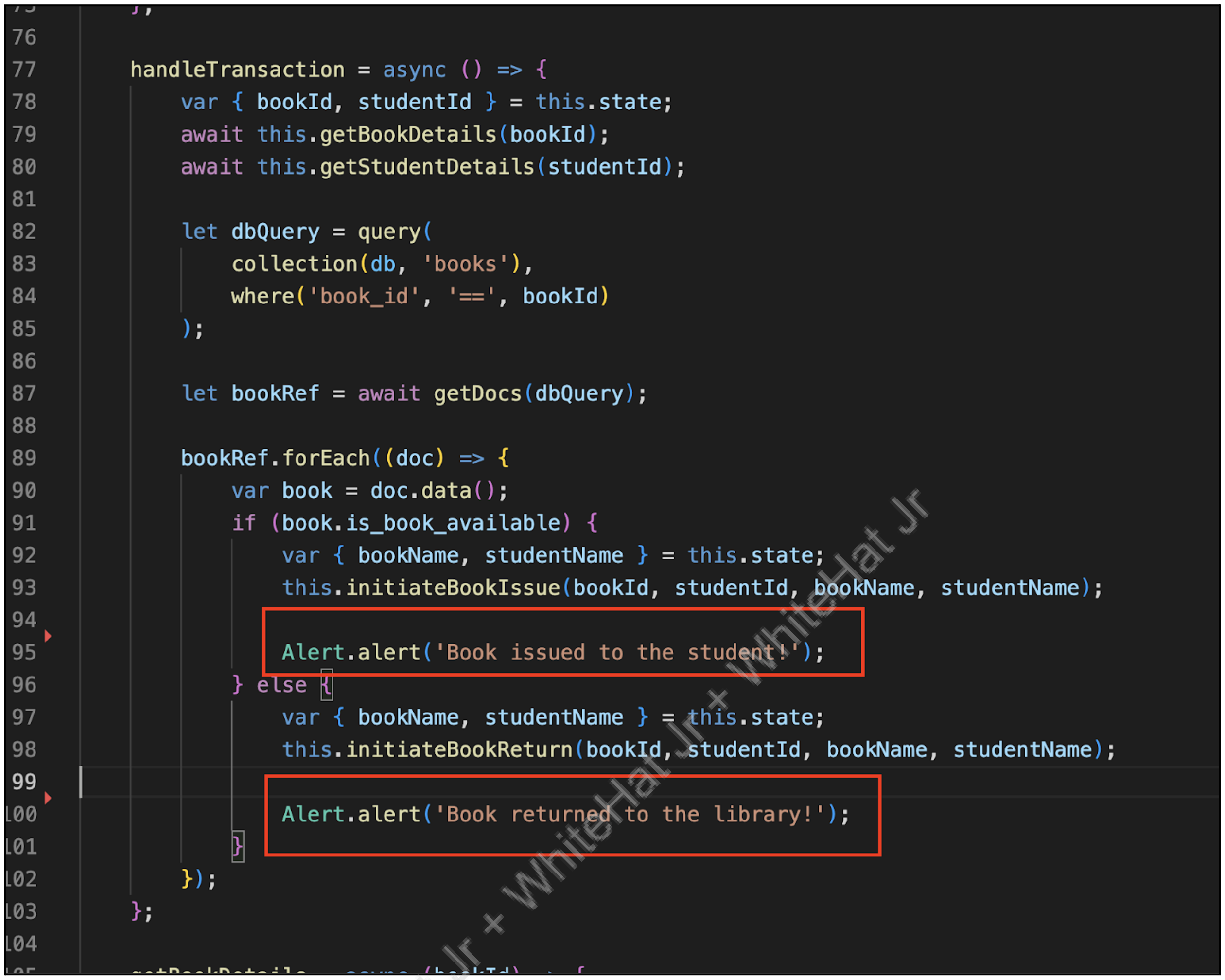


Now the text box is editable.

8. Display a message to the user when a transaction (issue or return) is completed, using the **ToastAndroid** Component to display a **Toast** Message.

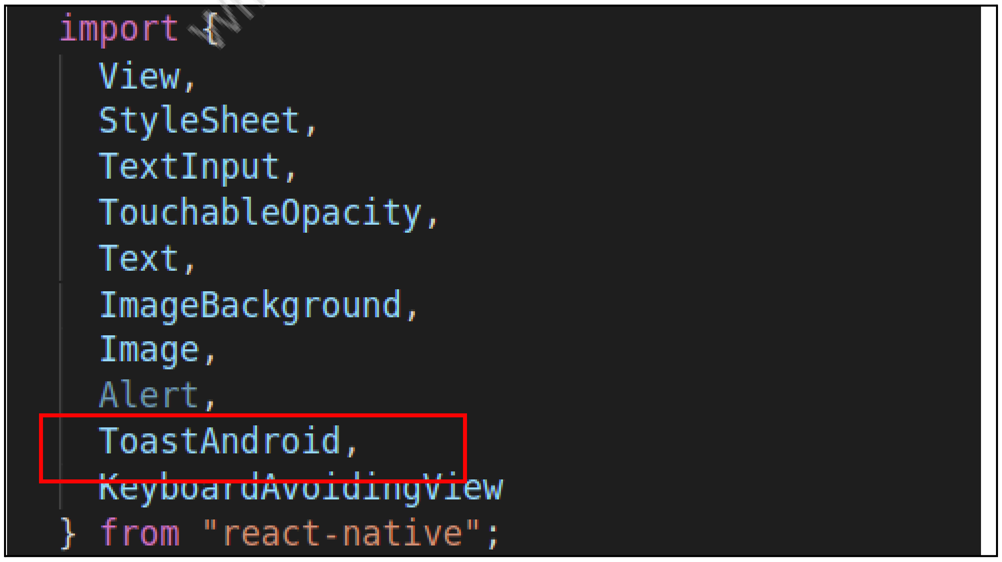
   Note: - **ToastAndroid** can only be used for android devices and not for iOS. For iOS users, use **Alert.**

   To code using **Alert**:

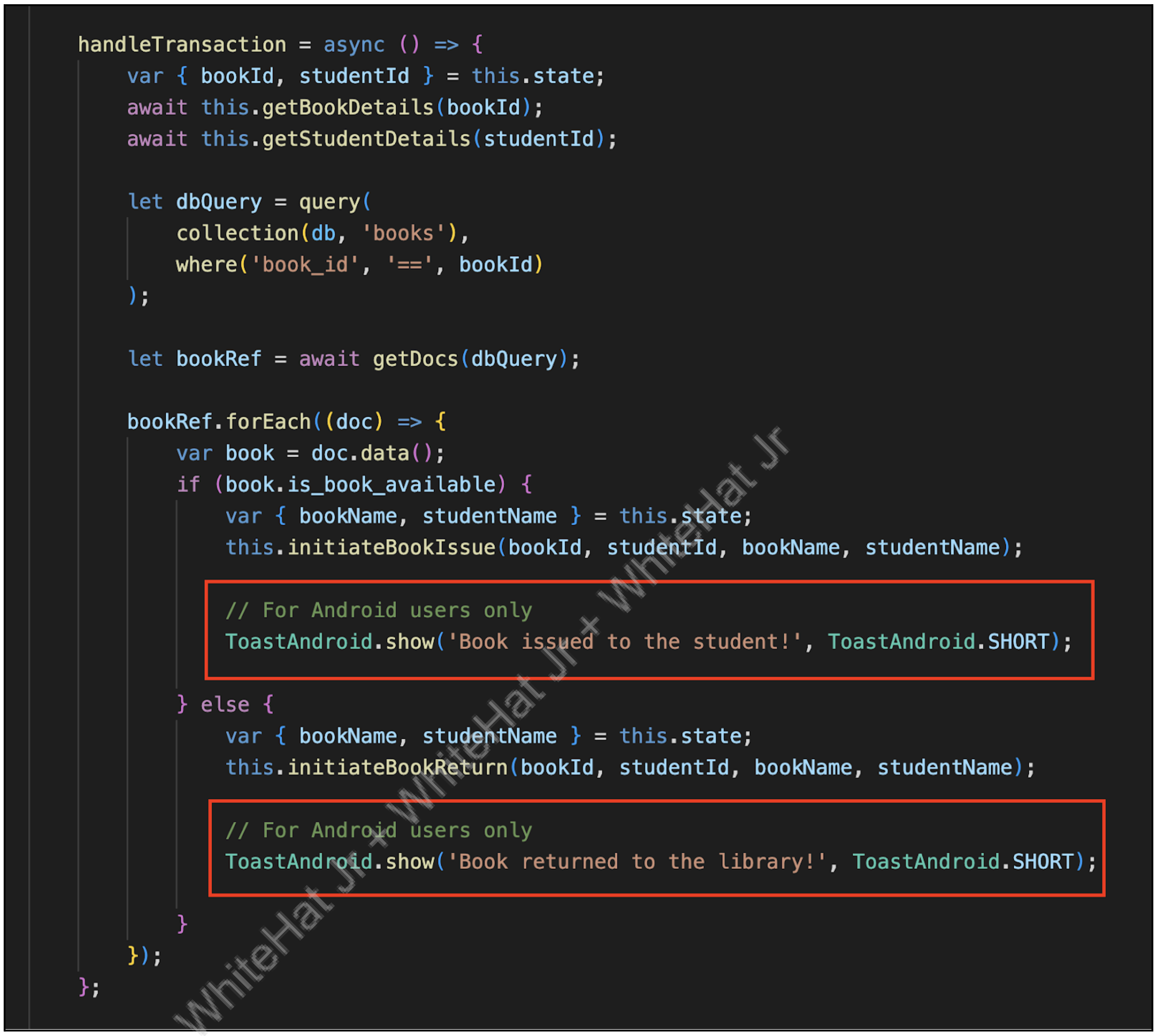


```
    handleTransaction = async () => {
        var { bookId, studentId } = this.state;
        await this.getBookDetails(bookId);
        await this.getStudentDetails(studentId);

        let dbQuery = query(
            collection(db, 'books'),
            where('book_id', '==', bookId)
        );

        let bookRef = await getDocs(dbQuery);

        bookRef.forEach((doc) => {
            var book = doc.data();
            if (book.is_book_available) {
                var { bookName, studentName } = this.state;
                this.initiateBookIssue(bookId, studentId, bookName, studentName);

                Alert.alert('Book issued to the student!');
            } else {
                var { bookName, studentName } = this.state;
                this.initiateBookReturn(bookId, studentId, bookName, studentName);

                Alert.alert('Book returned to the library!');
            }
        });
    };
```

To code using **ToastAndroid**:

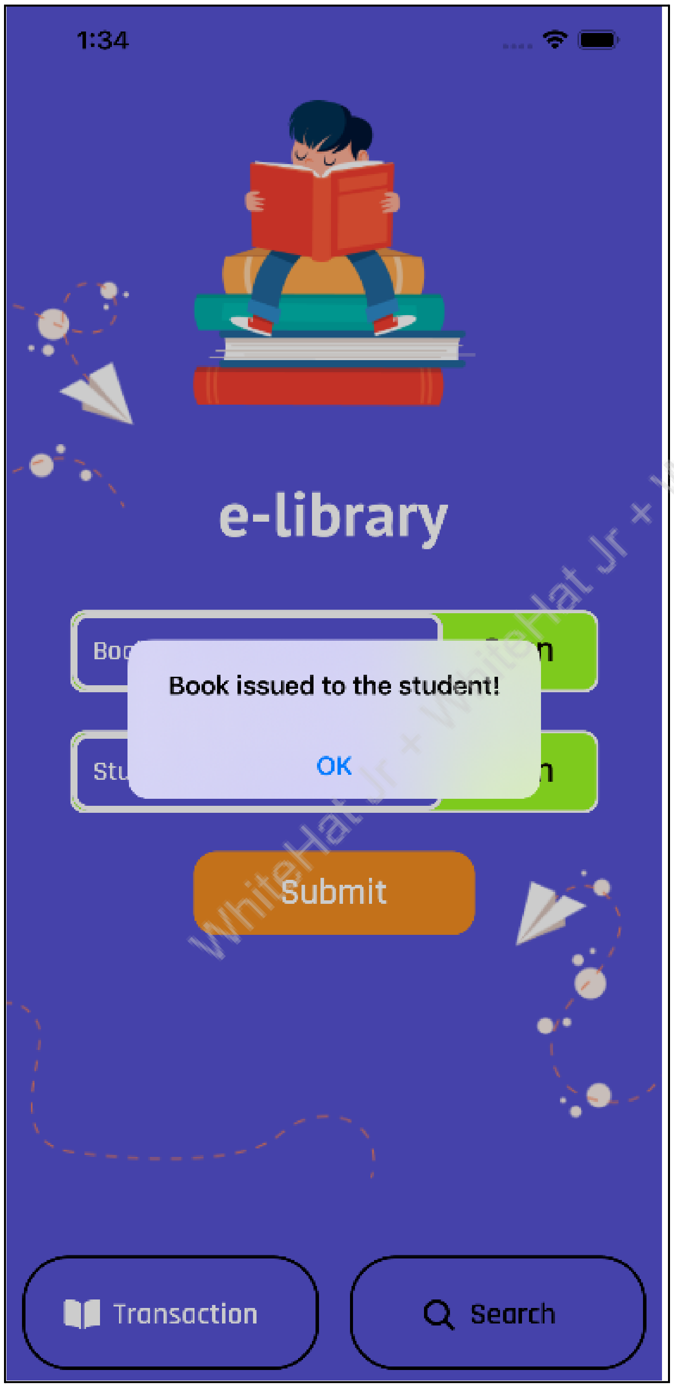


```
import {
  View,
  StyleSheet,
  TextInput,
  TouchableOpacity,
  Text,
  ImageBackground,
  Image,
  Alert,
  ToastAndroid,
  KeyboardAvoidingView
} from "react-native";
```



```
handleTransaction = async () => {
    var { bookId, studentId } = this.state;
    await this.getBookDetails(bookId);
    await this.getStudentDetails(studentId);

    let dbQuery = query(
        collection(db, 'books'),
        where('book_id', '==', bookId)
    );

    let bookRef = await getDocs(dbQuery);

    bookRef.forEach((doc) => {
        var book = doc.data();
        if (book.is_book_available) {
            var { bookName, studentName } = this.state;
            this.initiateBookIssue(bookId, studentId, bookName, studentName);

            // For Android users only
            ToastAndroid.show('Book issued to the student!', ToastAndroid.SHORT);

        } else {
            var { bookName, studentName } = this.state;
            this.initiateBookReturn(bookId, studentId, bookName, studentName);

            // For Android users only
            ToastAndroid.show('Book returned to the library!', ToastAndroid.SHORT);

        }
    });
};
```

Output:


Book issued to the student!

What's NEXT?

In the next class, we will learn how to make queries to a Firebase Database. Check for student eligibility and book eligibility before issuing/returning a book using firebase queries.

EXTEND YOUR KNOWLEDGE

1. Learn more about ToastAndroid: https://reactnative.dev/docs/toastandroid