

Experiment-6

Aim : To connect Flutter UI with Firebase.

Theory :

Google Firebase is a set of cloud-based development tools that helps mobile application developers build, deploy and scale their apps.

Firebase provides a variety of features, including the following:

- **Authentication-** Firebase provides a secure and easy way for users to sign into their app. Developers can use Firebase Authentication to support email and password login, Google Sign-In, Facebook Login and more.
- **Realtime Database-** The Firebase Realtime Database is a cloud-hosted NoSQL database that lets organizations store and sync data in real time across all of their users' devices. This makes it easy to build apps that are always up to date, even when users are offline.
- **Cloud Messaging-** Firebase Cloud Messaging (FCM) is a service that lets businesses send messages to their users' devices, even if they're not using the app. Developers can use FCM to send push notifications, update app content, and more.
- **Crashlytics-** Firebase Crashlytics is a service that helps organizations track and fix crashes in their app. Crashlytics provides detailed reports on crashes, so they can quickly identify the root cause and fix the problem.
- **Performance Monitoring.** Firebase Performance Monitoring provides insights into the performance of their app. Organizations can use Performance Monitoring to track metrics like CPU usage, memory usage and network traffic.
- **Test Lab.** Firebase Test Lab is a cloud-based service that lets developers test their app on a variety of devices and configurations. This helps them ensure the app works well on a variety of devices and in different network conditions.

Execution Steps:

How To Set Up Firebase with Flutter for Android Apps:

Firebase is a great backend solution for anyone that wants to use authentication, databases, cloud functions, ads, and countless other features within an app.

In this article, you will create a Firebase project for Android platforms using Flutter.

Prerequisites:

To complete this tutorial, you will need:

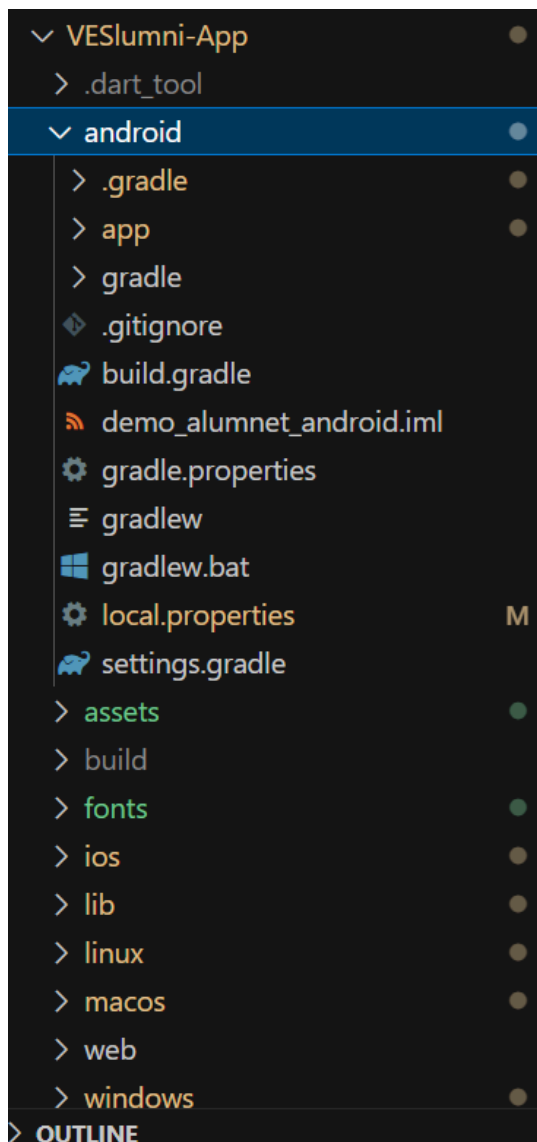
- A Google account to use Firebase.
- Developing for iOS will require XCode.
- To download and install Flutter.
- To download and install Android Studio and Visual Studio Code.
- It is recommended to install plugins for your code editor:
 - o [Flutter](#) and [Dart](#) plugins installed for Android Studio.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

This tutorial was verified with Flutter v2.0.6, Android SDK v31.0.2, and Android Studio v4.1.

Creating a New Flutter Project:

Once you have your environment set up for Flutter, you can run the following to create a new application:

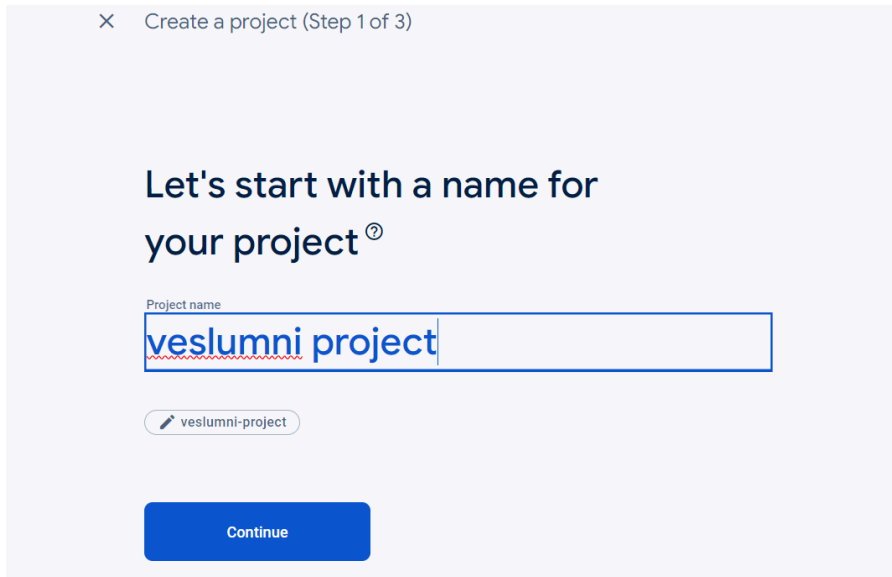
Using `flutter create` will produce a demo application that will display the number of times a button is clicked.



Now that we've got a Flutter project up and running, we can add Firebase.

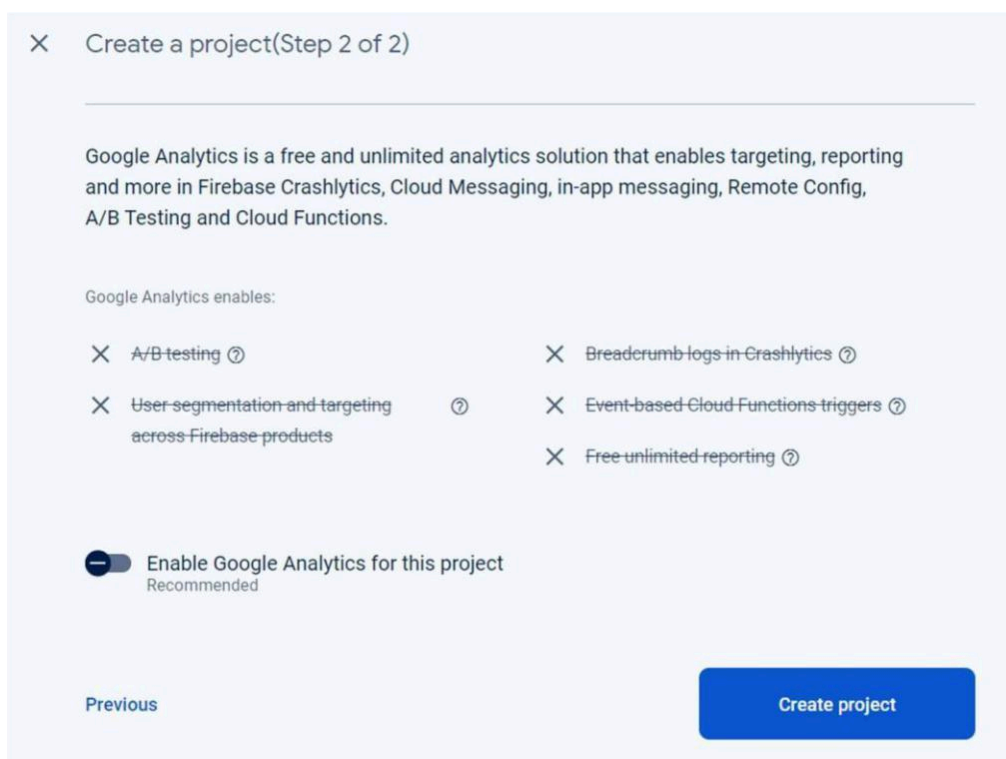
Creating a New Firebase Project:

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:



The screenshot shows the 'Create a project (Step 1 of 3)' dialog. At the top, there's a close button (X) and the title 'Create a project (Step 1 of 3)'. Below this, the text 'Let's start with a name for your project' is displayed with a help icon. A text input field labeled 'Project name' contains the text 'veslumni project'. Below the input field, there's a small button with a pencil icon and the text 'veslumni-project'. At the bottom, there is a large blue 'Continue' button.

Next, we're given the option to enable Google Analytics. This tutorial will not require Google Analytics, but you can also choose to add it to your project.



The screenshot shows the 'Create a project (Step 2 of 2)' dialog. At the top, there's a close button (X) and the title 'Create a project (Step 2 of 2)'. Below this, a horizontal line separates the title from the main content. The main content starts with a paragraph: '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.' Below this, the text 'Google Analytics enables:' is followed by a list of features, each with a close button (X) and a help icon (?):

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

At the bottom, there is a toggle switch labeled 'Enable Google Analytics for this project' with the text 'Recommended' below it. The toggle is currently turned off. At the very bottom, there are two buttons: 'Previous' on the left and 'Create project' on the right.

If you choose to use Google Analytics, you will need to review and accept the terms and conditions prior to project creation.

After pressing Continue, your project will be created and resources will be provisioned. You will then be directed to the dashboard for the new project.

Adding Android support

Registering the App

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

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.alumni_app”
```

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

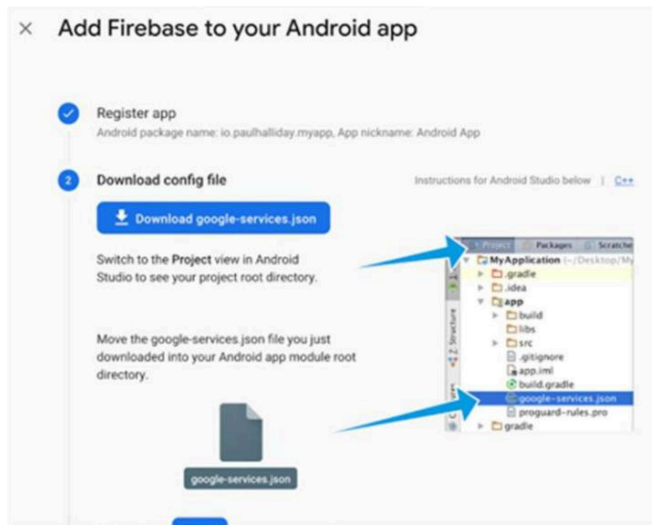
defaultConfig {
    // TODO: Specify your own unique Application ID
    (https://developer.android.com/studio/build/application-id.html).
    applicationId "com.example.alumni_app"
}

// You can update the following values to match your application needs.
```

You can skip the app nickname and debug signing keys at this stage. Select Register app to continue.

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:



Next, move the `google-services.json` file to the `android/app` directory within the Flutter project.

Adding the Firebase SDK:

We'll now need to update our Gradle configuration to include the Google Services plugin.

Open `android/build.gradle` in your code editor and modify it to include the following:

```
android/build.gradle

buildscript {
    ext.kotlin_version =
'1.7.10'    repositories {
    google()
        mavenCentral()
    }

    dependencies {        classpath "org.jetbrains.kotlin-
gradle-plugin:$kotlin_version" classpath
'com.google.gms:google-services:4.3.15'
```

```
}  
}
```

Finally, update the app level `android/app/build.gradle` to include the following:
file at `android/app/build.gradle`

```
plugins { id
```

```
"com.android.application" id
```

```
"kotlin-android"    id
```

```
"dev.flutter.flutter-gradle-plugin"
```

```
id 'com.google.gms.google-
```

```
services'
```

```
}
```

```
...
```

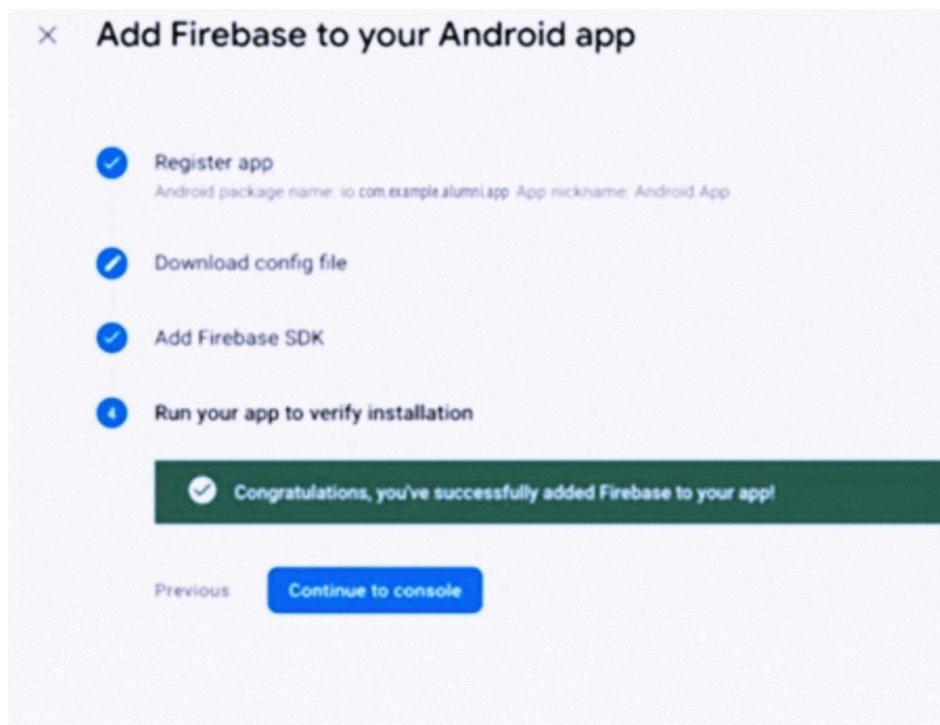
```
dependencies { implementation
```

```
platform('com.google.firebase:firebase-bom:32.7.3')
```

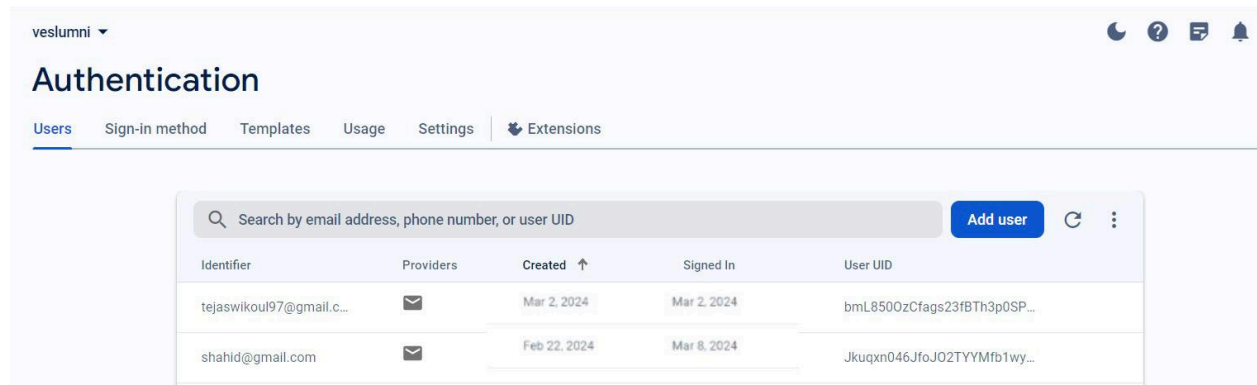
```
}
```

With this update, we're essentially applying the Google Services plugin as well as looking at how other Flutter Firebase plugins can be activated such as Analytics.

From here, run your application on an Android device or simulator. If everything has worked correctly, you should get the following message in the dashboard:



LOGIN/SIGNUP USING FIREBASE(AUTHENTICATION)



```
import 'package:demo_alumnet/components/widgets.dart';
import 'package:demo_alumnet/services/auth/auth_service.dart';
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';

class LoginPage extends StatefulWidget {
  final void Function()? onTap;
  const LoginPage({Key? key, required this.onTap});

  @override
  State<LoginPage> createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage> {
  // Text controllers
  final emailController = TextEditingController();
  final passwordController = TextEditingController();

  // Sign in user
  Future<void> signIn() async {
    // Get the auth service
    final authService = Provider.of<AuthService>(context, listen: false);
```

```

try {
  await authService.signInWithEmailAndPassword(
    emailController.text, passwordController.text);
} catch (e) {
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      content: Text(
        e.toString(),
      ),
    ),
  );
}
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Container(
      decoration: BoxDecoration(
        gradient: LinearGradient(
          begin: Alignment.topCenter,
          end: Alignment.bottomCenter,
          colors: [Colors.deepPurple[200]!, Colors.deepPurple[400]!],
        ),
      ),
    child: SafeArea(
      child: Center(
        child: Padding(
          padding: const EdgeInsets.symmetric(horizontal: 25.0),
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              // Stack to overlay circular logo on the gradient background
              Stack(
                alignment: Alignment.center,
                children: [
                  // Circular logo image
                  CircleAvatar(
                    radius: 60,
                    backgroundImage: AssetImage('assets/Vesitlogo.png'),
                  ),

```

```
    ],  
  ),  
  
  const SizedBox(  
    height: 10,  
  ),  
  
  const Text(  
    'VESlumni',  
    style: TextStyle(fontWeight: FontWeight.bold),  
  ),  
  
  const SizedBox(  
    height: 20,  
  ),  
  
  const Text(  
    'Welcome back, student!!',  
    style: TextStyle(fontWeight: FontWeight.w500),  
  ),  
  
  const SizedBox(  
    height: 50,  
  ),  
  
  // Email  
  MyTextField(  
    controller: emailController,  
    hintText: 'Enter your email',  
    labelText: 'Email',  
    obscureText: false,  
  ),  
  
  const SizedBox(  
    height: 20,  
  ),  
  
  // Password  
  MyTextField(  
    controller: passwordController,  
    hintText: 'Password',
```

```

        labelText: 'Password',
        obscureText: true,
      ),

      const SizedBox(
        height: 20,
      ),

      // Sign in button
      MyCustomBtn(onTap: signIn, text: "Sign In"),

      const SizedBox(
        height: 40,
      ),

      // Already a member? Sign In
      Row(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          const Text('Not a member?'),
          const SizedBox(
            width: 4,
          ),
          GestureDetector(
            onTap: widget.onTap,
            child: const Text(
              'Register Now',
              style: TextStyle(
                fontWeight: FontWeight.bold,
              ),
            ),
          ),
        ],
      ),
    ],
  ),
),
),
),
),
),
);

```

```
}  
}
```

SIGNUP-

```
import 'package:cloud_firestore/cloud_firestore.dart';  
import 'package:demo_alumnet/services/auth/auth_service.dart';  
import 'package:demo_alumnet/widgets/helper_functions.dart';  
import 'package:firebase_auth/firebase_auth.dart';  
import 'package:flutter/material.dart';  
import 'package:demo_alumnet/components/widgets.dart';  
import 'package:provider/provider.dart';  
  
class RegisterPage extends StatefulWidget {  
  final void Function()? onTap;  
  const RegisterPage({super.key, required this.onTap});  
  
  @override  
  State<RegisterPage> createState() => _RegisterPageState();  
}  
  
class _RegisterPageState extends State<RegisterPage> {  
  //text controllers  
  final nameController = TextEditingController();  
  final emailController = TextEditingController();  
  final passwordController = TextEditingController();  
  final confirmPasswordController = TextEditingController();  
  final bioController = TextEditingController();  
  
  // sign up user  
  Future<void> signUp() async {  
    // show loading circle  
    // showDialog(  
    //   context: context,  
    //   builder: (context) => const Center(  
    //     child: CircularProgressIndicator(),  
    //   ),  
    // );  
  }  
}
```

```

// check password match
if (passwordController.text != confirmPasswordController.text) {
    Navigator.pop(context);

    displayMessageToUser("Password do not match", context);
    return;
}

//get the auth service
final authService = Provider.of<AuthService>(context, listen: false);

try {
    await authService.signUpWithEmailAndPassword(emailController.text,
        passwordController.text, nameController.text, bioController.text);

    // create user credentials
} catch (e) {
    ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
            content: Text(
                e.toString(),
            ),
        ),
    );
}
}

// create user document and collect them in cloud firestore
Future<void> createUserDocument(UserCredential? userCredential) async {
    if (userCredential != null && userCredential.user != null) {
        await FirebaseFirestore.instance
            .collection("Users")
            .doc(userCredential.user!.email)
            .set({
                'email': userCredential.user!.email,
                'username': nameController.text,
                'bio': "Enter your Bio",
            });
    }
}
}

```

```

@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Colors.deepPurple[200],
    body: SafeArea(
      child: Center(
        child: SingleChildScrollView(
          child: Padding(
            padding: const EdgeInsets.symmetric(horizontal: 25.0),
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                // Circular logo image
                CircleAvatar(
                  radius: 60,
                  backgroundImage: AssetImage('assets/Vesitlogo.png'),
                ),

                const SizedBox(
                  height: 10,
                ),

                const Text(
                  'VESlumni',
                  style: TextStyle(fontWeight: FontWeight.bold),
                ),

                const SizedBox(
                  height: 20,
                ),

                const Text(
                  'Come connect with your alumni',
                  style: TextStyle(fontWeight: FontWeight.w500),
                ),

                const SizedBox(
                  height: 50,
                ),

                //name

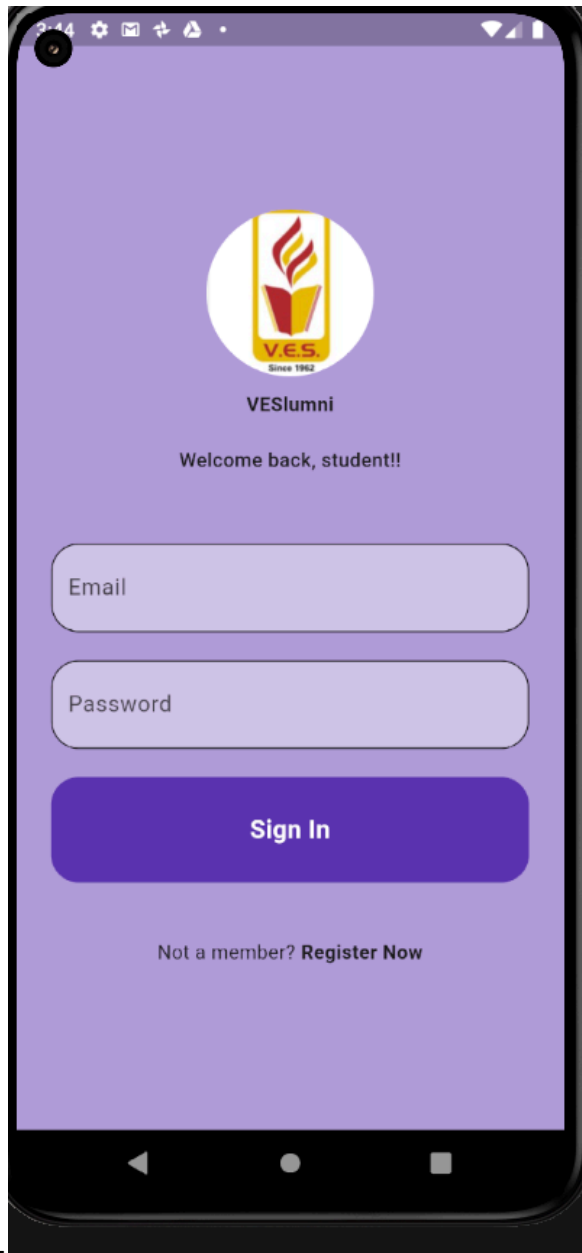
```

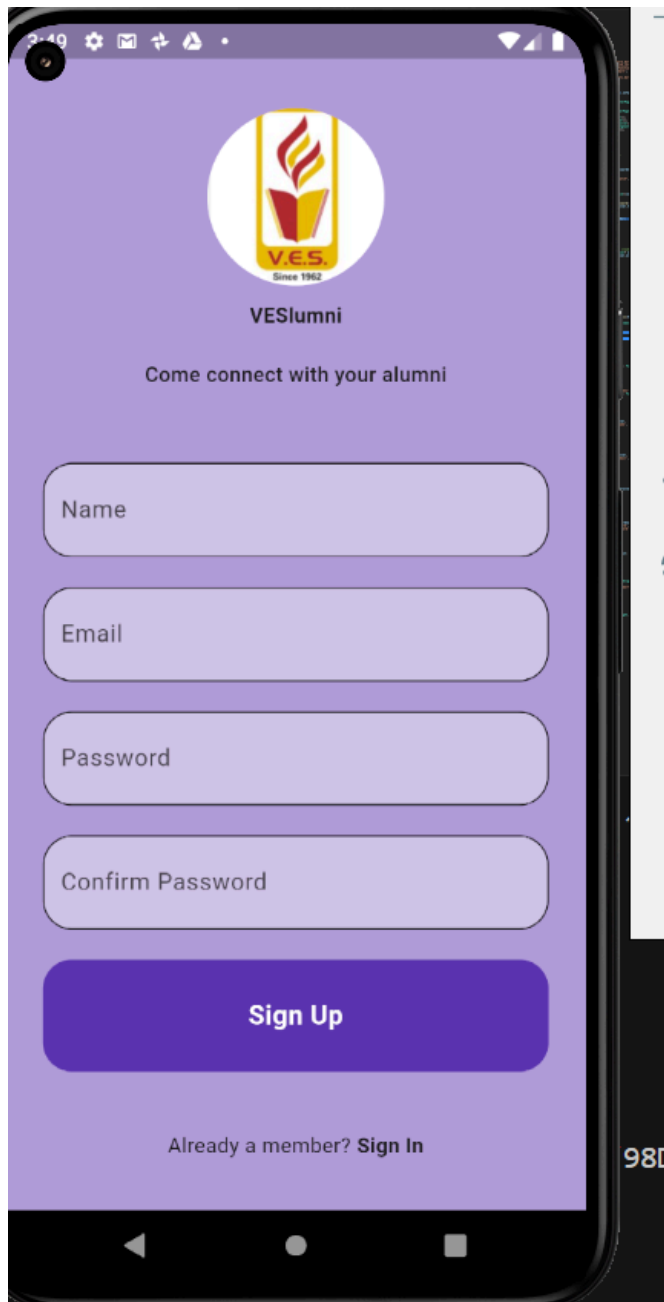
```
MyTextField(  
  controller: nameController,  
  hintText: 'Enter your name',  
  labelText: 'Name',  
  obscureText: false),  
  
const SizedBox(  
  height: 20,  
) ,  
  
//email  
MyTextField(  
  controller: emailController,  
  hintText: 'Enter your email',  
  labelText: 'Email',  
  obscureText: false),  
  
const SizedBox(  
  height: 20,  
) ,  
  
//password  
MyTextField(  
  controller: passwordController,  
  hintText: 'Password',  
  labelText: 'Password',  
  obscureText: true),  
  
const SizedBox(  
  height: 20,  
) ,  
  
//confirm password  
MyTextField(  
  controller: confirmPasswordController,  
  hintText: 'Confirm Password',  
  labelText: 'Confirm Password',  
  obscureText: true),  
  
const SizedBox(  
  height: 20,
```



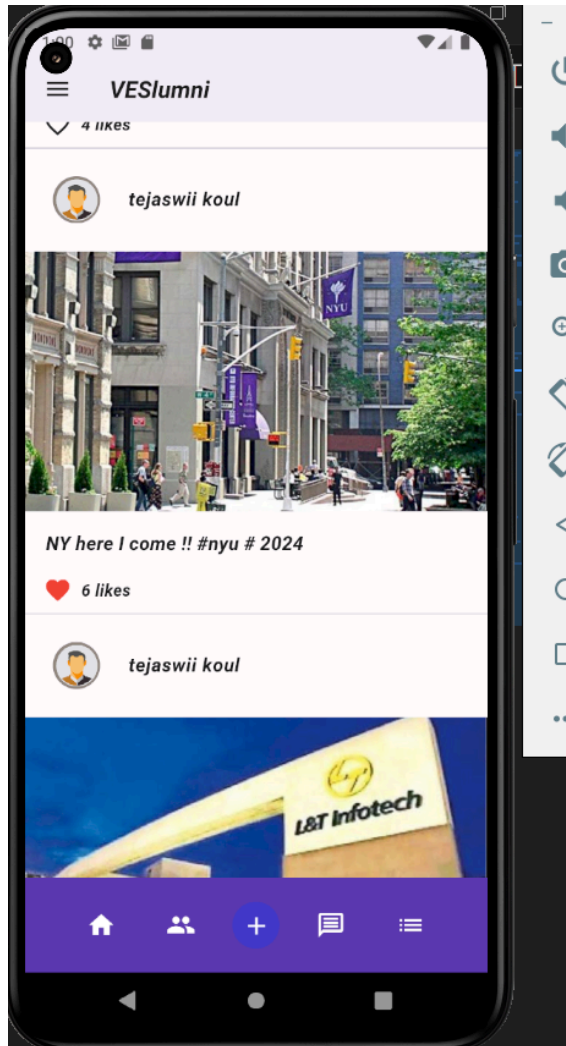
```
),  
  
    //sign up btn  
    MyCustomBtn(onTap: signUp, text: "Sign Up"),  
  
    const SizedBox(  
      height: 40,  
    ),  
  
    //Already a member? Sign In  
  
    Row(  
      mainAxisAlignment: MainAxisAlignment.center,  
      children: [  
        const Text('Already a member?'),  
        const SizedBox(  
          width: 4,  
        ),  
        GestureDetector(  
          onTap: widget.onTap,  
          child: const Text(  
            'Sign In',  
            style: TextStyle(  
              fontWeight: FontWeight.bold,  
            ),  
          ),  
        ),  
      ],  
    ),  
  ],  
),  
),  
),  
),  
),  
),  
);  
}  
}
```

OUTPUT-





After successful login we r redirected to the home page-



Conclusion- In this experiment, we grasped the concept of Firebase and seamlessly integrated it into our Flutter application by installing necessary packages and incorporating essential dependencies. Subsequently, we crafted a login page, signup page to assess the user's authentication status and provide a convenient signout option. Leveraging Firebase authentication, we implemented the login functionality, ensuring a smooth transition to the home page of our Alumni App upon successful authentication.

