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.

```jsx
<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>
);
```

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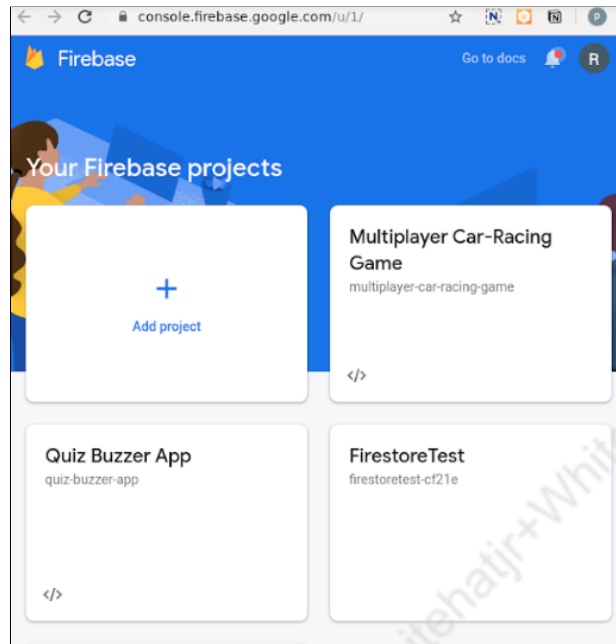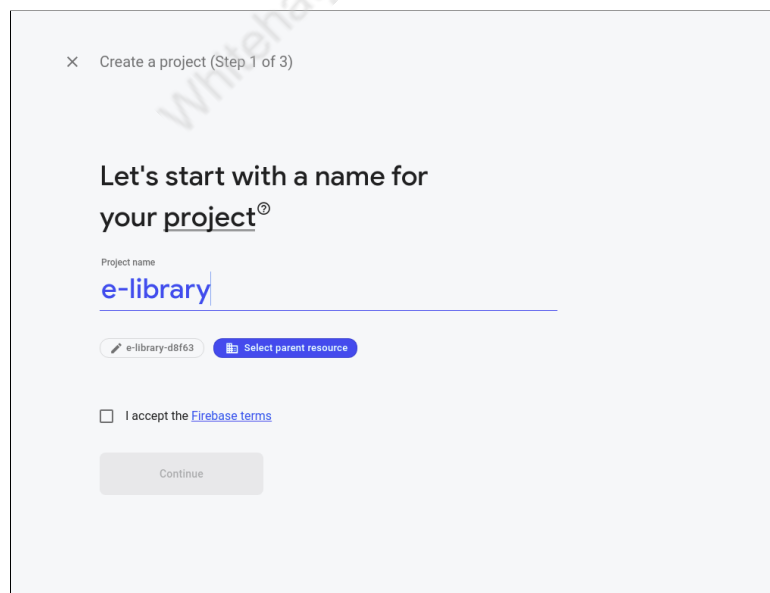
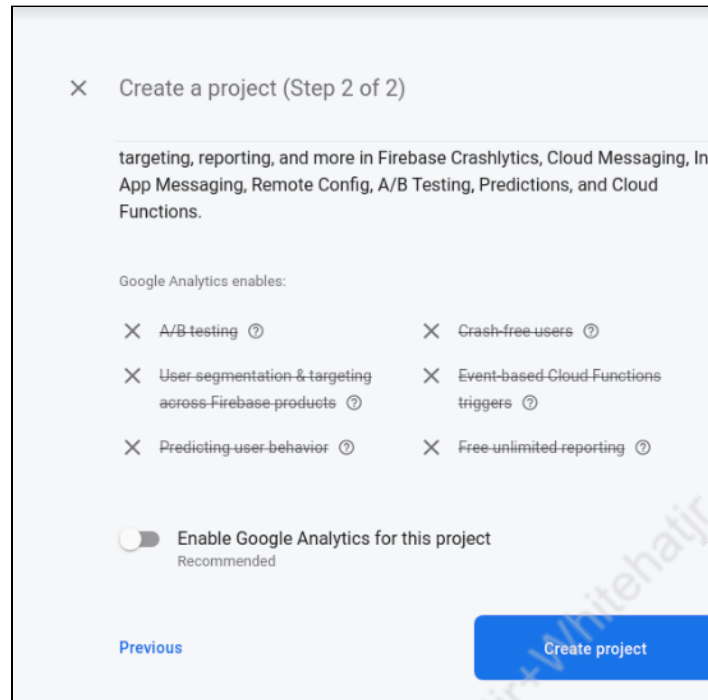3. Create a Firestore Database and learn how to use firestore in test mode.

- Click on add project.

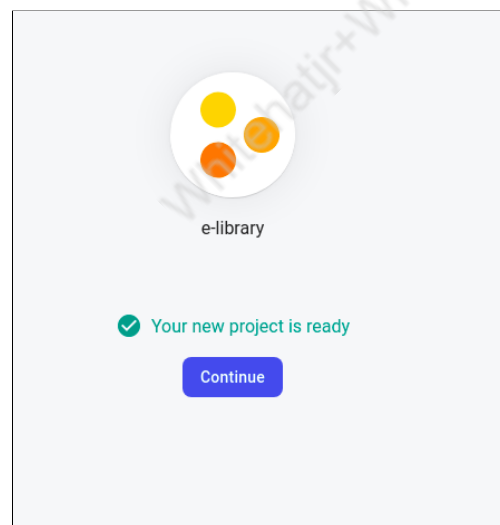- Add a name to the app.

- Disable google analytics.
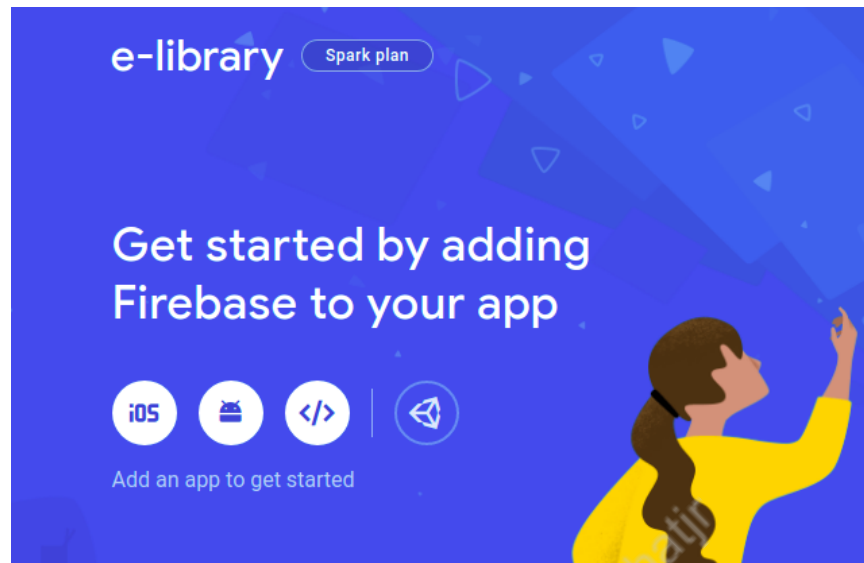

- Click on continue.

- Click the code Icon.

- Copy the credentials.

```
<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="https://www.gstatic.com/firebasejs/8.6.8/firebase-app.js"></script

<!-- TODO: Add SDKs for Firebase products that you want to use
     https://firebase.google.com/docs/web/setup#available-libraries -->

<script>
  // Your web app's Firebase configuration
  var firebaseConfig = {
    apiKey: "AIzaSyB3dzLbgE8DPcHufZzeVKYwgmpXD0HFZYs",
    authDomain: "e-library-5fd21.firebaseapp.com",
    projectId: "e-library-5fd21",
    storageBucket: "e-library-5fd21.appspot.com",
    messagingSenderId: "665049297120",
    appId: "1:665049297120:web:7442fb17f050bde800e085"
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);
</script>
```
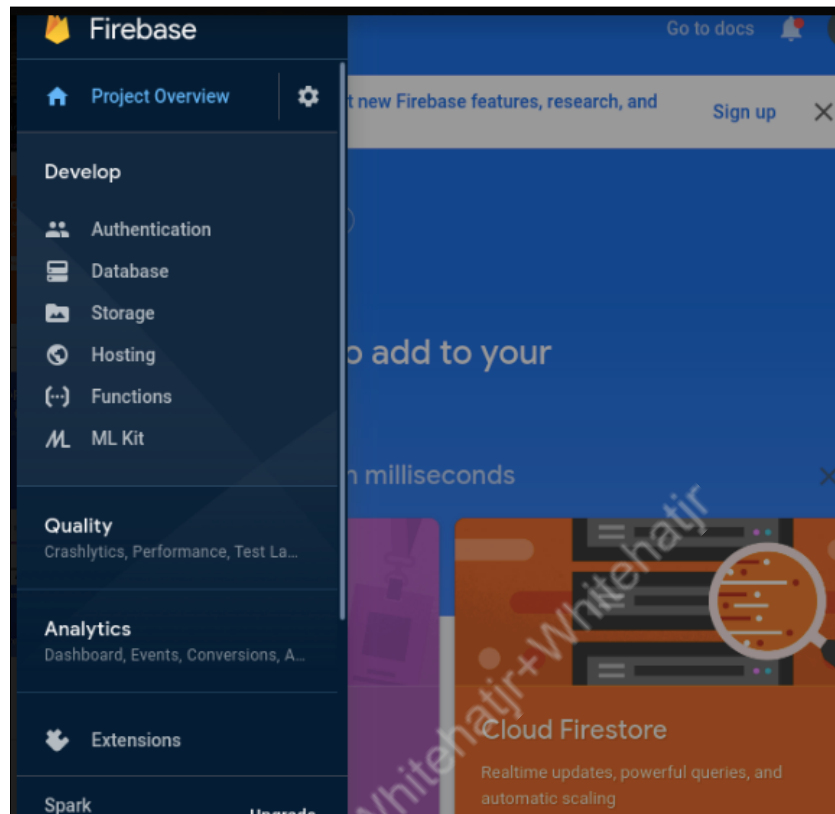
- Click on database inside the panel.

- Add Firebase database to the app.

- Create a database.
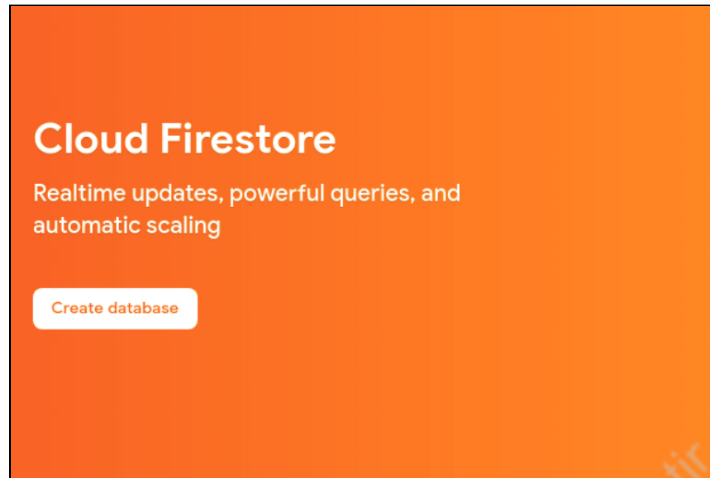
- Start database in Test mode.
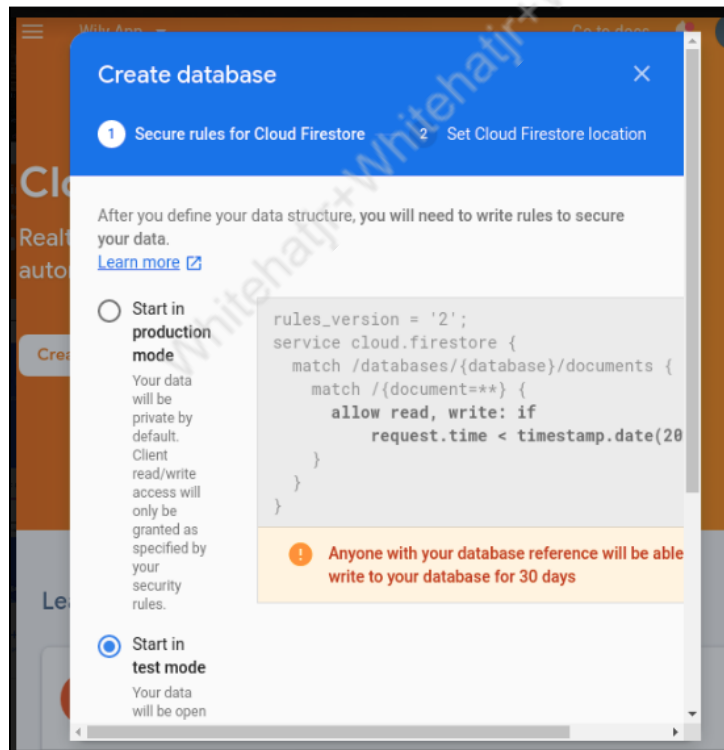
4. Firestore Database organizes all data in terms of collections and documents. Collection is the name given to a group of documents holding some common properties. Documents are data stored inside collections as separate entities.
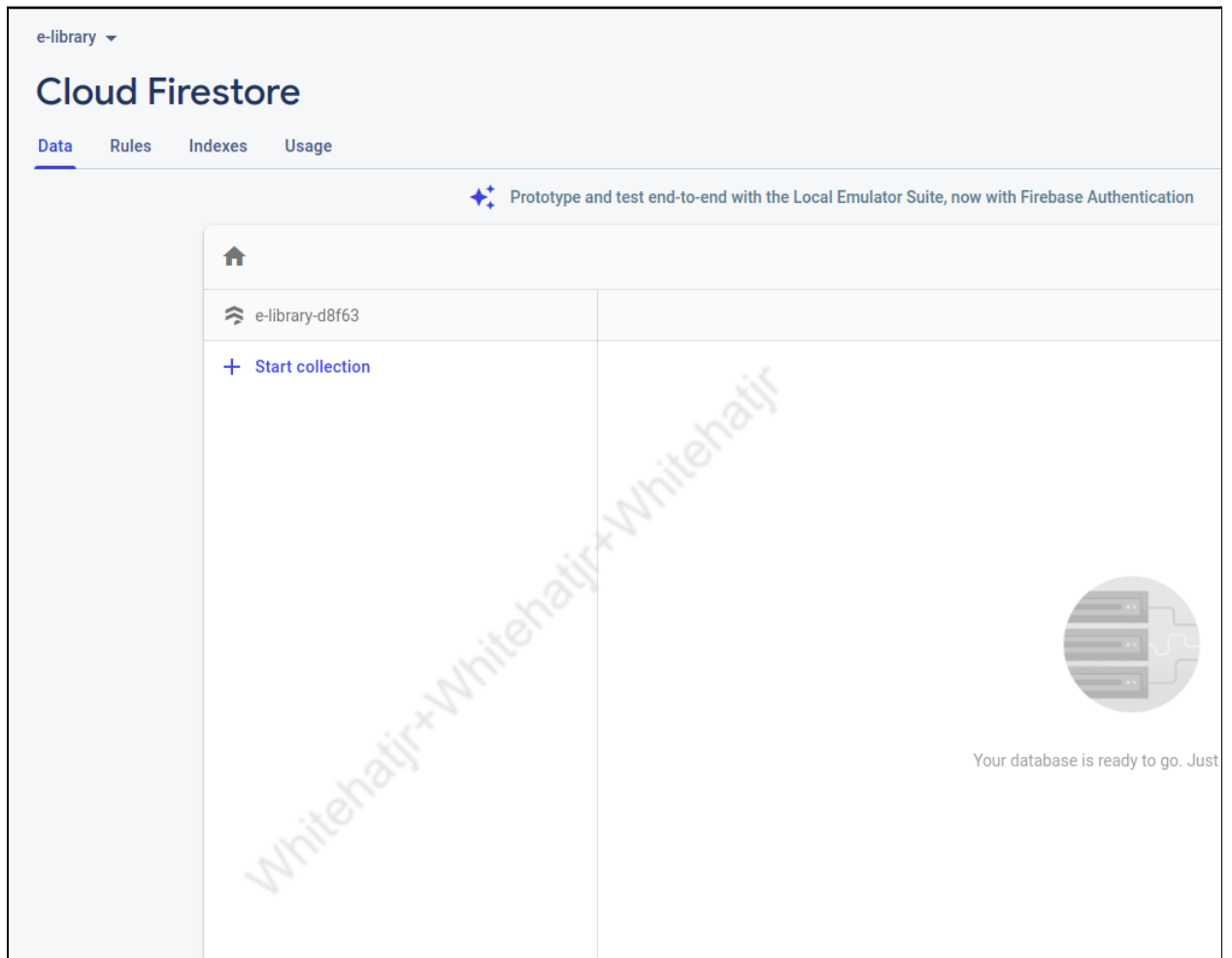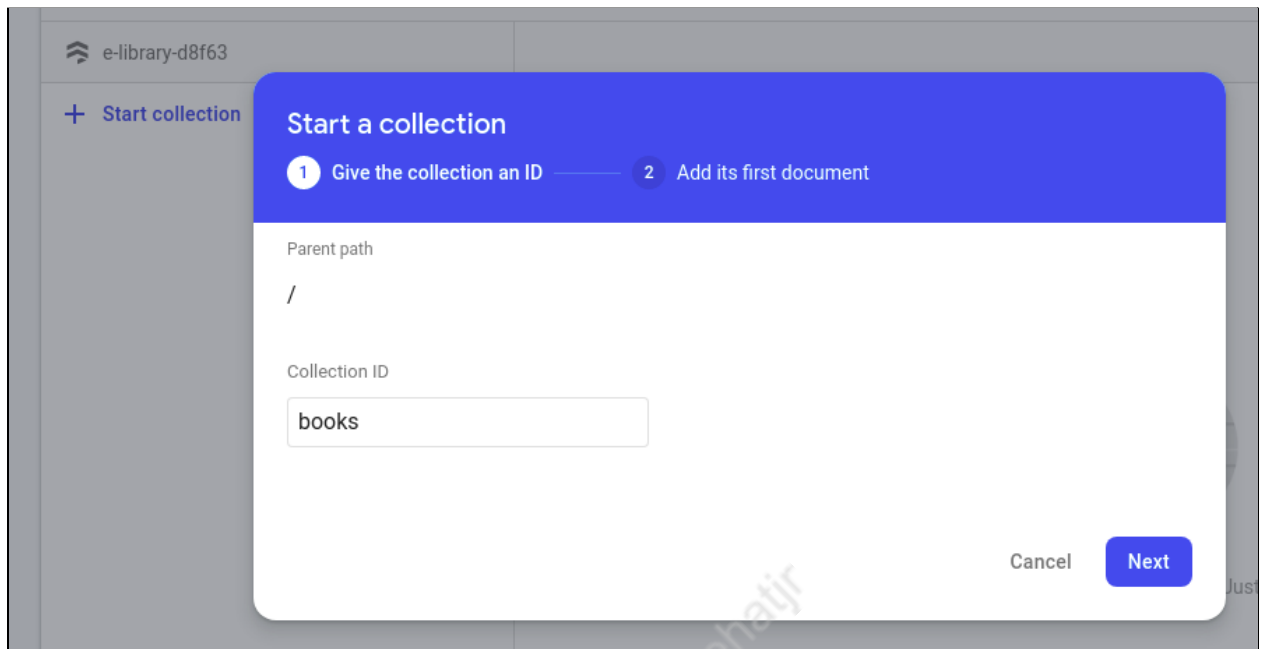
5. Create a collection in Firestore. Each document inside the collection will contain information about one book in the library. We will have as many documents as there are books in the library.
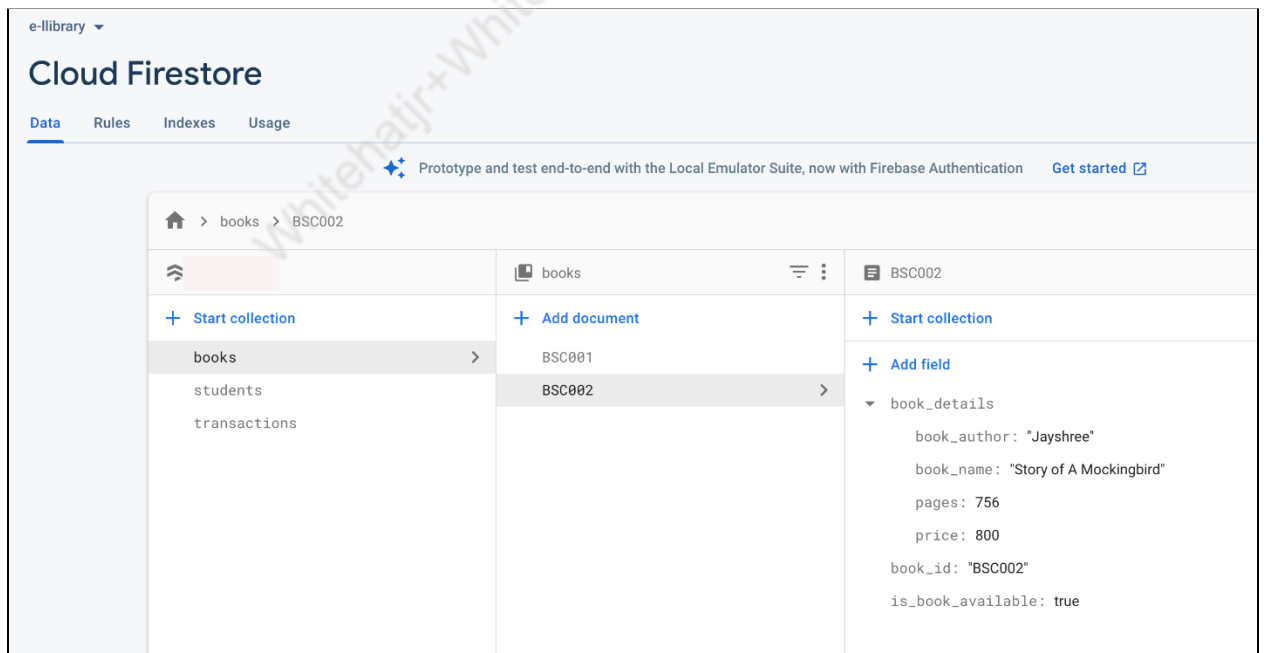
6. Create a Students Collection with a sample student document containing the fields - **studentId**, **studentDetails**, **numberOfBooksIssued.**
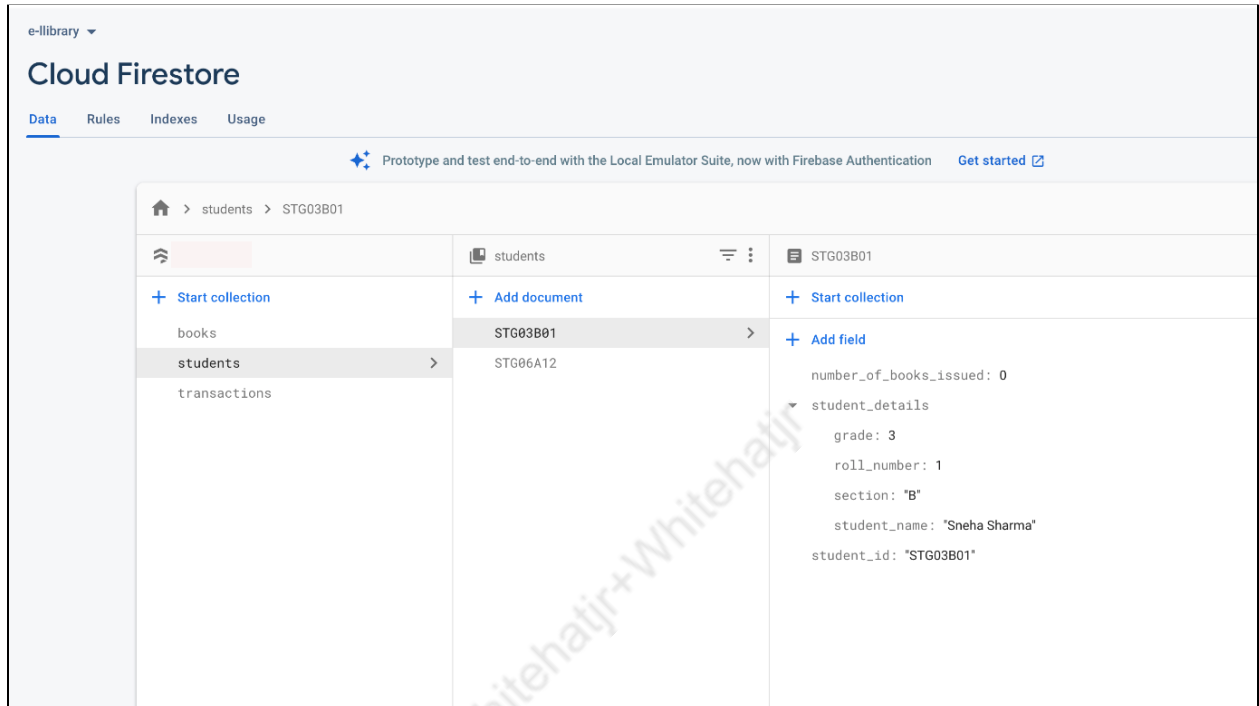
7. e-library app will be used to issue or return books. Each issue or return is a transaction. Create a collection called transactions.
   ● It will contain the bookId, the student id, the date of transaction and the type of transaction (issue/return).
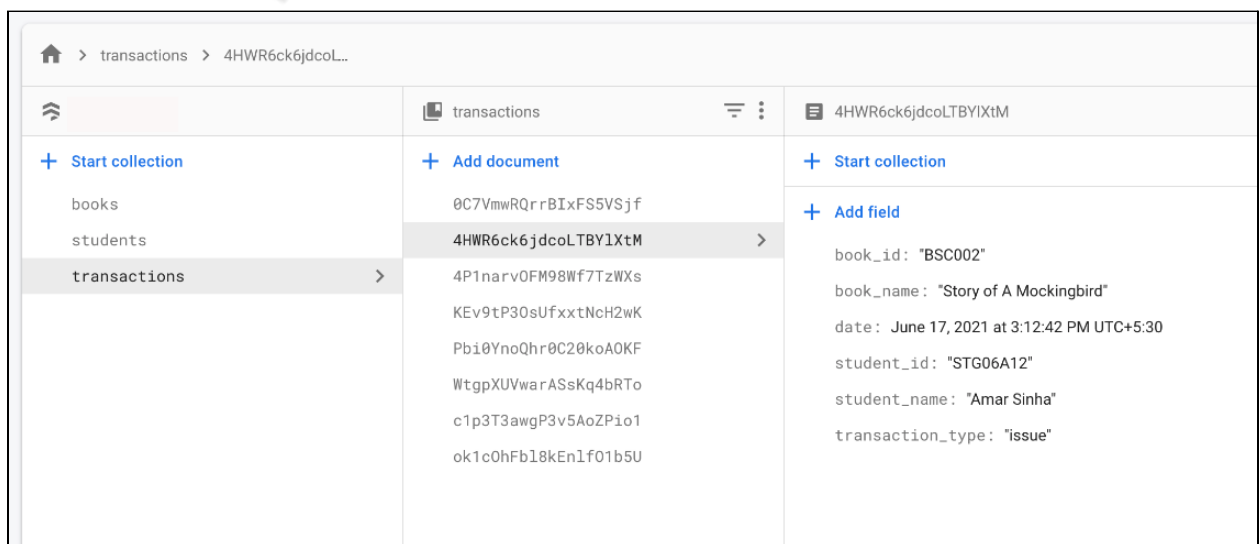
8. Let's create a **config.js** file in our app project where we initialize our firestore database and export **firebase.firestore()**.

```js
JS config.js > ...
1   import firebase from "firebase";
2   require("@firebase/firestore");
3
4   const firebaseConfig = {
5     apiKey: "AIzaSyAkp3gQdHbMPD5pCfHbBzIIgFpdCJXD5KM",
6     authDomain: "e-library-v2.firebaseapp.com",
7     projectId: "e-library-v2",
8     storageBucket: "e-library-v2.appspot.com",
9     messagingSenderId: "772559744213",
10    appId: "1:772559744213:web:9417b879882f231175065c",
11    measurementId: "G-H8Y352W6Z2"
12  };
13
14  firebase.initializeApp(firebaseConfig);
15
16  export default firebase.firestore();
17
```

```
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.

```javascript
        bookId: data,
        domState: "normal",
        scanned: true
      });
    } 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 => {
        console.log(doc.data())
      });
  };


  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}>
```

```
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.

```js
screens > JS Transaction.js > TransactionScreen
53        } else if (domState === "studentId") {
54          this.setState({
55            studentId: data,
56            domState: "normal",
57            scanned: true
58          });
59        }
60      };
61
62      handleTransaction = () => {
63        var { bookId } = this.state;
64        db.collection("books")
65          .doc(bookId)
66          .get()
67          .then(doc => {
68            var book = doc.data();
69            if (book.is_book_available) {
70              this.initiateBookIssue();
71            } else {
72              this.initiateBookReturn();
73            }
74          });
75      };
76
77      |
78
79      render() {
80        const { bookId, studentId, domState, scanned } = this.state;
81        if (domState !== "normal") {
82          return (
83            <BarCodeScanner
84              onBarCodeScanned={scanned ? undefined : this.handleBarCodeScanned}
85              style={StyleSheet.absoluteFillObject}
86            />
87          );
88        }
89        return (
90          <View style={styles.container}>
91            <ImageBackground source={bgImage} style={styles.bgImage}>
```

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 > JS Transaction.js > TransactionScreen
53        } else if (domState === "studentId") {
54          this.setState({
55            studentId: data,
56            domState: "normal",
57            scanned: true
58          });
59        }
60      };
61
62      handleTransaction = () => {
63        var { bookId } = this.state;
64        db.collection("books")
65          .doc(bookId)
66          .get()
67          .then(doc => {
68            var book = doc.data();
69            if (book.is_book_available) {
70              this.initiateBookIssue();
71            } else {
72              this.initiateBookReturn();
73            }
74          });
75      };
76
77      initiateBookIssue = () => {
78        console.log("Book issued to the student!");
79      };
80
81      initiateBookReturn = () => {
82        console.log("Book returned to the library!");
83      };
84
85      render() {
86        const { bookId, studentId, domState, scanned } = this.state;
87        if (domState !== "normal") {
88          return (
89            <BarCodeScanner
90              onBarCodeScanned={scanned ? undefined : this.handleBarCodeScanned}
91              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 > JS Transaction.js > ⚛ TransactionScreen > ⬡ render
187        borderRadius: 10,
188        borderWidth: 3,
189        fontSize: 18,
190        backgroundColor: "#5653D4",
191        fontFamily: "Rajdhani_600SemiBold",
192        color: "#FFFFFF"
193      },
194      scanbutton: {
195        width: 100,
196        height: 50,
197        backgroundColor: "#9DFD24",
198        borderTopRightRadius: 10,
199        borderBottomRightRadius: 10,
200        justifyContent: "center",
201        alignItems: "center"
202      },
203      scanbuttonText: {
204        fontSize: 24,
205        color: "#0A0101",
206        fontFamily: "Rajdhani_600SemiBold"
207      },
208      button: {
209        width: "43%",
210        height: 55,
211        justifyContent: "center",
212        alignItems: "center",
213        backgroundColor: "#F48D20",
214        borderRadius: 15
215      },
216      buttonText: {
217        fontSize: 24,
218        color: "#FFFFFF",
219        fontFamily: "Rajdhani_600SemiBold"
220      }
221    });
222
```

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
JS config.js > ...
 1    import firebase from "firebase";
 2    require("@firebase/firestore");
 3
 4    const firebaseConfig = {
 5      apiKey: "AIzaSyAkp3gQdHbMPD5pCfHbBzIIgFpdCJXD5KM",
 6      authDomain: "e-library-v2.firebaseapp.com",
 7      projectId: "e-library-v2",
 8      storageBucket: "e-library-v2.appspot.com",
 9      messagingSenderId: "772559744213",
10      appId: "1:772559744213:web:9417b879882f231175065c",
11      measurementId: "G-H8Y352W6Z2"
12    };
13
14    firebase.initializeApp(firebaseConfig);
15
16    export default firebase.firestore();
17
```

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

screens > JS Transaction.js > TransactionScreen

```
53        } else if (domState === "studentId") {
54          this.setState({
55            studentId: data,
56            domState: "normal",
57            scanned: true
58          });
59        }
60      };
61
62      handleTransaction = () => {
63        var { bookId } = this.state;
64        db.collection("books")
65          .doc(bookId)
66          .get()
67          .then(doc => {
68            var book = doc.data();
69            if (book.is_book_available) {
70              this.initiateBookIssue();
71            } else {
72              this.initiateBookReturn();
73            }
74          });
75      };
76
77    |
78
79      render() {
80        const { bookId, studentId, domState, scanned } = this.state;
81        if (domState !== "normal") {
82          return (
83            <BarCodeScanner
84              onBarCodeScanned={scanned ? undefined : this.handleBarCodeScanned}
85              style={StyleSheet.absoluteFillObject}
86            />
87          );
88        }
89        return (
90          <View style={styles.container}>
91            <ImageBackground source={bgImage} style={styles.bgImage}>
```

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

```
screens > JS Transaction.js > TransactionScreen
53        } else if (domState === "studentId") {
54          this.setState({
55            studentId: data,
56            domState: "normal",
57            scanned: true
58          });
59        }
60      };
61
62      handleTransaction = () => {
63        var { bookId } = this.state;
64        db.collection("books")
65          .doc(bookId)
66          .get()
67          .then(doc => {
68            var book = doc.data();
69            if (book.is_book_available) {
70              this.initiateBookIssue();
71            } else {
72              this.initiateBookReturn();
73            }
74          });
75      };
76
77      initiateBookIssue = () => {
78        console.log("Book issued to the student!");
79      };
80
81      initiateBookReturn = () => {
82        console.log("Book returned to the library!");
83      };
84
85      render() {
86        const { bookId, studentId, domState, scanned } = this.state;
87        if (domState !== "normal") {
88          return (
89            <BarCodeScanner
90              onBarCodeScanned={scanned ? undefined : this.handleBarCodeScanned}
91              style={StyleSheet.absoluteFillObject}
```

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