

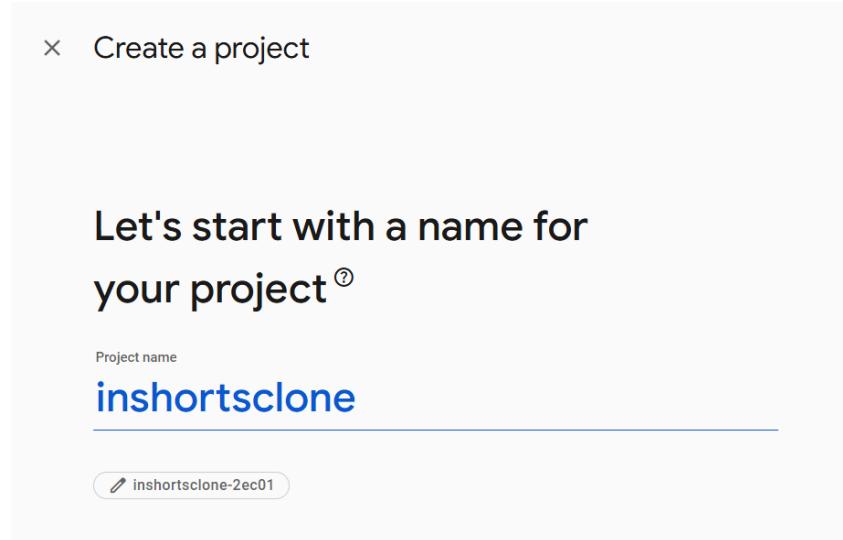
## **EXPERIMENT NO:- 6**

**Name:- Arnav Sawant      D15A      Roll-no:-52**

Aim:- To Connect flutter UI with firebase database

---

### **Creating a New 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

In order to add Android support to our Flutter application, select the Android logo from the dashboard. This brings us to the following screen:

1 Register app

Android package name ②  
com.company.appname

App nickname (optional) ②  
My Android App

Debug signing certificate SHA-1 (optional) ②  
00:00:00:00:00:00:00:00:00:00:00:00:00:00:  
 ⓘ Required for Dynamic Links, and Google Sign-In or phone number support in Auth.  
Edit SHA-1s in Settings.

**Register app**

The most important thing here is to match up the Android package name that you choose here with the one inside of our application.

Then download the google-services.json file, that you will get.

2 Download and then add config file

Instructions for Android Studio below | [Unity](#) | [C++](#)

[Download google-services.json](#)

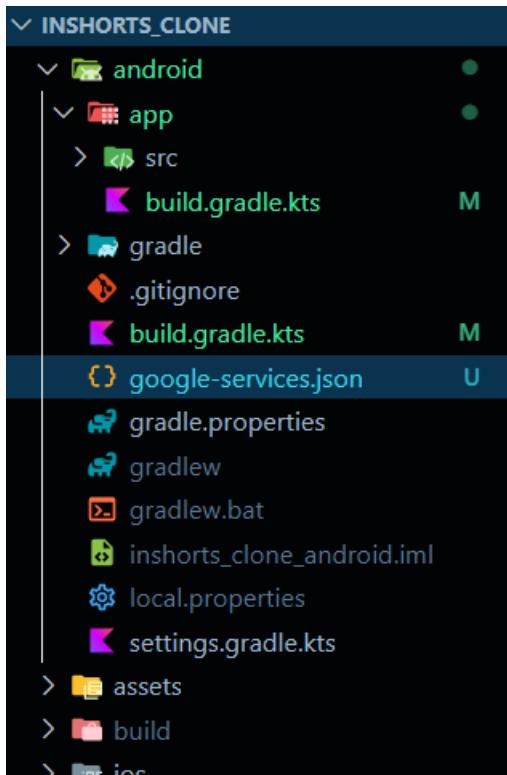
Switch to the Project view in Android Studio to see your project root directory.

Move your downloaded `google-services.json` file into your module (app-level) root directory.

The diagram shows a blue arrow pointing from a local file icon labeled "google-services.json" to the "Project" view in Android Studio. Inside the "Project" view, another blue arrow points from the "My Application [My Application]" module to the "app" module. The "app" module's root directory is highlighted, showing files like ".gradle", ".idea", "app", "libs", "src", ".gitignore", "build.gradle.kts", and "google-services.json".

[Next](#)

put that file in the android folder (root level)



then select the `build.gradle.kts` (Kotlin DSL) part, and then follow the rest instructions

### 3 Add Firebase SDK

Instructions for Gradle | [Unity](#) [C++](#)

★ Are you still using the `buildscript` syntax to manage plugins? Learn how to [add Firebase plugins](#) using that syntax.

1. To make the `google-services.json` config values accessible to Firebase SDKs, you need the Google services Gradle plugin.

Kotlin DSL (`build.gradle.kts`)  Groovy (`build.gradle`)

Add the plugin as a dependency to your **project-level** `build.gradle.kts` file:

**Root-level (project-level) Gradle file** (`<project>/build.gradle.kts`):

```
plugins {  
    // ...  
  
    // Add the dependency for the Google services Gradle plugin  
    id("com.google.gms.google-services") version "4.4.2" apply false Copy  
}
```

2. Then, in your **module (app-level)** `build.gradle.kts` file, add both the `google-services` plugin and any Firebase SDKs that you want to use in your app:

**Module (app-level) Gradle file** (`<project>/<app-module>/build.gradle.kts`):

```
plugins {  
    id("com.android.application") Copy  
    // Add the Google services Gradle plugin  
    id("com.google.gms.google-services") Copy  
    ...  
}  
  
dependencies {  
    // Import the Firebase BoM  
    implementation(platform("com.google.firebase:firebase-bom:33.9.0")) Copy  
  
    // TODO: Add the dependencies for Firebase products you want to use  
    // When using the BoM, don't specify versions in Firebase dependencies  
    // https://firebase.google.com/docs/android/setup#available-libraries  
}
```

By using the Firebase Android BoM, your app will always use compatible Firebase library versions. [Learn more](#) Copy

### 4 Next steps

You're all set!

Make sure to check out the [documentation](#) to learn how to get started with each Firebase product that you want to use in your app.

You can also explore [sample Firebase apps](#).

Or, continue to the console to explore Firebase.

[Previous](#)

[Continue to console](#)

Generate the `firebase_options.dart` file, based on the `google-services.json` file

```

import 'package:firebase_core/firebase_core.dart';

class DefaultFirebaseOptions {
    static FirebaseOptions get currentPlatform {
        return const FirebaseOptions(
            apiKey: "AIzaSyA_VIR7fj4d-b6tNzhw9qJ6GRRx5EXKqs0",
            appId: "1:388272292768:android:33180b2382688b18781ac5",
            messagingSenderId: "388272292768",
            projectId: "inshortsclone-848a9",
            storageBucket: "inshortsclone-848a9.firebaseiostorage.app",
            androidClientId: "1:388272292768:android:33180b2382688b18781ac5",
        );
    }
}

```

In your part select the sign-in method and enable it.

The screenshot shows the Firebase Authentication console interface. At the top, there's a dropdown menu labeled 'Inshortsclone'. Below it, a navigation bar has 'Authentication' selected. Underneath, there are tabs for 'Users', 'Sign-in method' (which is underlined in blue), 'Templates', 'Usage', 'Settings', and 'Extensions'. A sub-section titled 'Sign-in providers' is expanded, showing two options: 'Email/Password' and 'Email link (passwordless sign-in)'. Each option has an 'Enable' button with a checkmark. At the bottom right of this section are 'Cancel' and 'Save' buttons.

### Code:-

```

import 'package:flutter/material.dart';
import 'login/login.dart';
import 'profile/editprofile.dart';
import 'profile/profile.dart';
import 'settings.dart';
import 'search/searchpage.dart';
import 'home.dart'; // Import
HomeScreen
import
'package:firebase_core/firebase_core.
dart';
import 'firebase_options.dart';

```

```

void main() async {
    WidgetsFlutterBinding.ensureInitialized();
    try {
        await Firebase.initializeApp(
            options:
                DefaultFirebaseOptions.currentPlatform,
        );
        debugPrint('🔥 Firebase initialized
successfully!');
    }
}

```

```

        } catch (e) {
            debugPrint('Firebase initialization failed: $e');
        }
        runApp(MyApp());
    }

    class MyApp extends StatelessWidget {
        @override
        Widget build(BuildContext context) {
            return MaterialApp(
                debugShowCheckedModeBanner: false,
                home: HomeScreenWrapper(), // Use a wrapper for bottom navigation
            );
        }
    }

    class HomeScreenWrapper extends StatefulWidget {
        @override
        _HomeScreenWrapperState createState() => _HomeScreenWrapperState();
    }

    class _HomeScreenWrapperState extends State<HomeScreenWrapper> {
        int _selectedIndex = 1;

        final List<Widget> _pages = [
            SearchPage(),
            HomeScreen(), // Changed LoginPage to HomeScreen
            ProfileScreen(),
            SettingsPage(),
        ];
        void _onItemTapped(int index) {
    
```

```

            setState(() {
                _selectedIndex = index;
            });
        }

        @override
        Widget build(BuildContext context) {
            return Scaffold(
                body: IndexedStack(
                    index: _selectedIndex,
                    children: _pages,
                ),
                bottomNavigationBar: BottomNavigationBar(
                    items: const <BottomNavigationBarItem>[
                        BottomNavigationBarItem(
                            icon: Icon(Icons.search),
                            label: 'Search',
                        ),
                        BottomNavigationBarItem(
                            icon: Icon(Icons.home),
                            label: 'Home',
                        ),
                        BottomNavigationBarItem(
                            icon: Icon(Icons.account_circle),
                            label: 'Profile',
                        ),
                        BottomNavigationBarItem(
                            icon: Icon(Icons.settings),
                            label: 'Settings',
                        ),
                    ],
                    currentIndex: _selectedIndex,
                    selectedItemColor: Colors.blue,
                    unselectedItemColor: Colors.grey,
                    onTap: _onItemTapped,
                    type: BottomNavigationBarType.fixed,
    
```

```
        },
    }
);
```

Once you run this code this should be your output, if the firebase setup is successful , it would print “Firebase initialized successfully!”.

```
[3]: Edge (edge)
Please choose one (or "q" to quit): 2
Launching lib/main.dart on Chrome in debug mode...
Waiting for connection from debug service on Chrome...          20.9s
This app is linked to the debug service: ws://127.0.0.1:56889/EEaa4qKHJvU=/ws
Debug service listening on ws://127.0.0.1:56889/EEaa4qKHJvU=/ws

To hot restart changes while running, press "r" or "R".
For a more detailed help message, press "h". To quit, press "q".

A Dart VM Service on Chrome is available at: http://127.0.0.1:56889/EEaa4qKHJvU=
The Flutter DevTools debugger and profiler on Chrome are available at: http://127.0.0.1:9101?uri=http://127.0.0.1:56889/EEaa4qKHJvU=
  Firebase initialized successfully!
```