INTRODUCTION TO FIRESTORE

What is our GOAL for this MODULE?

In this class we designed the collections and documents in the firestore database for the e-library app and programmed the submit button so that a book is issued or returned to a student as a library transaction.

What did we ACHIEVE in the class TODAY?

- Design e-library app database in firestore.
- Write code to issue/return a book by updating the database.

Which CONCEPTS/CODING BLOCKS did we cover today?

- Usage of firestore database
- Connecting firestore database to our app

How did we DO the activities?

1. Create a submit button and style it.

```
<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>
        <TouchableOpacity
          style={[styles.button, { marginTop: 25 }]}
        >
          <Text style={styles.buttonText}>Submit</Text>
        </TouchableOpacity>
      </View>
    </ImageBackground>
  </View>
);
```

```
scanbutton: {
  width: 100,
  height: 50,
  backgroundColor: "#9DFD24",
  borderTopRightRadius: 10,
  borderBottomRightRadius: 10,
  justifyContent: "center",
  alignItems: "center"
},
scanbuttonText: {
  fontSize: 24,
  color: "#0A0101",
  fontFamily: "Rajdhani_600SemiBold"
},
button: {
```

```
  width: "43%",
  height: 55,
  justifyContent: "center",
  alignItems: "center",
  backgroundColor: "#F48D20",
  borderRadius: 15
},
buttonText: {
  fontSize: 24,
  color: "#FFFFFF",
  fontFamily: "Rajdhani_600SemiBold"
}
});
```

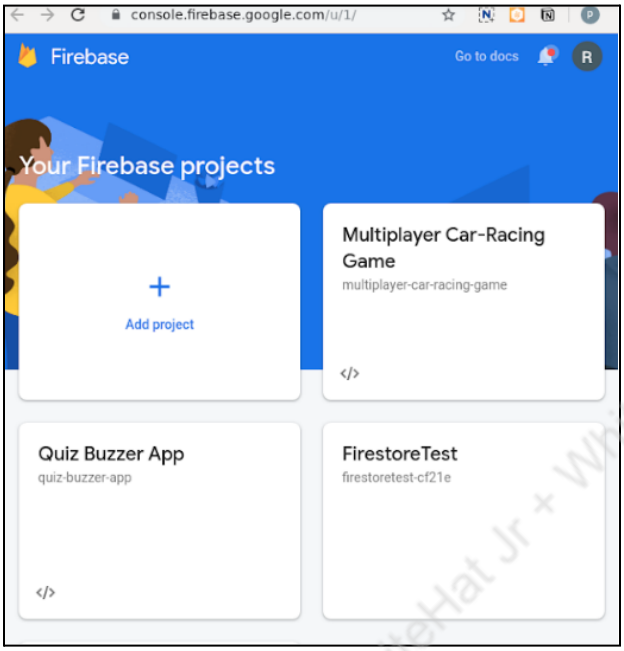
2. Create a handle transaction function and call **this.handleTransaction** in the **onPress** prop for Submit Button.

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

  handleTransaction = () => {

  };

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

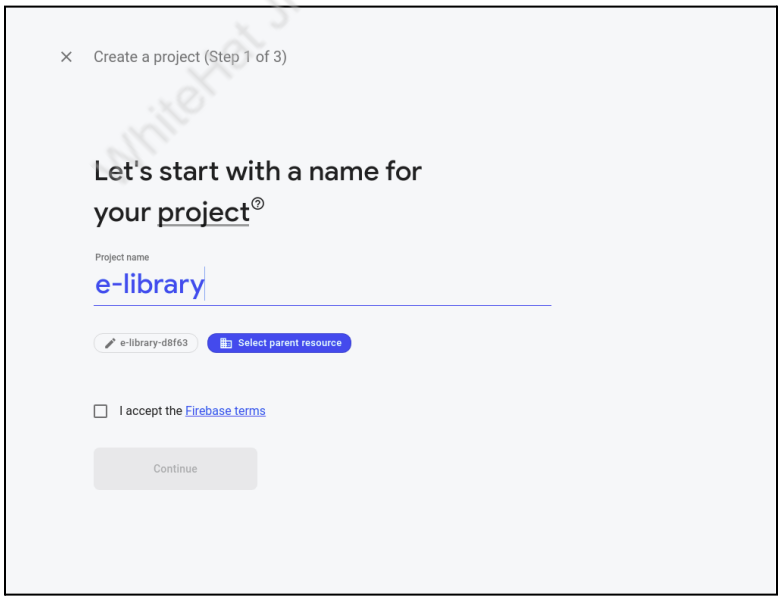
```
<TouchableOpacity
  style={[styles.button, { marginTop: 25 }]}
  onPress={this.handleTransaction}
>
  <Text style={styles.buttonText}>Submit</Text>
</TouchableOpacity>
```

3. Create a Firestore Database and learn how to use firestore in test mode.

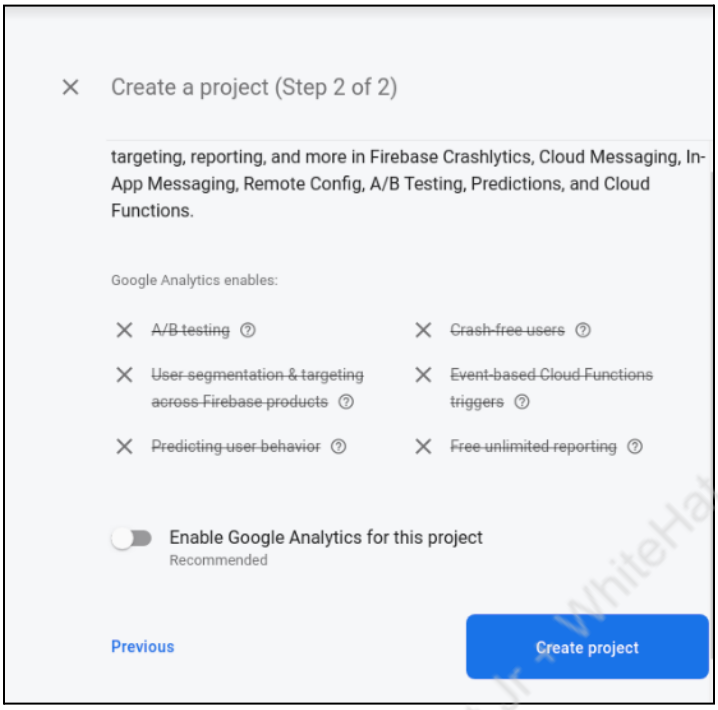
- Click on add project.



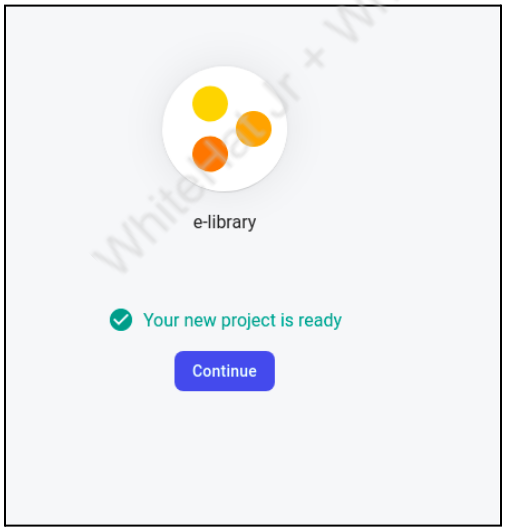
- Add a name to the app.



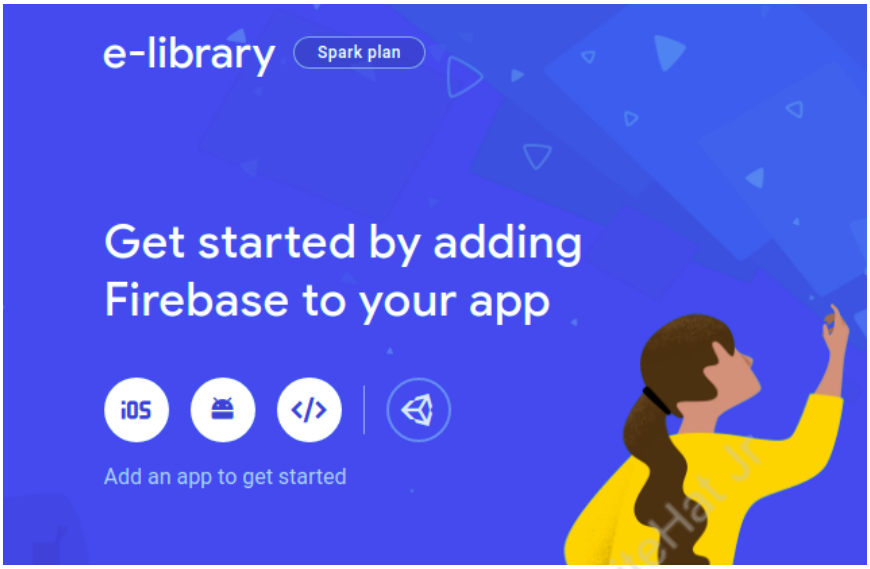
- Disable google analytics.



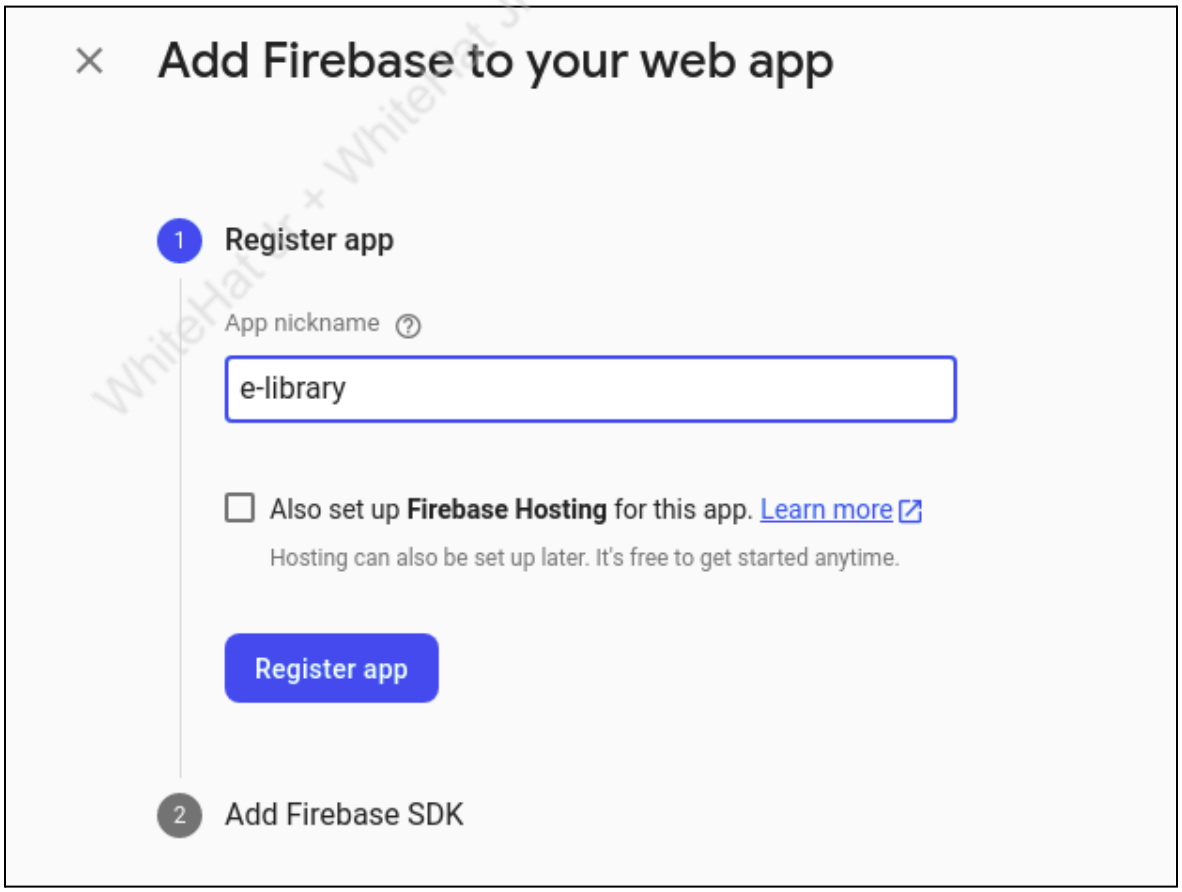
- Click on continue.



- Click the code Icon.



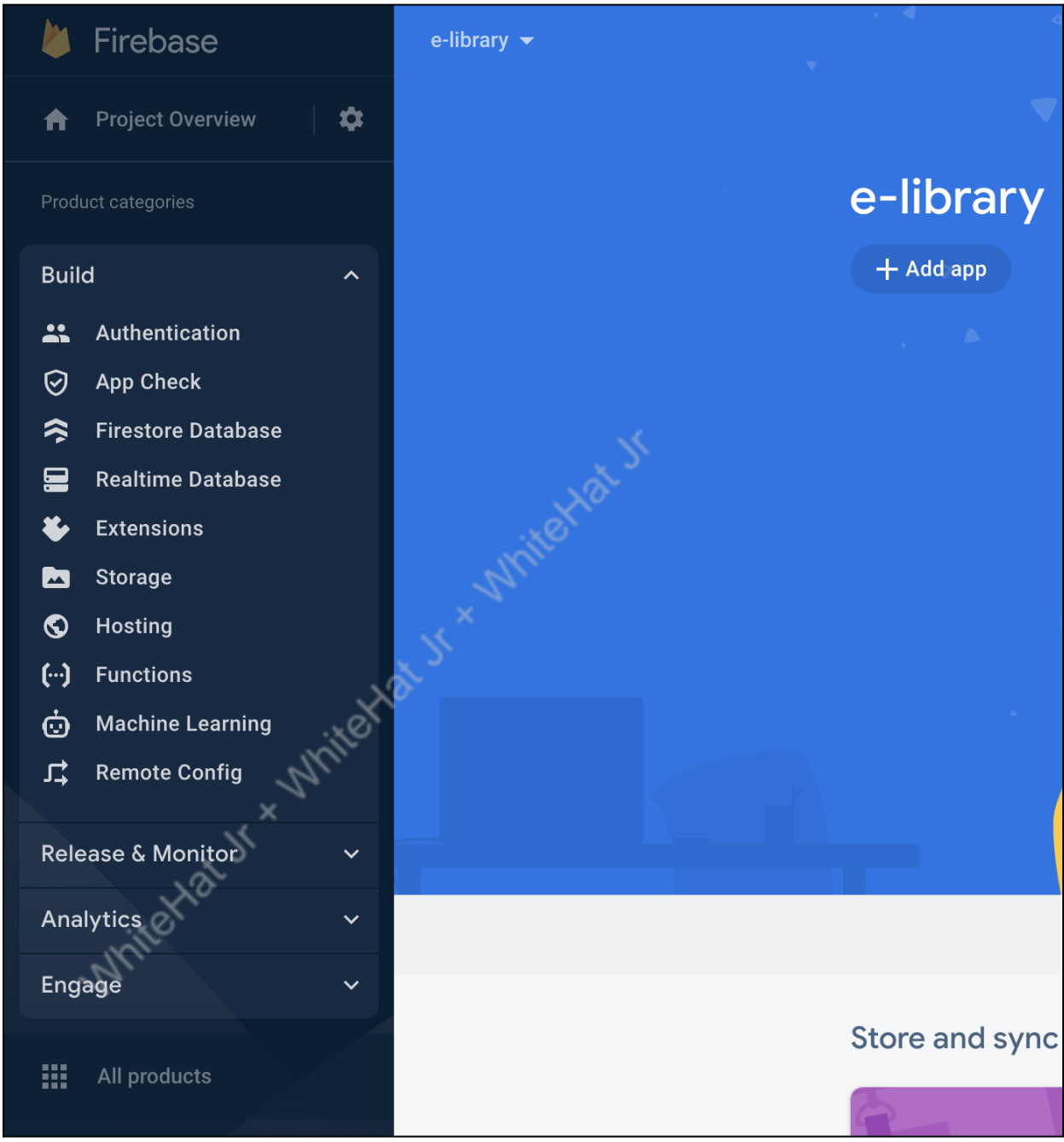
- Add firebase database to the webapp.



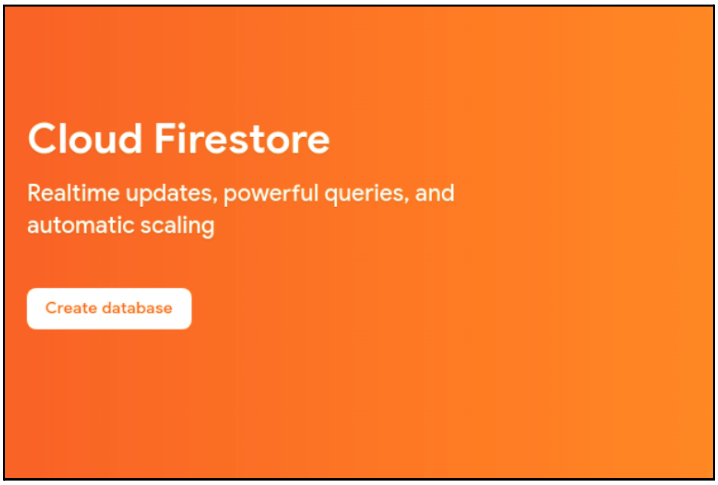
- Copy the credentials.

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
import { getAnalytics } from "firebase/analytics";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "AIzaSyBRxkWqTQzSOkU0GyO1dtZPl5A9AGMH1Wg",
  authDomain: "e-library-8416b.firebaseapp.com",
  projectId: "e-library-8416b",
  storageBucket: "e-library-8416b.appspot.com",
  messagingSenderId: "1064331388238",
  appId: "1:1064331388238:web:27b4a8df763199ca8d2a1f",
  measurementId: "G-V9B1ERK29G"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
const analytics = getAnalytics(app);
```

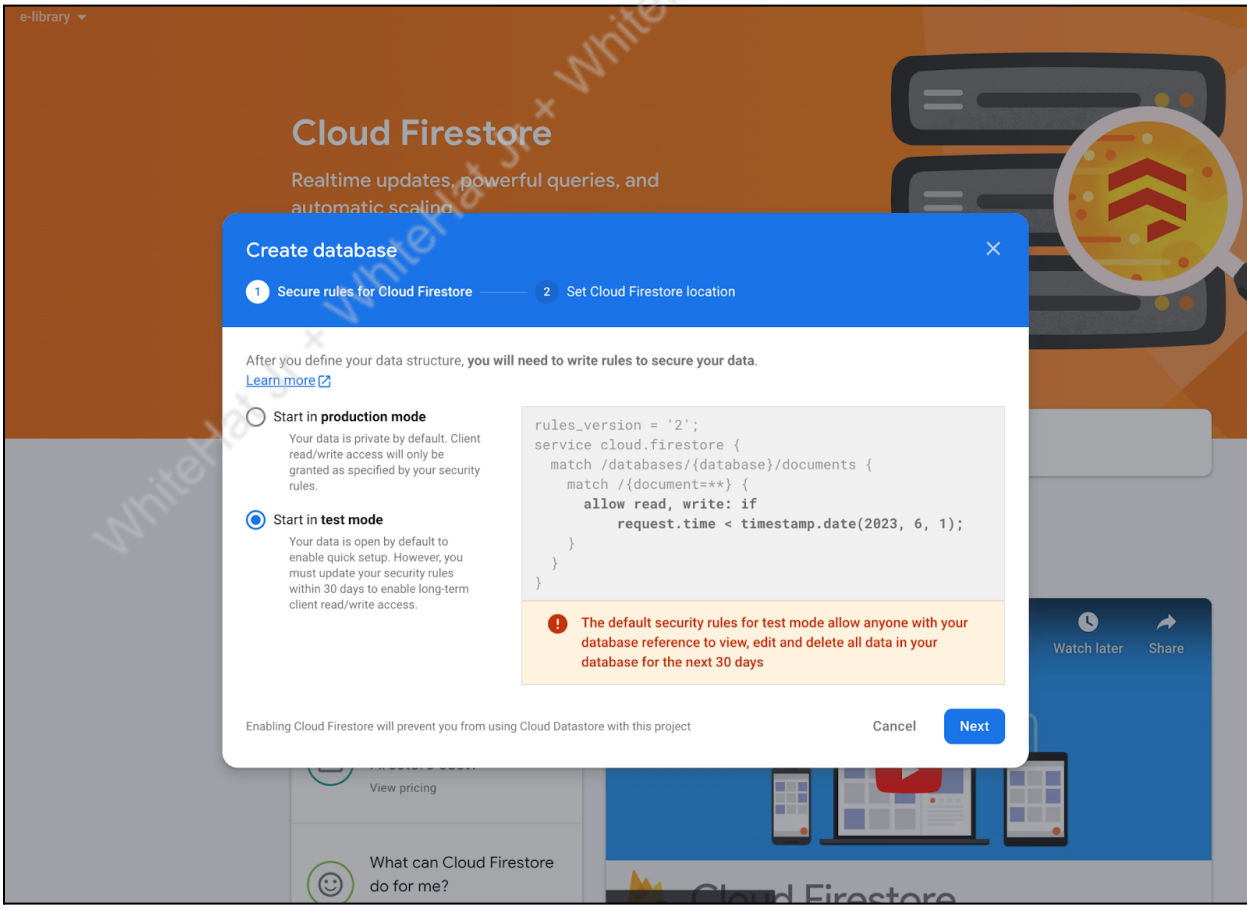
- Click on database inside the panel.



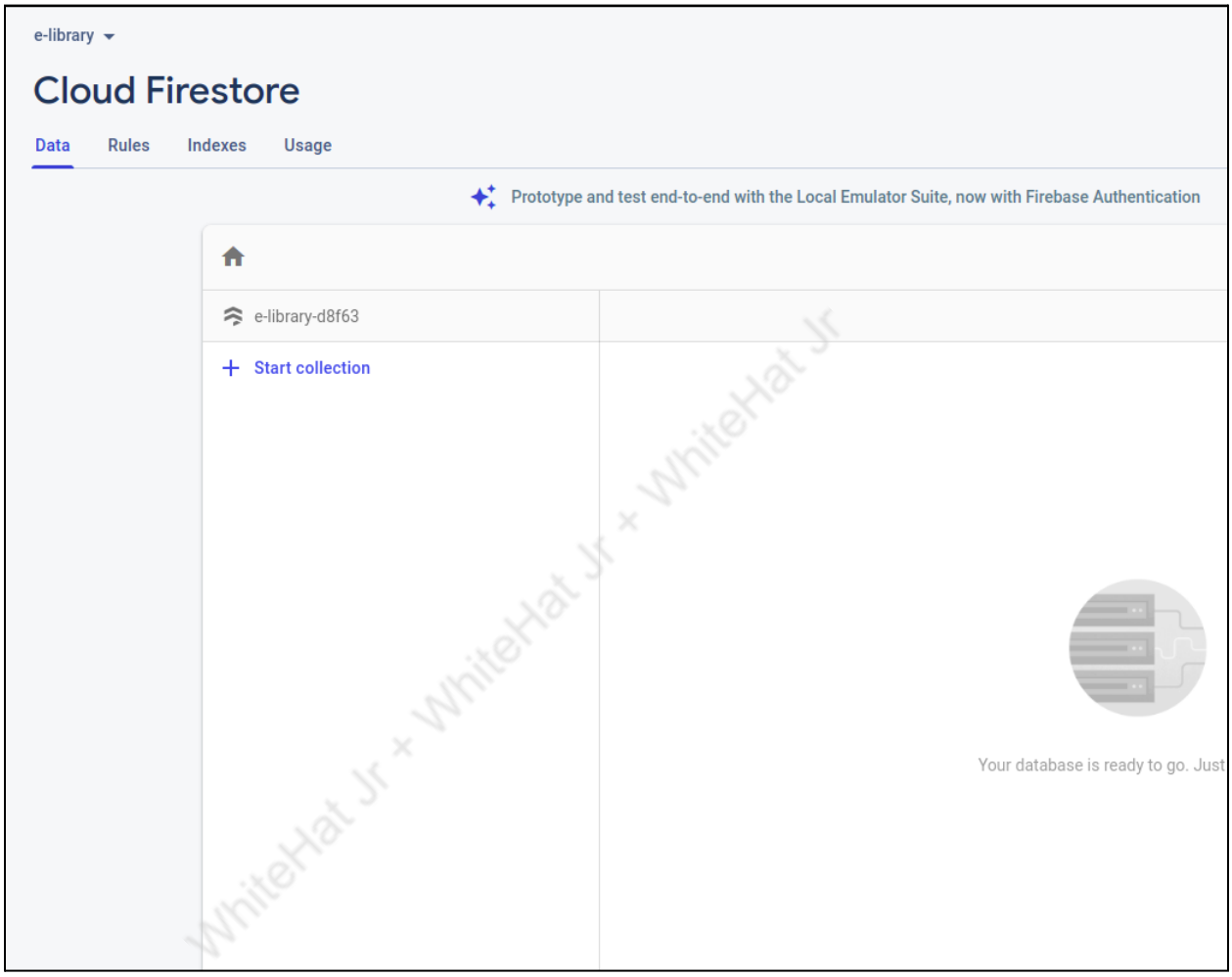
- Create a database.

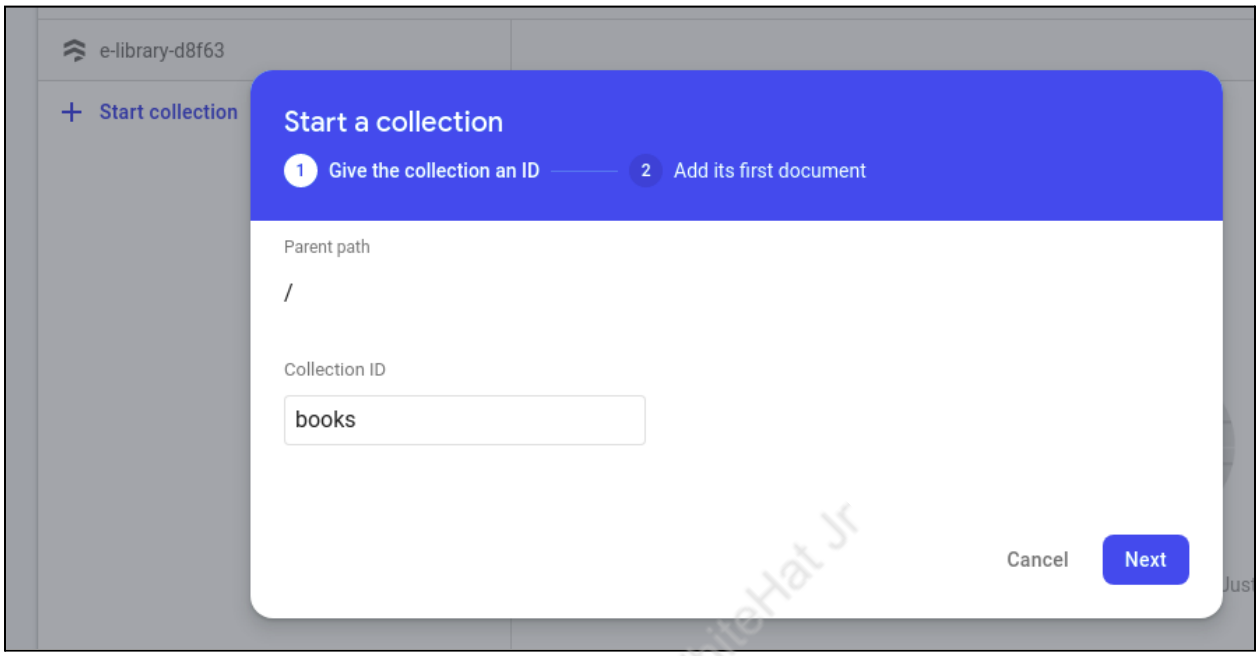


- Start database in Test mode.

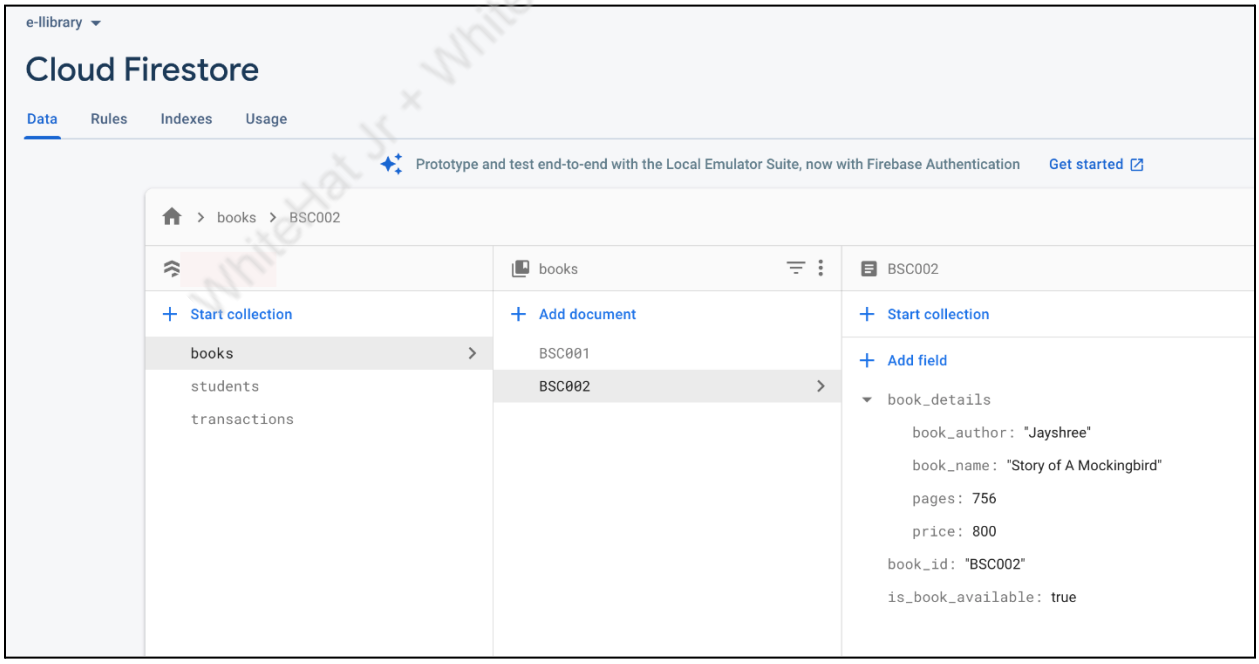


4. Firestore Database organizes all data in terms of collections and documents. Collection is the name given to a group of documents holding some common properties. Documents are data stored inside collections as separate entities.

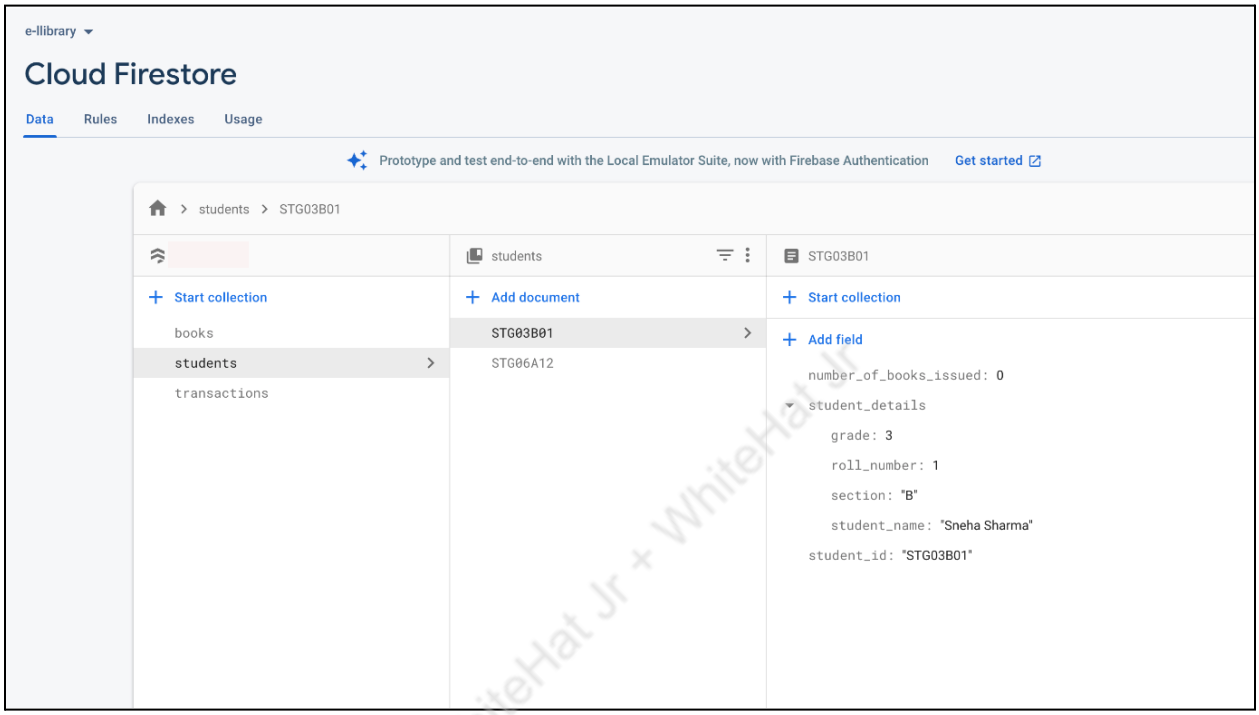




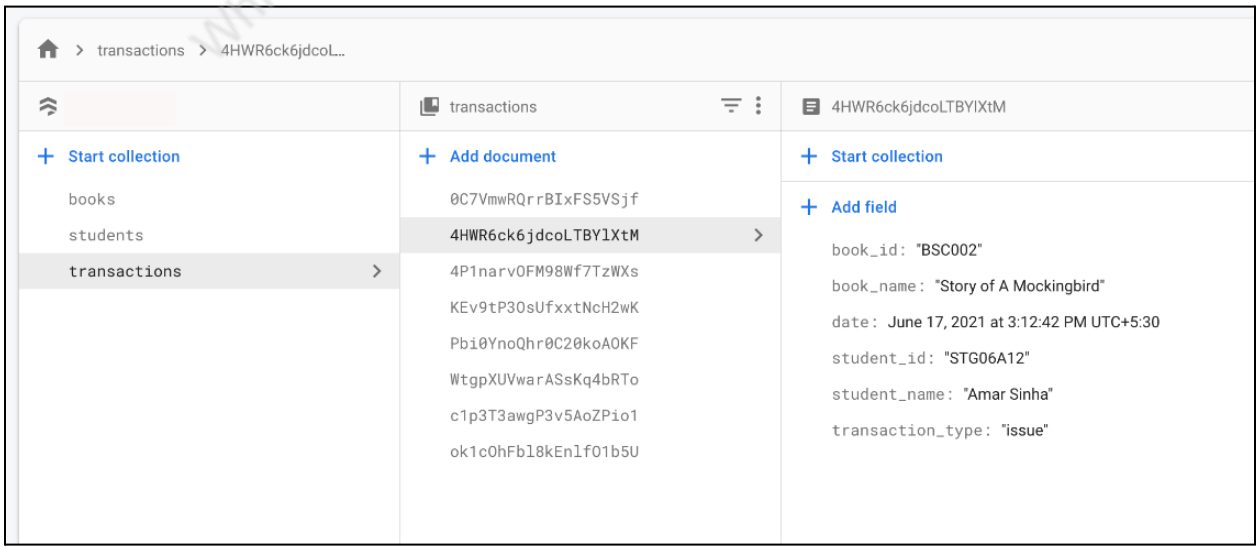
5. Create a collection in Firestore. Each document inside the collection will contain information about one book in the library. We will have as many documents as there are books in the library.



6. Create a Students Collection with a sample student document containing the fields - **studentId**, **studentDetails**, **numberOfBooksIssued.**



7. e-library app will be used to issue or return books. Each issue or return is a transaction. Create a collection called transactions.
   - It will contain the bookId, the student id, the date of transaction and the type of transaction (issue/return).



8. Let's create a **config.js** file in our app project where we initialize our firestore database and export **firebase.firestore()**.

```
JS config.js > ...
import { initializeApp } from 'firebase/app';
import { getFirestore } from 'firebase/firestore';

const firebaseConfig = {
    apiKey: 'AIzaSyA8URZYNDZM816bx1US9DXMz1Ra0bzrH9o',
    authDomain: 'wilyapp-d4952.firebaseapp.com',
    projectId: 'wilyapp-d4952',
    storageBucket: 'wilyapp-d4952.appspot.com',
    messagingSenderId: '690423895839',
    appId: '1:690423895839:web:0fc9718379abab5c14822a',
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
const db = getFirestore(app);

export default db;
```

```
C:\Users\ADMIN>npm install firebase
[..................] - rollbackFailedOptional: verb npm-session 734bf667a2aeba65
```

9. Import **firebase.firestore()** as db. There are a number of functions pre-defined on db which we are going to use to update or create documents in our database.

```
import React, { Component } from "react";
import {
  View,
  StyleSheet,
  TextInput,
  TouchableOpacity,
  Text,
  ImageBackground,
  Image
} from "react-native";
import * as Permissions from "expo-permissions";
import { BarCodeScanner } from "expo-barcode-scanner";
import db from "../config";

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

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

10. Create two QR codes corresponding to the studentId and bookId we created in our database so that we can test the code we are going to write.

```
10  } from 'react-native';
11  import * as Permissions from 'expo-permissions';
12  import { BarCodeScanner } from 'expo-barcode-scanner';
13  import db from '../config';
14  import { collection, query, where, getDocs } from 'firebase/firestore';
15
```

screens > JS Transaction.js > TransactionScreen > initiateBookIssue

```
54          } else if (domState === 'studentId') {
55              this.setState({
56                  studentId: data,
57                  domState: 'normal',
58                  scanned: true,
59              });
60          }
61      };
62
63      handleTransaction = async () => {
64          var { bookId } = this.state;
65
66          let dbQuery = query(
67              collection(db, 'books'),
68              where('book_id', '==', bookId)
69          );
70
71          let bookRef = await getDocs(dbQuery);
72
73          bookRef.forEach((doc) => {
74              console.log(doc.data());
75          });
76      };
77
78      initiateBookIssue = () => {
```

```
Object {
  "book_details": Object {
    "book_author": "Abhijeet Holkar",
    "book_name": "Travel Magazine",
    "pages": 209,
    "price": 499,
  },
  "book_id": "BSC001",
  "is_book_available": false,
}
```

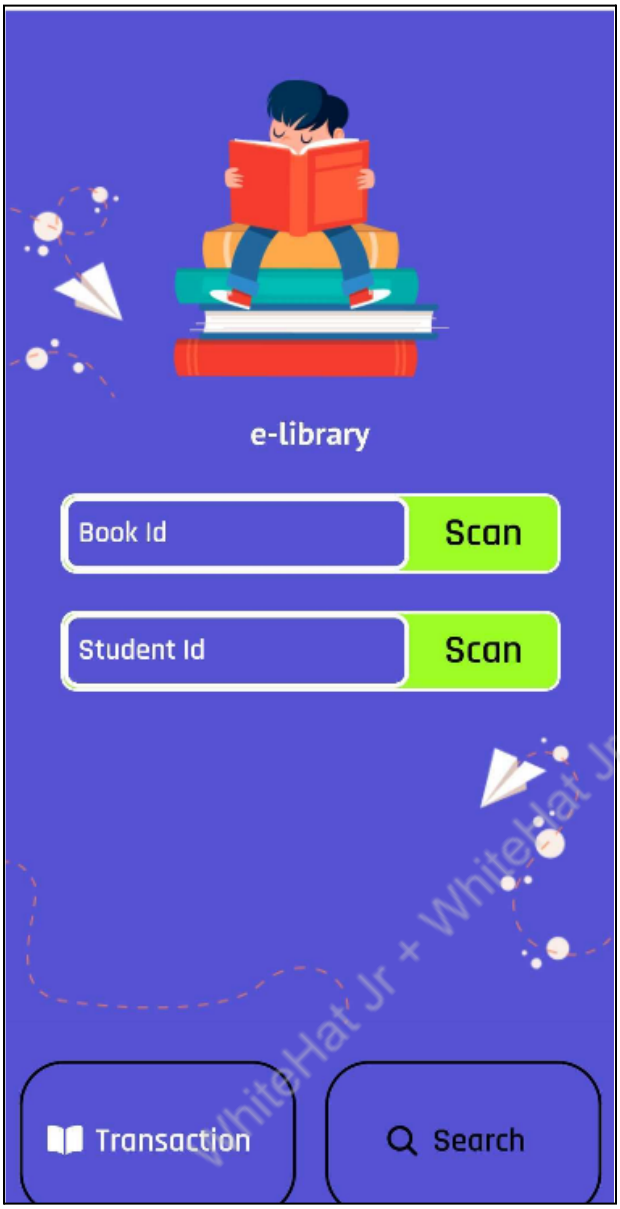
11. Write code to call **initiateBookIssue()** if the book is available or **initiateBookReturn()** when the book is not available.

```
        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) {
                this.initiateBookIssue();
            } else {
                this.initiateBookReturn();
            }
        });
    };

    initiateBookIssue = () => {
        console.log('Book issued to the student!');
    };

    initiateBookReturn = () => {
        console.log('Book returned to the library!');
    };

    render() {
        const { bookId, studentId, domState, scanned } = this.state;
        if (domState !== 'normal') {
            return (
                <BarCodeScanner
```

12. Get the bookAvailability field from the data. If the book is available, we want to issue the book. If the book is not available, we want to return the book. Use two abstract functions - initiateBookIssue() and initiateBookReturn() to write our code.

screens > Transaction.js > TransactionScreen

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

  handleTransaction = () => {
    var { bookId } = this.state;
    db.collection("books")
      .doc(bookId)
      .get()
      .then(doc => {
        var book = doc.data();
        if (book.is_book_available) {
          this.initiateBookIssue();
        } else {
          this.initiateBookReturn();
        }
      });
  };

  initiateBookIssue = () => {
    console.log("Book issued to the student!");
  };

  initiateBookReturn = () => {
    console.log("Book returned to the library!");
  };

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



13. Create and style the submit button.

```
            <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>

            <TouchableOpacity
              style={[styles.button, { marginTop: 25 }]}
            >
              <Text style={styles.buttonText}>Submit</Text>
            </TouchableOpacity>

          </View>
        </ImageBackground>
      </View>
    );
  }
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: "#FFFFFF"
  },
  bgImage: {
    flex: 1,
    resizeMode: "cover",
    justifyContent: "center"
  },
```

screens > Transaction.js > TransactionScreen > render

```
    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"
  },
  button: {
    width: "43%",
    height: 55,
    justifyContent: "center",
    alignItems: "center",
    backgroundColor: "#F48D20",
    borderRadius: 15
  },
  buttonText: {
    fontSize: 24,
    color: "#FFFFFF",
    fontFamily: "Rajdhani_600SemiBold"
  }
});
```

14. Create a firestore database with books, students and transaction collections and create a test book and student document.

```
C:\Users\ADMIN>npm install firebase
[..................] - rollbackFailedOptional: verb npm-session 734bf667a2aeba65
```

```
JS config.js > ...
import { initializeApp } from 'firebase/app';
import { getFirestore } from 'firebase/firestore';

const firebaseConfig = {
    apiKey: 'AIzaSyA8URZYNDZM816bx1US9DXMz1Ra0bzrH9o',
    authDomain: 'wilyapp-d4952.firebaseapp.com',
    projectId: 'wilyapp-d4952',
    storageBucket: 'wilyapp-d4952.appspot.com',
    messagingSenderId: '690423895839',
    appId: '1:690423895839:web:0fc9718379abab5c14822a',
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
const db = getFirestore(app);

export default db;
```

15. Create a function called **handleTransaction** which is called when the submit button is pressed.

```
10  } from 'react-native';
11  import * as Permissions from 'expo-permissions';
12  import { BarCodeScanner } from 'expo-barcode-scanner';
13  import db from '../config';
14  import { collection, query, where, getDocs } from 'firebase/firestore';
15
```

```
56                  studentId: data,
57                  domState: 'normal',
58                  scanned: true,
59              });
60          }
61      };
62
63      handleTransaction = async () => {
64          var { bookId } = this.state;
65
66          let dbQuery = query(
67              collection(db, 'books'),
68              where('book_id', '==', bookId)
69          );
70
71          let bookRef = await getDocs(dbQuery);
72
73          bookRef.forEach((doc) => {
74              var book = doc.data();
75              if (book.is_book_available) {
76                  this.initiateBookIssue();
77              } else {
78                  this.initiateBookReturn();
79              }
80          });
81      };
82
83      initiateBookIssue = () => {
```

16. Get the book availability data from the scanned book id and call issue or return functions to complete the book transaction.

```
        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) {
                this.initiateBookIssue();
            } else {
                this.initiateBookReturn();
            }
        });
    };

    initiateBookIssue = () => {
        console.log('Book issued to the student!');
    };

    initiateBookReturn = () => {
        console.log('Book returned to the library!');
    };

    render() {
        const { bookId, studentId, domState, scanned } = this.state;
        if (domState !== 'normal') {
            return (
                <BarCodeScanner
```

What's NEXT?

In the next class we will fix the issue of Keyboard Overlapping and show messages using Toasts.

EXTEND YOUR KNOWLEDGE

1. Cloud Firestore Documentation: https://firebase.google.com/docs/firestore