**EXPERIMENT NO: - 05**

Name:- Akruti Dabas Class:- D15A Roll:No: - 11

**AIM:**

To apply navigation, routing and gestures in Flutter App.

**THEORY:**

What is Firebase?

Firebase is a comprehensive set of cloud-based tools and services provided by Google, designed to help developers build, deploy, and scale mobile and web applications efficiently. It offers a variety of features, including **Authentication**, **Realtime Database**, **Cloud Messaging**, **Crashlytics**, **Performance Monitoring**, and **Test Lab**[2](https://www.techtarget.com/searchmobilecomputing/definition/Google-Firebase).

Firebase Authentication

Firebase Authentication is a service that provides an end-to-end identity solution for apps, allowing users to sign in using various methods such as email and password, phone numbers, and federated identity providers like Google, Facebook, and Twitter[3](https://firebase.google.com/docs/auth)[4](https://firebase.google.com/products/auth). It integrates well with other Firebase services and supports industry standards like OAuth 2.0 and OpenID Connect. This makes it easy to implement secure authentication systems without extensive backend coding.

**Key Features of Firebase Authentication:**

* **Multi-Platform Support**: Works seamlessly across mobile and web platforms.
* **Easy Integration**: Can be set up with minimal code.
* **Security**: Leverages Google's expertise in managing large account databases.
* **Customizable UI**: FirebaseUI provides a customizable drop-in auth solution.

Firebase Storage

Firebase Storage is a service that allows developers to store and serve user-generated content, such as images and videos, in a scalable and secure manner. It uses Google Cloud Storage buckets, which can be accessed from both Google Cloud and Firebase platforms. This service integrates well with Firebase Authentication, enabling developers to manage access to stored files based on user identity[1](https://www.ashutec.com/blog/what-is-google-firebase-storage-and-how-to-use-it-for-application-development-4f9f462d1487)[6](https://marcus-obst.de/blog/firebase-storage-and-firebase-auth).

**Key Features of Firebase Storage:**

* **Scalability**: Automatically scales to meet storage needs.
* **Security**: Integrates with Firebase Authentication for access control.
* **Ease of Use**: No server-side coding required for basic operations.
* **Complex Operations**: Supports advanced operations via Google Cloud Storage APIs.

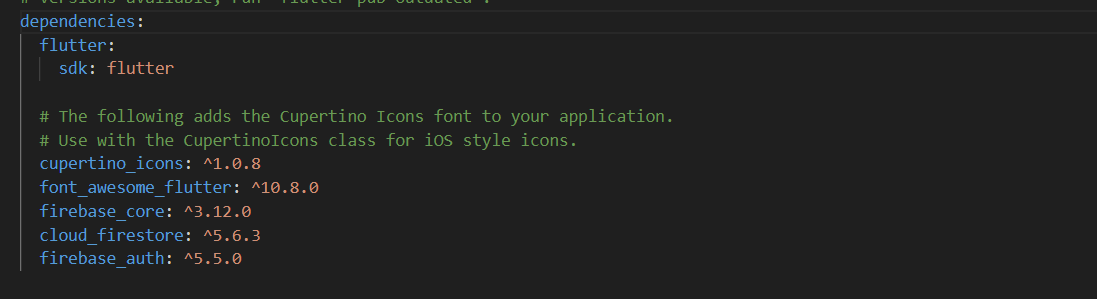
**Using Firebase for Both Authentication and Storage**

1. **Authentication Setup**: Implement Firebase Authentication to manage user identities and access control.
2. **Storage Integration**: Use Firebase Storage to store user-generated content.
3. **Access Control**: Configure security rules to restrict access to stored files based on user authentication status.

This combination allows developers to build applications where users can securely upload, manage, and access their content, all while leveraging Firebase's scalable and secure infrastructur

**CODE:**

In Pub.ysml

****

 Future<void> \_signInWithEmailPassword() async {

    setState(() {

      \_isLoading = true;

      \_errorMessage = null;

    });

    try {

      UserCredential userCredential =

          await FirebaseAuth.instance.signInWithEmailAndPassword(

        email: \_emailController.text.trim(),

        password: \_passwordController.text.trim(),

      );

      User? user = userCredential.user;

      if (user != null) {

        Navigator.pushReplacement(

          context,

          MaterialPageRoute(builder: (context) => const HomeScreen()),

        );

      } else {

        setState(() {

          \_errorMessage = "User not found. Please check your credentials.";

        });

      }

    } catch (e) {

      setState(() {

        if (e is FirebaseAuthException) {

          \_errorMessage = e.message;

        } else {

          \_errorMessage = "An unexpected error occurred: $e";

        }

      });

    } finally {

      setState(() {

        \_isLoading = false;

      });

    }

  }

  Widget \_buildTextButton(String text) {

    return TextButton(

      onPressed: () {

        // Handle button press

      },

      child: Text(

        text,

        style: const TextStyle(color: Colors.green, fontSize: 14),

      ),

    );

  }

}

**SCREENSHOTS**

