Experiment 6

Aim: To Connect Flutter UI with fireBase database

Theory:

Firebase is a comprehensive mobile and web application development platform provided by Google. It offers a wide range of features and services that enable developers to build high-quality apps quickly and efficiently.

At its core, Firebase provides tools for various aspects of app development, including authentication, real-time database, cloud storage, hosting, and more. Here's a breakdown of some key Firebase components:

Authentication: Firebase Authentication allows developers to easily integrate secure authentication methods into their apps. It supports various authentication methods, including email/password, phone number, Google, Facebook, Twitter, and more. This feature handles user management, authentication, and security, making it simple to add user sign-up, sign-in, and access control to apps.

Realtime Database: Firebase Realtime Database is a NoSQL cloud database that enables developers to store and sync data in real time. It's particularly useful for applications that require real-time updates, such as chat apps, collaboration tools, and gaming apps. The database is JSON-based, which makes it flexible and easy to use. It automatically synchronizes data across all connected clients and devices, ensuring that changes made by one user are immediately reflected on others' screens.

Cloud Firestore: Cloud Firestore is Firebase's newer, more scalable NoSQL database solution. It offers more powerful querying capabilities, better scalability, and improved performance compared to the Realtime Database. Firestore organizes data into collections and documents, allowing for more structured and efficient data storage. It also supports real-time updates, offline data access, and powerful querying capabilities.

Cloud Storage: Firebase Cloud Storage provides secure and reliable cloud storage for user-generated content, such as images, videos, and documents. It allows developers to easily upload and download files from their apps using simple APIs. Cloud Storage integrates seamlessly with other Firebase services, making it easy to store and serve user-generated content in Firebase-powered apps.

Hosting: Firebase Hosting allows developers to quickly and securely deploy web apps and static content to a global content delivery network (CDN). It provides fast and reliable hosting with SSL encryption, custom domain support, and automatic scaling. With Firebase Hosting, developers can deploy web apps with just a few simple commands, eliminating the need for complex server configurations and maintenance.

Cloud Functions: Firebase Cloud Functions allow developers to run backend code in response to events triggered by Firebase features and HTTPS requests. It enables developers to extend the functionality of their apps without managing servers or infrastructure. Cloud Functions are written in JavaScript or TypeScript and can be deployed with a single command using the Firebase CLI (Command Line Interface).

Analytics: Firebase Analytics provides insights into app usage and user behavior, helping developers understand how users interact with their apps. It tracks key metrics such as active users, retention, engagement, and conversion, allowing developers to make informed decisions about app improvements and optimizations.

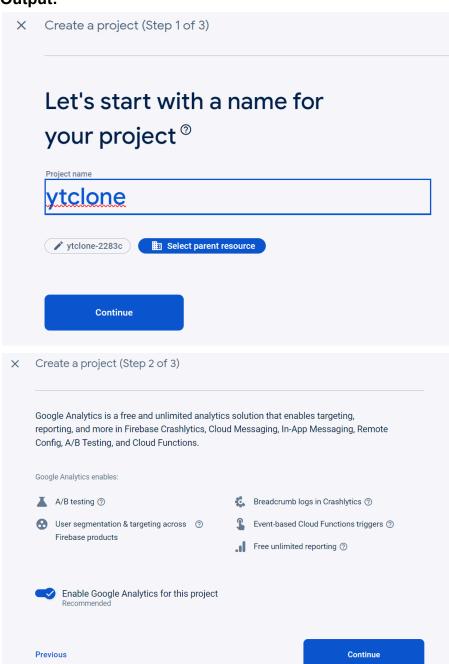
Performance Monitoring: Firebase Performance Monitoring helps developers identify and fix performance issues in their apps. It provides detailed insights into app performance, including app startup time, network latency, and UI responsiveness. Performance Monitoring helps developers optimize their apps for better user experiences and higher ratings.

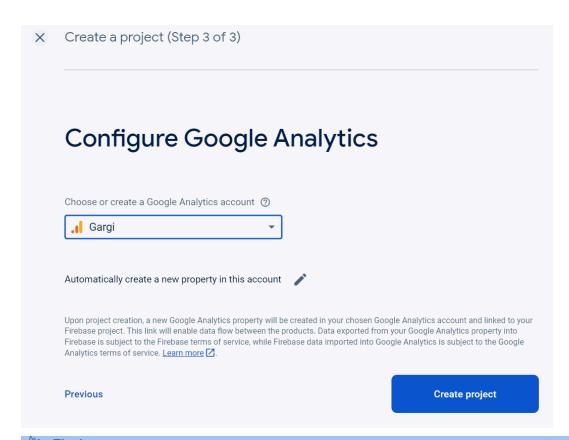
In addition to these core features, Firebase offers many other services, including Cloud Messaging for push notifications, Remote Config for dynamic app configuration, Machine Learning for integrating machine learning models into apps, and more. Overall, Firebase provides a powerful and comprehensive platform for building, managing, and scaling mobile and web applications.

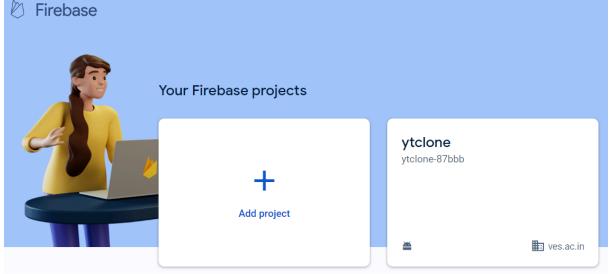
Code in main.dart:

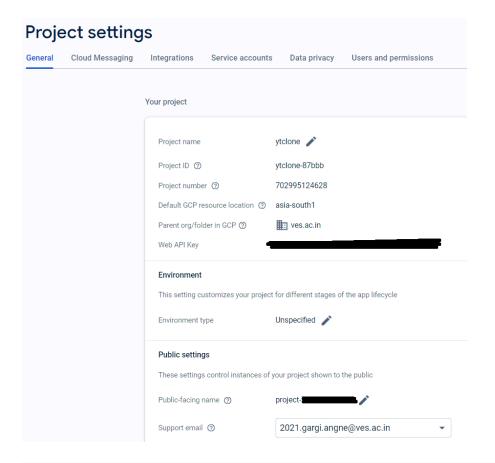
```
import 'package:firebase core/firebase core.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:flutter riverpod/flutter riverpod.dart';
import 'package:youtube/login.dart';
void main() async{
WidgetsFlutterBinding.ensureInitialized();
try {
    await Firebase.initializeApp(
     options: FirebaseOptions(
      apiKey: "AlzaSyAyMitBa5a5qF9SSVLyeeu3aZUce f y3c", // paste your api key here
      appld: "1:702995124628:android:119b3a052e863e5f757f81", //paste your app id here
      messagingSenderId: "702995124628", //paste your messagingSenderId here
      projectId: "ytclone-87bbb", //paste your project id here
     ),
 );
    runApp(ProviderScope(child: MyApp()));
} catch (e) {
 print('Error initializing Firebase: $e');
 // Handle the error gracefully, possibly show an error dialog or fallback screen
}
}
class MyApp extends StatelessWidget {
@override
Widget build(BuildContext context) {
 SystemChrome.setPreferredOrientations([DeviceOrientation.portraitUp]);
  return MaterialApp(
   title: 'Flutter YouTube UI',
   debugShowCheckedModeBanner: false,
   theme: ThemeData(
    brightness: Brightness.dark,
    bottomNavigationBarTheme:
    const BottomNavigationBarThemeData(selectedItemColor: Colors.white),
   home: GoogleSignInScreen(),
 );
}
}
```

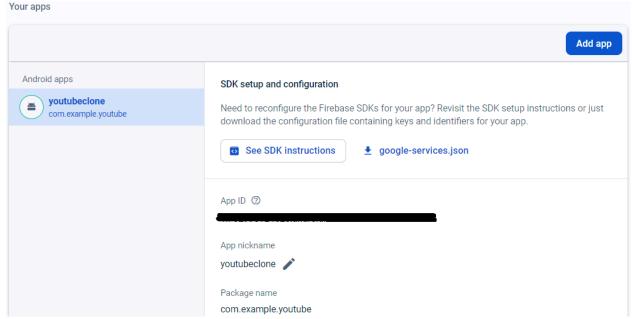
Output:

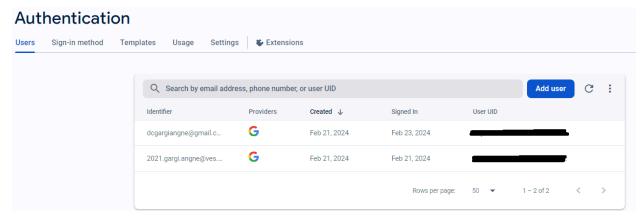


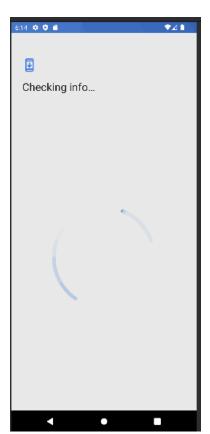




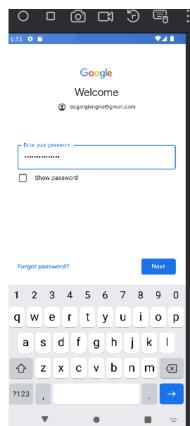


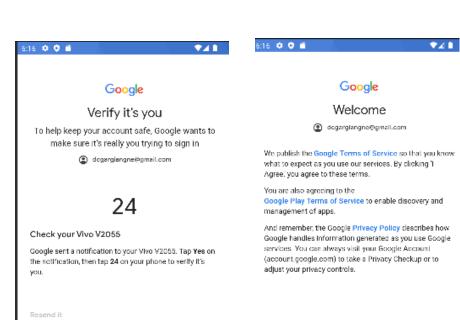












Try another way

Lagra