FIRESTORE QUERIES

What is our GOAL for this MODULE?

In this class, we learned how to make queries to a Firebase Database and check for student eligibility and book eligibility before issuing/returning a book using firebase queries.

What did we ACHIEVE in the class TODAY?

- Made a firebase query to check if the book is eligible to be issued/returned.
- Made a firebase query to check if the student is eligible for the transaction.
- Displayed a message to the user for a successful/unsuccessful transaction.

Which CONCEPTS/CODING BLOCKS did we cover today?

- Writing firebase queries.

How did we DO the activities?

1. Make a firebase query to check if the book and student ID exist in the database.

    - Write a query on the books' collection to check if any of the book ID matches our given **bookId**.
    - If the query doesn't return anything then we'll return the transaction type as false, else using the **ternary** operator we'll check the **is_book_available** flag to **return** the issue if the book is present or return if the book is not available. And then return the transaction type.

```
checkBookAvailability = async (bookId) => {
    let dbQuery = query(
        collection(db, 'books'),
        where('book_id', '==', bookId)
    );

    let bookRef = await getDocs(dbQuery);

    var transactionType = '';
    if (bookRef.docs.length == 0) {
        transactionType = false;
    } else {
        bookRef.forEach((doc) => {
            //if the book is available then transaction type will be issue
            // otherwise it will be return
            transactionType = doc.data().is_book_available ? 'issue' : 'return';
        });
    }

    return transactionType;
};
```

2. Show the alert that the book doesn't exist in the library, if the transaction type is **Issue** then we check if the student is eligible for issuing the book. If the student is eligible, then we call the **initiateBookIssue** function and give an alert that the book is issued to the student.

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

    var transactionType = await this.checkBookAvailability(bookId);

    if (!transactionType) {
        this.setState({ bookId: '', studentId: '' });
        // For Android users only
        // ToastAndroid.show("The book doesn't exist in the library database!", ToastAndroid.SHORT);
        Alert.alert("The book doesn't exist in the library database!");
    } else if (transactionType === 'issue') {
        var { bookName, studentName } = this.state;
        this.initiateBookIssue(bookId, studentId, bookName, studentName);

        // For Android users only
        // ToastAndroid.show("Book issued to the student!", ToastAndroid.SHORT);
        Alert.alert('Book issued to the student!');
    } 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);
        Alert.alert('Book returned to the library!');
    }
};
```

3. Write the **checkStudentEligibilityForBookIssue()** function to check for student eligibility.
    - It returns **true** if the student is eligible and **false** if the student is not eligible.
    - If the student is eligible, Call the **initiateBookIssue()** function and issue an alert that the book has been issued.
    - Empty the **TextInputs** after the books have been issued.

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

    var transactionType = await this.checkBookAvailability(bookId);

    if (!transactionType) {
        this.setState({ bookId: '', studentId: '' });
        // For Android users only
        // ToastAndroid.show("The book doesn't exist in the library database!", ToastAndroid.SHORT);
        Alert.alert("The book doesn't exist in the library database!");
    } else if (transactionType === 'issue') {
        var isEligible = await this.checkStudentEligibilityForBookIssue(
            studentId
        );

        if (isEligible) {
            var { bookName, studentName } = this.state;
            this.initiateBookIssue(bookId, studentId, bookName, studentName);
        }
        // For Android users only
        // ToastAndroid.show("Book issued to the student!", ToastAndroid.SHORT);
        Alert.alert('Book issued to the student!');
    } else {
        var isEligible = await this.checkStudentEligibilityForBookReturn(
            bookId,
            studentId
        );

        if (isEligible) {
            var { bookName, studentName } = this.state;
            this.initiateBookReturn(bookId, studentId, bookName, studentName);
        }
        // For Android users only
        // ToastAndroid.show("Book returned to the library!", ToastAndroid.SHORT);
        Alert.alert('Book returned to the library!');
    }
};
```

4. The **handleTransactions** function is almost ready. Now check if the student is eligible to issue or return a book.

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

  var transactionType = await this.checkBookAvailability(bookId);

  if (!transactionType) {
    this.setState({ bookId: "", studentId: "" });
    // For Android users only
    // ToastAndroid.show("The book doesn't exist in the library database!", ToastAndroid.SHORT);
    Alert.alert("The book doesn't exist in the library database!");
  } else if (transactionType === "issue") {
    var isEligible = await this.checkStudentEligibilityForBookIssue(
      studentId
    );

    if (isEligible) {
      var { bookName, studentName } = this.state;
      this.initiateBookIssue(bookId, studentId, bookName, studentName);
    }
    // For Android users only
    // ToastAndroid.show("Book issued to the student!", ToastAndroid.SHORT);
    Alert.alert("Book issued to the student!");
  } else {
    var isEligible = await this.checkStudentEligibilityForBookReturn(
      bookId,
      studentId
    );

    if (isEligible) {
      var { bookName, studentName } = this.state;
      this.initiateBookReturn(bookId, studentId, bookName, studentName);
    }
    // For Android users only
    // ToastAndroid.show("Book returned to the library!", ToastAndroid.SHORT);
    Alert.alert("Book returned to the library!");
  }
};
```

5. Write a query to the student collection to check if the student with the scanned student ID exists and make changes in function **checkStudentEligibilityForBookIssue()**.

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

    let studentRef = await getDocs(dbQuery);

    var isStudentEligible = '';
    if (studentRef.docs.length == 0) {
        this.setState({
            bookId: '',
            studentId: '',
        });
        isStudentEligible = false;
        Alert.alert("The student id doesn't exist in the database!");
    } else {
        studentRef.forEach((doc) => {
            if (doc.data().number_of_books_issued < 2) {
                isStudentEligible = true;
            } else {
                isStudentEligible = false;
                Alert.alert('The student has already issued 2 books!');
                this.setState({
                    bookId: '',
                    studentId: '',
                });
            }
        });
    }

    return isStudentEligible;
};
```

6. Write a query to the transactions collection to check the last transaction for the book and make changes in the function **checkStudentEligibilityForReturn()**.

```
checkStudentEligibilityForBookReturn = async (bookId, studentId) => {
    let dbQuery = query(
        collection(db, 'transactions'),
        where('book_id', '==', bookId),
        limit(1)
    );

    let transactionRef = await getDocs(dbQuery);

    var isStudentEligible = '';
    transactionRef.forEach((doc) => {
        var lastBookTransaction = doc.data();
        if (lastBookTransaction.student_id === studentId) {
            isStudentEligible = true;
        } else {
            isStudentEligible = false;
            Alert.alert("The book wasn't issued by this student!");
            this.setState({
                bookId: '',
                studentId: '',
            });
        }
    });
    return isStudentEligible;
};
```


What's NEXT?

In the next class, we will learn how to display items in a list using the FlatList component. And also learn to build a search bar in the search screen to display the list of transactions queried by the user.

EXTEND YOUR KNOWLEDGE

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