EXPERIMENT 6

Aim : To connect Flutter app with Firebase

Theory:

Firebase is a comprehensive platform for building mobile and web applications developed by Google. It provides a wide range of backend services and tools to help developers build high-quality apps more efficiently. Here's an overview of the key components and features of Firebase:

Firebase Features:
Realtime Database: Firebase Realtime Database is a cloud-hosted NoSQL database that allows you to store and sync data in real-time across multiple clients. It's ideal for applications that require real-time updates, such as chat apps or collaborative tools.

Cloud Firestore: Firestore is a flexible, scalable database for mobile, web, and server development. It offers more powerful querying capabilities, offline support, and automatic syncing across devices. Firestore is often preferred for larger and more complex applications.

Authentication: Firebase Authentication provides easy-to-use SDKs and ready-to-use UI libraries to authenticate users using email/password, phone number, Google, Facebook, Twitter, and more. It handles identity verification and securely manages user accounts.

Cloud Storage: Firebase Cloud Storage allows you to store and serve user-generated content such as photos, videos, and files. It offers secure uploads and downloads, powerful access control, and integration with Firebase Authentication.

Cloud Functions: Firebase Cloud Functions lets you run backend code in response to events triggered by Firebase features and HTTPS requests. You can write serverless functions to handle tasks like sending notifications, processing data, or integrating with third-party services.

Hosting: Firebase Hosting provides fast and secure web hosting for your static and dynamic content. You can deploy web apps and static websites with a single command, and Firebase takes care of SSL, CDN, and caching for improved performance.

Analytics: Firebase Analytics helps you understand user behavior and app performance with comprehensive analytics and reporting tools. You can track user engagement, retention, conversion rates, and more to optimize your app and marketing strategies.

Cloud Messaging: Firebase Cloud Messaging (FCM) enables you to send targeted notifications and messages to your app users across platforms (iOS, Android, and web). You can send notifications based on user segments, user actions, or specific topics.

Remote Config: Firebase Remote Config allows you to dynamically customize and personalize your app's behavior and appearance without requiring app updates. You can modify app

configurations, feature flags, and UI elements remotely to improve user experience and test new features.

Performance Monitoring: Firebase Performance Monitoring helps you identify and fix performance issues in your app by measuring app startup time, network latency, and UI rendering performance. It provides insights and metrics to optimize app performance and user experience.

App Distribution: Firebase App Distribution simplifies the process of distributing pre-release versions of your app to testers and stakeholders. It supports both iOS and Android platforms and provides feedback and crash reporting for better testing and debugging.

Benefits of Firebase:
Integrated Platform: Firebase offers a unified platform with all the backend services and tools you need to build, scale, and monitor your app.

Scalability: Firebase services are built on Google's infrastructure and can scale automatically to handle large user bases and traffic spikes.

Real-Time Updates: Firebase provides real-time data synchronization across clients, enabling collaborative and interactive app experiences.

Simplified Development: Firebase SDKs and libraries offer simple APIs and easy-to-use tools to streamline app development and reduce development time.

No Server Management: With Firebase, you can focus on building your app without worrying about server maintenance, infrastructure setup, or backend management.

Built-in Security: Firebase offers built-in security features such as authentication, data encryption, and access control to protect your app and user data.

Analytics and Insights: Firebase provides powerful analytics and monitoring tools to track app usage, user engagement, and performance metrics, helping you make data-driven decisions and optimize your app.

Cost-Effective: Firebase offers a generous free tier and flexible pricing plans based on usage, making it accessible to startups and small businesses.

Overall, Firebase is a powerful and versatile platform that empowers developers to build high-quality apps with rich features, real-time functionality, and robust backend infrastructure. Whether you're building a simple mobile app or a complex web application, Firebase provides the tools and services you need to succeed.

Code:

Login Screen:

```
import 'package:blogapp/screens/home_screen.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import '../components/round_button.dart';
import '../services/notification_service.dart';

class LoginScreen extends StatefulWidget {
 const LoginScreen({super.key});

 @override
 State<LoginScreen> createState() => _LoginScreenState();
}

class _LoginScreenState extends State<LoginScreen> {
 FirebaseAuth _auth = FirebaseAuth.instance;
 NotificationServices _notificationService = NotificationServices();

 TextEditingController emailController = TextEditingController();
 TextEditingController passwordController = TextEditingController();

 final _formkey = GlobalKey<FormState>();
 String email = "", password = "";

 @override
 Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      backgroundColor: Colors.deepOrange,
      title: const Text('Login', style: TextStyle(color: Colors.white)),
    ),
    body: Container(
      color: Colors.orange[100],
      child: Padding(
        padding: const EdgeInsets.symmetric(horizontal: 20),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.center,
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            const Text(
              'Login',
```

```
  style: TextStyle(fontWeight: FontWeight.bold, fontSize: 25),
),
Padding(
  padding: const EdgeInsets.symmetric(vertical: 30),
  child: Form(
    key: _formkey,
    child: Column(
      children: [
        TextFormField(
          controller: emailController,
          keyboardType: TextInputType.emailAddress,
          decoration: const InputDecoration(
              hintText: 'Email',
              labelText: 'Email',
              prefixIcon: Icon(Icons.email),
              border: OutlineInputBorder()),
          onChanged: (String value) {
            email = value;
          },
          validator: (value) {
            return value!.isEmpty ? 'Enter Email' : null;
          },
        ),
        Padding(
          padding: const EdgeInsets.symmetric(vertical: 30),
          child: TextFormField(
            controller: passwordController,
            keyboardType: TextInputType.visiblePassword,
            obscureText: true,
            decoration: const InputDecoration(
                hintText: 'Password',
                labelText: 'Password',
                prefixIcon: Icon(Icons.lock),
                border: OutlineInputBorder()),
            onChanged: (String value) {
              password = value;
            },
            validator: (value) {
              return value!.isEmpty ? 'Enter Password' : null;
            },
          ),
        ),
        RoundButton(
            title: 'SignIn',
            onPress: () async {
```

```
if (_formkey.currentState!.validate()) {
  try {
    final user = await _auth
        .signInWithEmailAndPassword(
      email: email.toString().trim(),
      password: password.toString().trim(),
    );

    if (user != null) {
      print('Success');
      ScaffoldMessenger.of(context)
          .showSnackBar(
        const SnackBar(
          content:
          Text('Login Successful'),
          duration: Duration(seconds: 4),
          backgroundColor:
          Colors.deepOrange,
        ),
      );

      // Send FCM notification
      // //await _notificationService
      //     .sendLoginNotification();

      Navigator.push(
          context,
          MaterialPageRoute(
              builder: (context) =>
                  const HomeScreen()));
    }
  } catch (e) {
    print(e.toString());
    ScaffoldMessenger.of(context)
        .showSnackBar(
      SnackBar(
        content: Text('Login Failed: $e'),
        duration: const Duration(seconds: 4),
        backgroundColor:
        Colors.deepOrange,
      ),
    );
  }
 }
}),
```

```
                ],
              ),
            ),
          )
        ],
      ),
    ),
  ),
 );
}
}
```

Upload Blog Screen:

```
import 'dart:io';
import 'package:flutter/material.dart';
import 'package:blogapp/components/round_button.dart';
import 'package:blogapp/services/firebase_service.dart';
import 'package:image_picker/image_picker.dart';

class AddPostScreen extends StatefulWidget {
 const AddPostScreen({Key? key}) : super(key: key);

 @override
 State<AddPostScreen> createState() => _AddPostScreenState();
}

class _AddPostScreenState extends State<AddPostScreen> {
 final FirebaseService firebaseService = FirebaseService();
 File? _image;
 TextEditingController titleController = TextEditingController();
 TextEditingController descriptionController = TextEditingController();

 final _formkey = GlobalKey<FormState>();
 String title = "", description = "";

 Future<void> getImageGallery() async {
  final pickedFile = await ImagePicker().pickImage(source: ImageSource.gallery);
  setState(() {
    if (pickedFile != null) {
      _image = File(pickedFile.path);
    } else {
      print('No image selected');
    }
  });
 }
```

```
Future<void> getImageCamera() async {
  final pickedFile = await ImagePicker().pickImage(source: ImageSource.camera);
  setState(() {
    if (pickedFile != null) {
      _image = File(pickedFile.path);
    } else {
      print('No image selected');
    }
  });
}

Future<void> uploadBlog() async {
  if (_formkey.currentState!.validate()) {
    try {
      await firebaseService.uploadBlog(
        title: titleController.text,
        description: descriptionController.text,
        image: _image,
      );

      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(
          content: Text('Blog Uploaded'),
          duration: Duration(seconds: 4),
          backgroundColor: Colors.deepOrange,
        ),
      );
    } catch (error) {
      print(error.toString());
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          content: Text('Could not upload the blog: $error'),
          duration: Duration(seconds: 4),
          backgroundColor: Colors.deepOrange,
        ),
      );
    }
  }
}

void dialog(context) {
  showDialog(
    context: context,
    builder: (BuildContext context) {
```

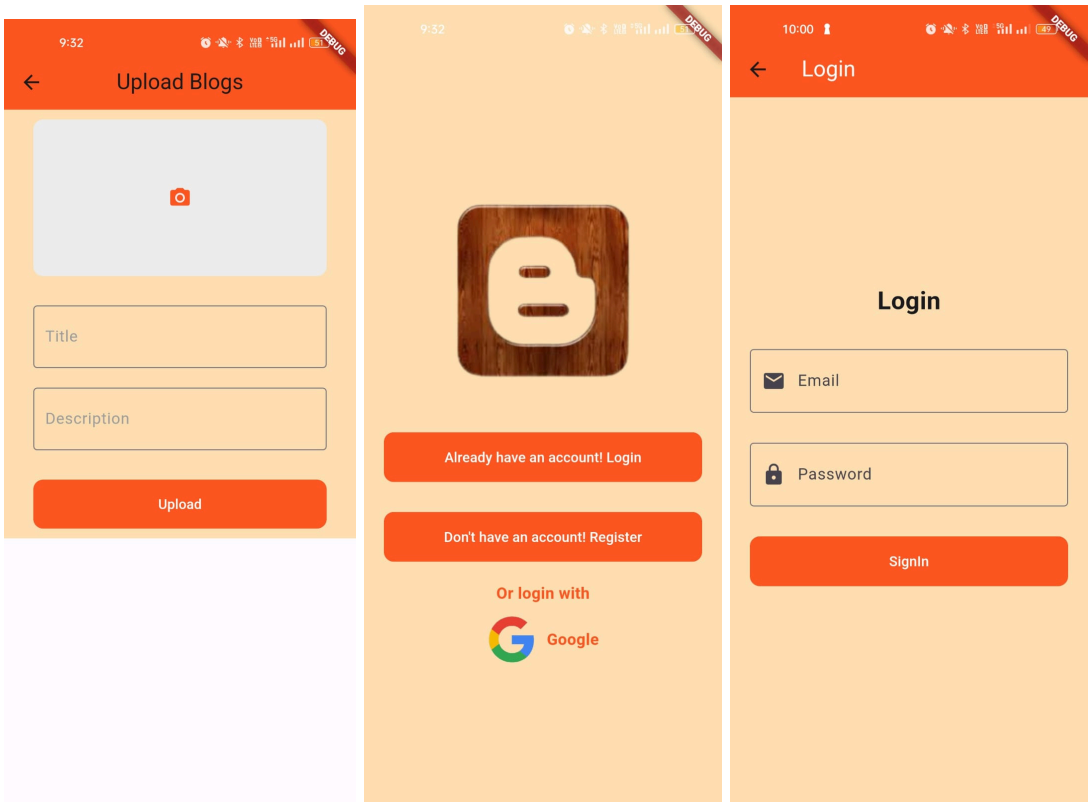
```
      return AlertDialog(
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(10),
        ),
        content: Container(
          height: 120,
          child: Column(
            children: [
              InkWell(
                onTap: () {
                  getImageCamera();
                  Navigator.pop(context);
                },
                child: ListTile(
                  leading: Icon(Icons.camera),
                  title: Text('Camera'),
                ),
              ),
              InkWell(
                onTap: () {
                  getImageGallery();
                  Navigator.pop(context);
                },
                child: ListTile(
                  leading: Icon(Icons.photo_library),
                  title: Text('Gallery'),
                ),
              )
            ],
          ),
        ),
      );
    },
  );
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      backgroundColor: Colors.deepOrange,
      title: Text('Upload Blogs'),
      centerTitle: true,
    ),
    body: SingleChildScrollView(
```

```
child: Container(
  color: Colors.orange[100],
  padding: const EdgeInsets.symmetric(vertical: 10, horizontal: 30),
  child: Column(
    children: [
      InkWell(
        onTap: () {
          dialog(context);
        },
        child: Center(
          child: Container(
            height: MediaQuery.of(context).size.height * .2,
            width: MediaQuery.of(context).size.height * 1,
            child: _image != null
                ? ClipRect(
              child: Image.file(
                _image!.absolute,
                width: 100,
                height: 100,
                fit: BoxFit.fitHeight,
              ),
            )
                : Container(
              decoration: BoxDecoration(
                  color: Colors.grey.shade200,
                  borderRadius: BorderRadius.circular(10)),
              height: 100,
              width: 100,
              child: const Icon(
                Icons.camera_alt,
                color: Colors.deepOrange,
              ),
            ),
          ),
        ),
      ),
      SizedBox(height: 30,),
      Form(
        key: _formkey,
        child: Column(
          children: [
            TextFormField(
              controller: titleController,
              keyboardType: TextInputType.text,
              decoration: const InputDecoration(
```

```
        labelText: 'Title',
        hintText: 'Enter the title of your blog',
        border: OutlineInputBorder(),
        hintStyle: TextStyle(
            color: Colors.grey, fontWeight: FontWeight.normal),
        labelStyle: TextStyle(
            color: Colors.grey, fontWeight: FontWeight.normal),
      ),
      onChanged: (String value) {
        title = value;
      },
      validator: (value) {
        return value!.isEmpty ? 'Enter Title' : null;
      },
    ),
    SizedBox(height: 20,),
    TextFormField(
      controller: descriptionController,
      keyboardType: TextInputType.text,
      minLines: 1,
      maxLines: 10,
      decoration: const InputDecoration(
        labelText: 'Description',
        hintText: 'Enter the description of your blog',
        border: OutlineInputBorder(),
        hintStyle: TextStyle(
            color: Colors.grey, fontWeight: FontWeight.normal),
        labelStyle: TextStyle(
            color: Colors.grey, fontWeight: FontWeight.normal),
      ),
      onChanged: (String value) {
        description = value;
      },
      validator: (value) {
        return value!.isEmpty ? 'Enter Description' : null;
      },
    ),
    SizedBox(height: 30,),
    RoundButton(
      title: 'Upload',
      onPress: () async {
        uploadBlog();
      },
    )
  ],
```

```
                ),
              )
            ],
          ),
        ),
      ),
    );
  }
}
```

OUTPUT:



Conclusion: From the above experiment we came to know about firebase. Then we came to know about how to implement/connect firebase to our application.