NAME:DIVIT WADHWANI EXP:6 ROLL NO:71

AIM: Integrating Firebase with Flutter

THEORY:

The aim of this lab experiment is to connect a Flutter application to Firebase, a powerful backend-as-a-service (BaaS) platform provided by Google. This integration will enable real-time data management, authentication, and cloud storage capabilities within the Flutter app.

Theory:

1. Firebase Overview:

Firebase is a comprehensive suite of cloud-based tools and services that facilitates the development of scalable and feature-rich mobile and web applications. It offers a range of services, including Realtime Database, Authentication, Cloud Firestore, Cloud Functions, Cloud Storage, and more.

2. Realtime Database vs. Cloud Firestore:
   - Realtime Database: A NoSQL cloud database that stores and syncs data in real-time. It's a JSON tree-like structure where changes trigger updates across all connected clients simultaneously.
   - Cloud Firestore: A more flexible and scalable NoSQL database that stores data in documents and collections. It offers real-time synchronization and supports more complex queries.
3. Setting Up Firebase:
   - Create a new project on the Firebase console (https://console.firebase.google.com/).
   - Add your Flutter app to the Firebase project by registering its package name.
   - Download the google-services.json file and place it in the Android app directory for Android apps or GoogleService-Info.plist for iOS apps.
4. Firebase Authentication:
   - Implement user authentication using Firebase Authentication. This allows users to sign in with email/password, Google, Facebook, etc.
5. Real-Time Database Integration:
   - Connect your Flutter app to Firebase Realtime Database or Cloud Firestore.
   - Set up listeners to receive real-time updates when data in the database changes.
6. Cloud Storage Integration:
   - Utilize Firebase Cloud Storage to store and retrieve files, such as images or documents.
   - Implement file upload and download functionality in your Flutter app.

7. Handling Security Rules:
    - Configure security rules to control access to your Firebase resources. Define who can read or write data in the database and access files in storage.

Experiment Steps:

Flutter Project Setup:
- Create a new Flutter project using the Flutter CLI or an IDE like VSCode.
- Add necessary dependencies in pubspec.yaml for Firebase services.

Firebase Configuration:
- Create a new project on the Firebase console.
- Register your Flutter app and download the configuration files.

Integration with Authentication:
- Implement Firebase Authentication to enable user sign-in.

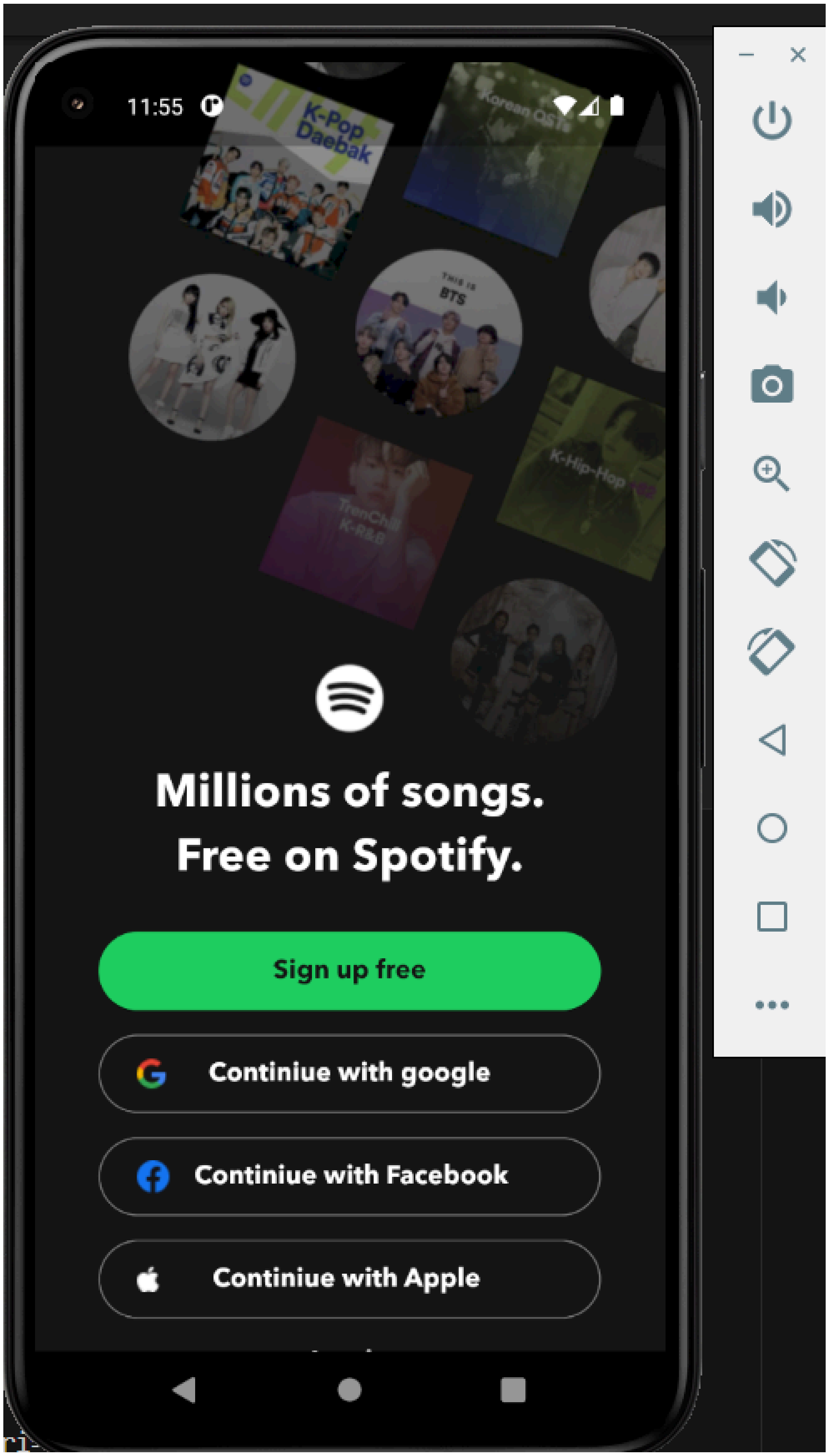
Database Integration:
- Connect your Flutter app to Firebase Realtime Database or Cloud Firestore.
- Create, read, update, and delete data in real-time.

Cloud Storage Integration:
- Set up Firebase Cloud Storage and implement file upload and download functionality in your Flutter app.

Testing and Troubleshooting:
- Test the integration by running the Flutter app and ensuring data synchronization and user authentication work as expected.
- Troubleshoot any potential issues or errors that may arise during the integration process.

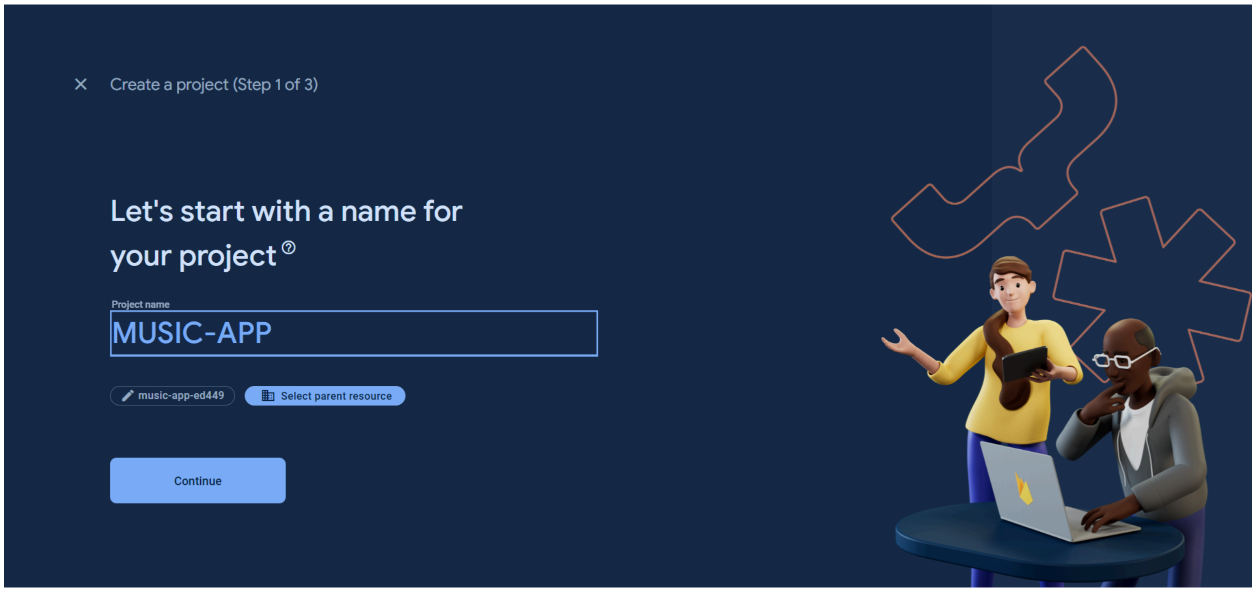
Login screen:

NAME:DIVIT WADHWANI EXP:6 ROLL NO:71



NAME:DIVIT WADHWANI EXP:6 ROLL NO:71

FIREBASE SETUP:



Create a project (Step 1 of 3)

Let's start with a name for your project

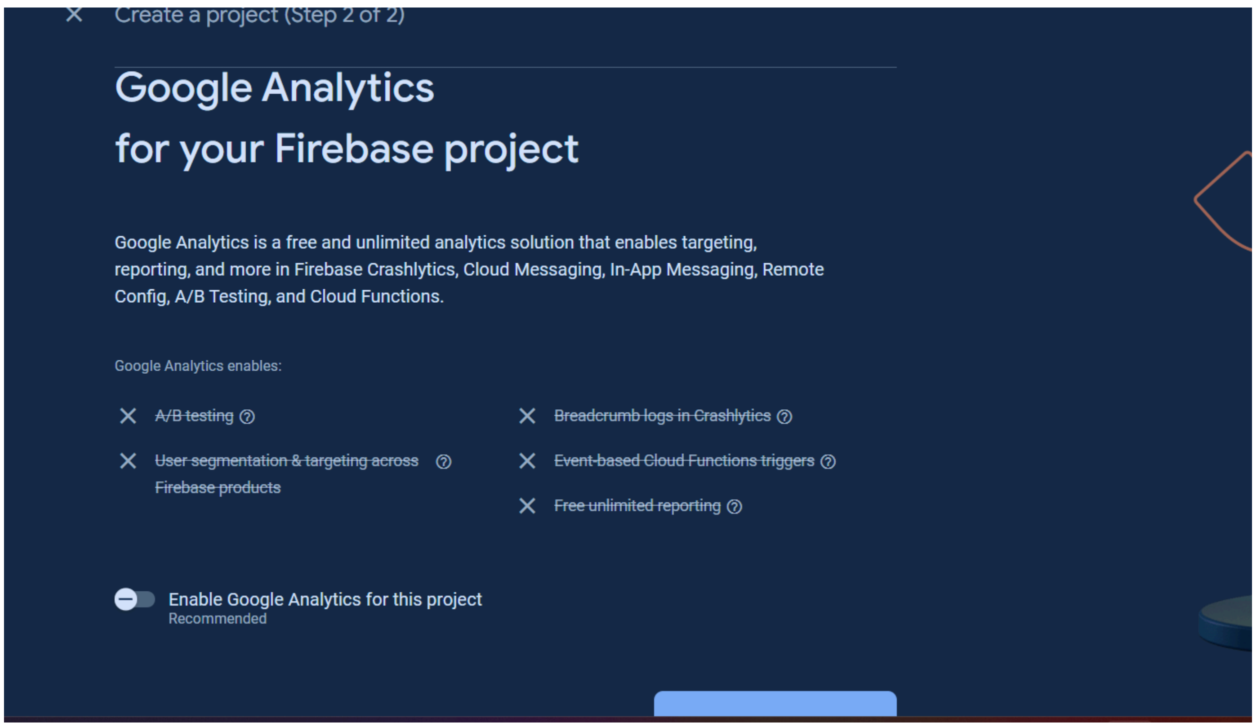
Project name

MUSIC-APP

music-app-ed449

Select parent resource

Continue



Create a project (Step 2 of 2)

Google Analytics for your Firebase project

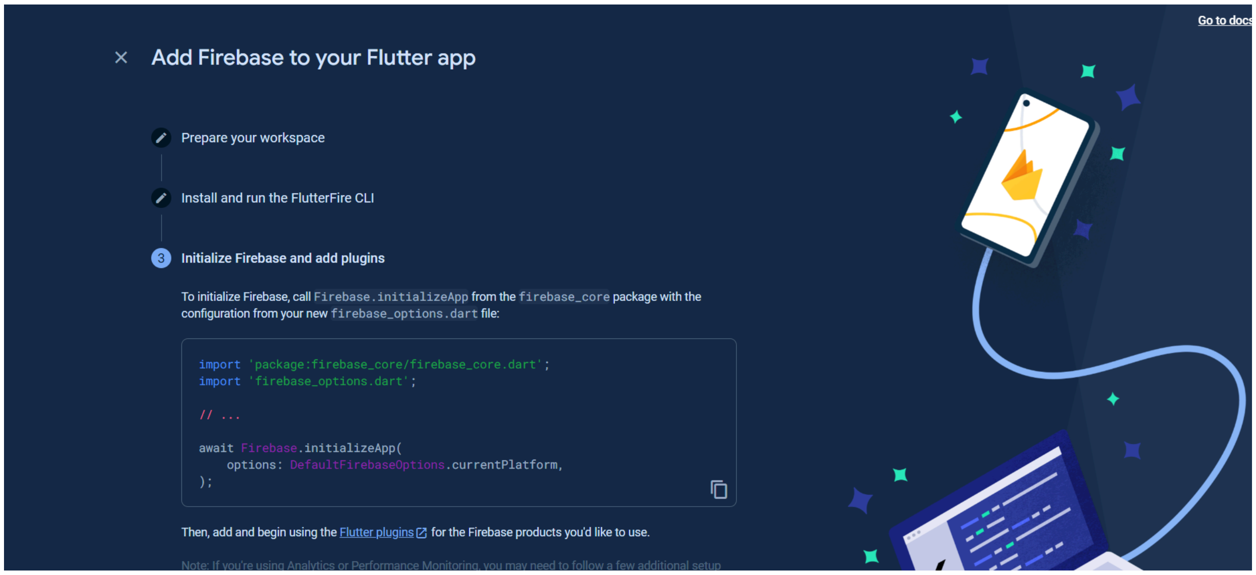
Google Analytics is a free and unlimited analytics solution that enables targeting, reporting, and more in Firebase Crashlytics, Cloud Messaging, In-App Messaging, Remote Config, A/B Testing, and Cloud Functions.

Google Analytics enables:

- ~~A/B testing~~
- ~~User segmentation & targeting across Firebase products~~
- ~~Breadcrumb logs in Crashlytics~~
- ~~Event-based Cloud Functions triggers~~
- ~~Free unlimited reporting~~

Enable Google Analytics for this project
Recommended

NAME:DIVIT WADHWANI EXP:6 ROLL NO:71



Add Firebase to your Flutter app

Prepare your workspace

Install and run the FlutterFire CLI

3 Initialize Firebase and add plugins

To initialize Firebase, call `Firebase.initializeApp` from the `firebase_core` package with the configuration from your new `firebase_options.dart` file:

```
import 'package:firebase_core/firebase_core.dart';
import 'firebase_options.dart';

// ...

await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
);
```

Then, add and begin using the Flutter plugins for the Firebase products you'd like to use.

Note: If you're using Analytics or Performance Monitoring, you may need to follow a few additional setup

Conclusion:
Upon successful completion of this lab experiment, you will have gained practical experience in integrating Firebase services with a Flutter application. This integration opens up possibilities for real-time data management, user authentication, and cloud storage, providing a robust and scalable backend infrastructure for your Flutter projects.