MPL Lab Exp 6

Name: Tanmay Bhosale Roll No. 08

Div: D15A Batch: A

Aim: To Connect Flutter UI with FireBase database

Theory:

Prerequisites

To complete this tutorial, you will need:

- A Google account to use Firebase.
- Developing for iOS will require XCode.
- To download and install Flutter.
- To download and install Android Studio and Visual Studio Code.
- It is recommended to install plugins for your code editor:
  - Flutter and Dart plugins installed for Android Studio.
  - Flutter extension installed for Visual Studio

Code. Create a Firebase Project:

First, log in with your Google account to manage your Firebase projects. From within the Firebase dashboard,

select the Create new project button and give it a name:

Go to the Firebase Console and create a new

project. Add your Flutter app to the Firebase project:

Register your app in the Firebase project, and follow the instructions to download the configuration files (google-services.json for Android, GoogleService-Info.plist for iOS).

The most important thing here is to match up the Android package name that you choose here with the one

inside of our application.

The structure consists of at least two segments. A common pattern is to use a domain name, a company

name, and the application name:

com.example.flutterfirebaseexample

Once you've decided on a name, open android/app/build.gradle in your code editor and update the applicationId to match

the Android package name:

android/app/build.gradle

...

defaultConfig {

// TODO: Specify your own unique Application ID

(https://developer.android.com/studio/build/application-id.html)

. applicationId 'com.example.flutterfirebaseexample'

...

} ...

Downloading the Config File

The next step is to add the Firebase configuration file into our Flutter project. This is important as it contains

the API keys and other critical information for Firebase to

use. Select Download google-services.json from this page:

2. Add Firebase to your Flutter project:

Add Dependencies:

Open your pubspec.yaml file and add the necessary dependencies:

yaml

Code:

main.dart

```
import
'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import
'package:jio_saavn_auth/firebase_options.dart';
import 'package:jio_saavn_auth/models/song.dart';
import
'package:jio_saavn_auth/screens/welcome_screen.dart';
import 'package:provider/provider.dart';

Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
```

```
);
// runApp(
// ChangeNotifierProvider(
```

```
  //    create: (context) => LikedSongsModel(),
  //    child: MyApp(),
  // ),
  // );
  runApp(
    MultiProvider(
      providers: [
        ChangeNotifierProvider(create: (context) => LikedSongsModel()),
        // Add other providers if needed
      ],
      child: MyApp(),
    ),
  );
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return ChangeNotifierProvider(
      create: (context) => LikedSongsModel(),
      child: MaterialApp(
        title: 'Spotify',
        home: WelcomePage(), // Use your main screen here
      ),
    );
  }
}
```

Firebase_options.dart:

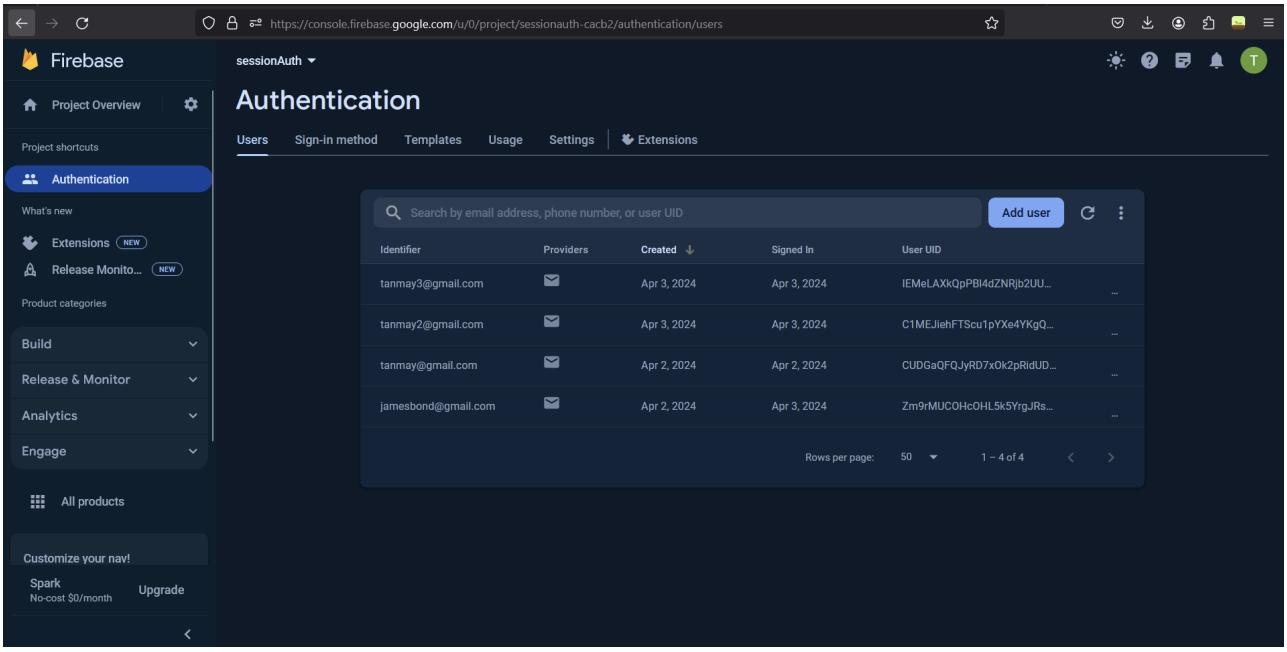
```
// File generated by FlutterFire CLI.
// ignore_for_file: lines_longer_than_80_chars,
avoid_classes_with_only_static_members
import 'package:firebase_core/firebase_core.dart' show FirebaseOptions;
```

```dart
import 'package:flutter/foundation.dart'
    show defaultTargetPlatform, kIsWeb, TargetPlatform;

/// Default [FirebaseOptions] for use with your Firebase apps.
///
/// Example:
/// ```dart
/// import 'firebase_options.dart';
/// // ...
/// await Firebase.initializeApp(
/// options: DefaultFirebaseOptions.currentPlatform,
/// );
/// ```
class DefaultFirebaseOptions {
  static FirebaseOptions get currentPlatform {
    if (kIsWeb) {
      throw UnsupportedError(
        'DefaultFirebaseOptions have not been configured for web - '
        'you can reconfigure this by running the FlutterFire CLI
        again.',
      );
    }
    switch (defaultTargetPlatform) {
      case TargetPlatform.android:
        return android;
      case TargetPlatform.iOS:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for ios - '
          'you can reconfigure this by running the FlutterFire CLI
          again.',
        );
      case TargetPlatform.macOS:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for macos -
```

```
  ' 'you can reconfigure this by running the FlutterFire CLI
  again.',
);
```

```
      case TargetPlatform.windows:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for windows - '
          'you can reconfigure this by running the FlutterFire CLI again.',
        );
      case TargetPlatform.linux:
        throw UnsupportedError(
          'DefaultFirebaseOptions have not been configured for linux - '
          'you can reconfigure this by running the FlutterFire CLI
          again.',
        );
      default:
        throw UnsupportedError(
          'DefaultFirebaseOptions are not supported for this platform.',
        );
    }
  }

  static const FirebaseOptions android = FirebaseOptions(
    apiKey: 'AIzaSyBCubxVCQHG3ieLoA11-3EzmohuAsMET8c',
    appId: '1:867583680837:android:81bfa85c82da3424599e2c',
    messagingSenderId: '867583680837',
    projectId: 'phone-authentication-ddb5c',
    storageBucket: 'phone-authentication-ddb5c.appspot.com',
  );
}
```

Firebase console:



Conclusion:

In this experiment, we have successfully connected firebase database and authenticated using google signin and email and password with out flutter application successfully.