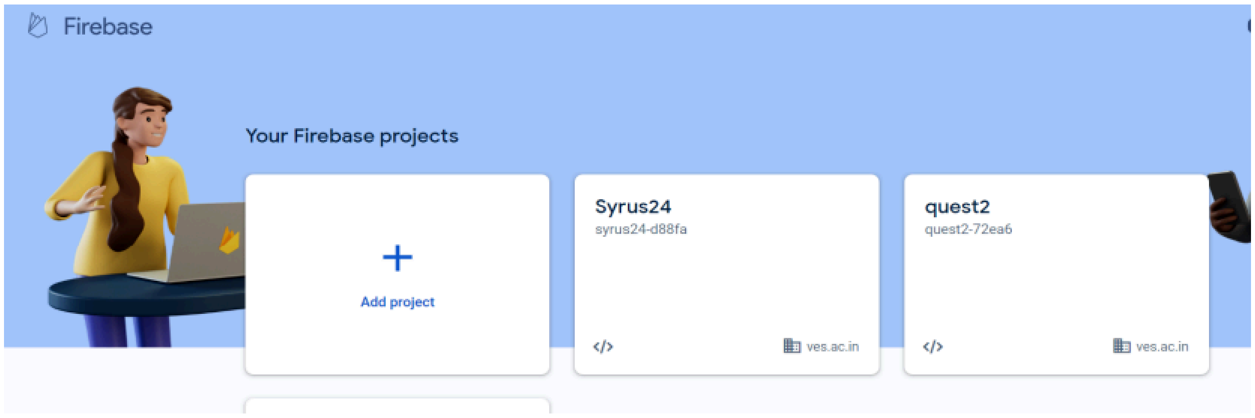
Experiment 6

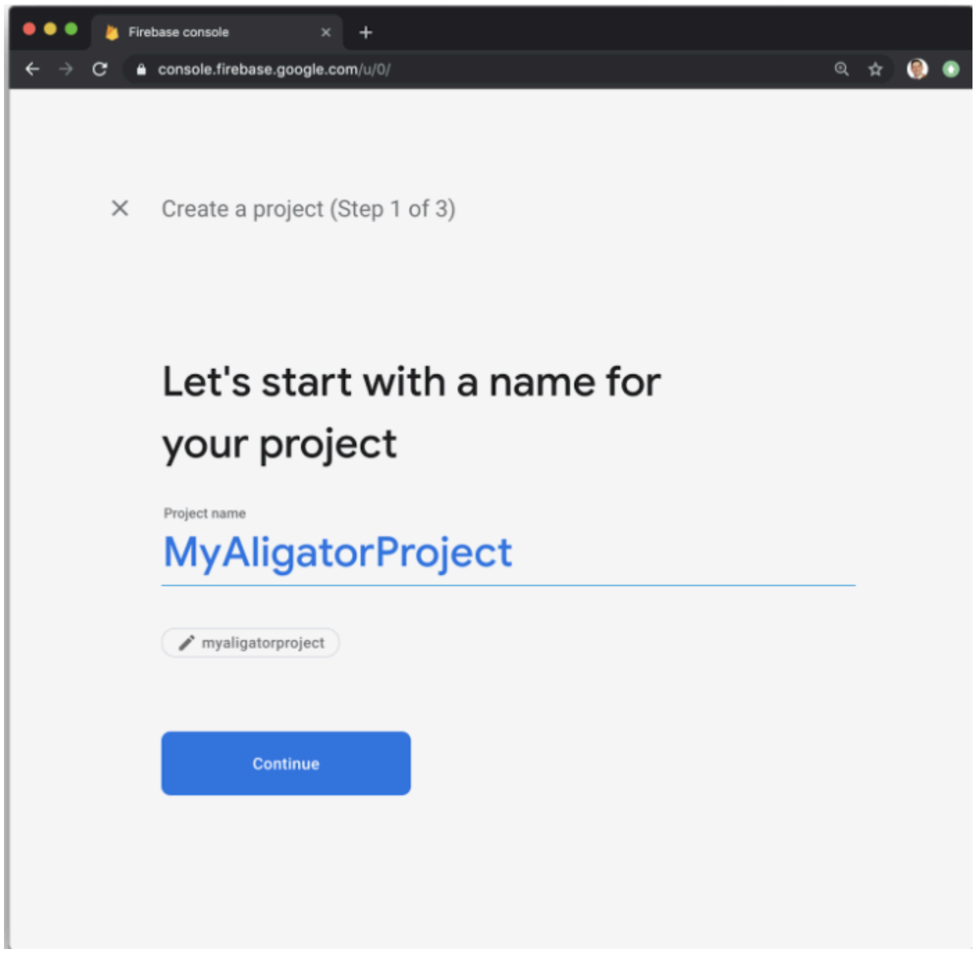
AIM : To connect Flutter app UI with Firebase database

THEORY :
To integrate Flutter with Firebase, begin by setting up a Firebase project, linking your Flutter app, and configuring dependencies in the `pubspec.yaml` file. Initialize Firebase in your app's entry point and consider adding the optional `firebase_auth` package for authentication. For database operations, utilize the `cloud_firestore` package to interact with Firestore, performing CRUD operations and enabling real-time data updates. If your app involves file storage, integrate Firebase Storage using the `firebase_storage` package. Implement robust error-handling mechanisms for asynchronous Firebase operations. Thoroughly test and debug your app, ensuring seamless functionality on both iOS and Android platforms. Once satisfied, deploy your Flutter app, now connected to Firebase, to make the most of the platform's features for authentication, real-time databases, and storage.

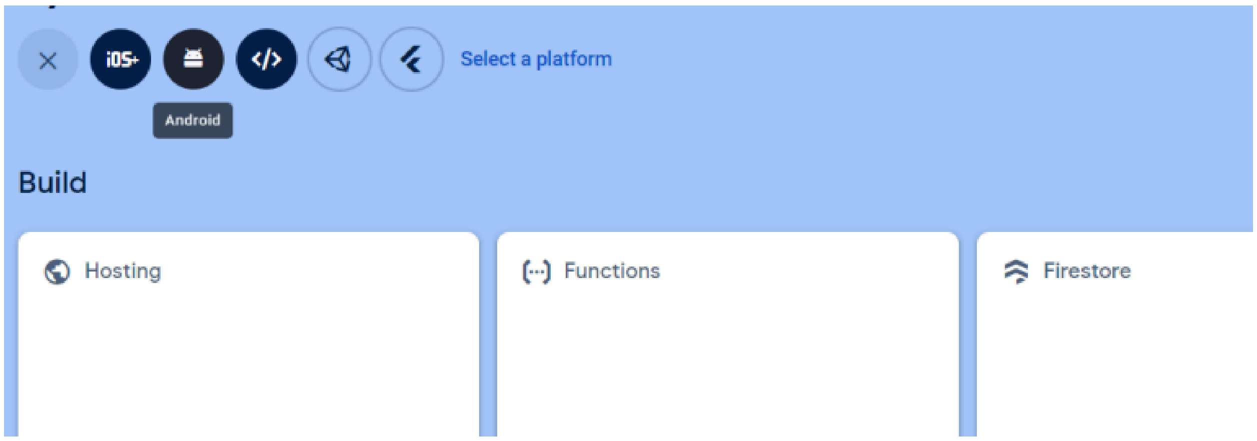
Connecting a Flutter app UI with a Firebase database involves several steps. Firebase is a mobile and web application development platform that provides various services, including a real-time NoSQL database.

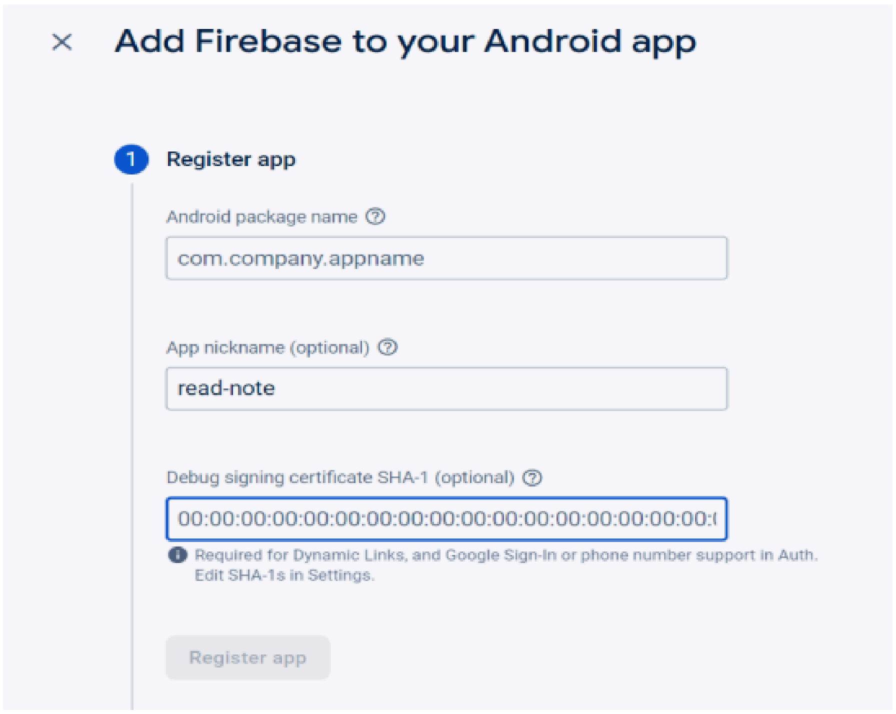
1. Create a Firebase Project:
- Go to the [Firebase Console](https://console.firebase.google.com/).
- Click on "Add Project" and follow the setup instructions.




Firebase console
console.firebase.google.com/u/0/
Create a project (Step 1 of 3)
Let's start with a name for your project
Project name
MyAligatorProject
myaligatorproject
Continue

2. Add a Flutter App to Firebase Project:
- In the Firebase Console, select your project.
- Click on "Add app" and choose the Flutter icon.
- Follow the setup instructions to register your app





Add Firebase to your Android app

1 Register app

Android package name: com.company.appname

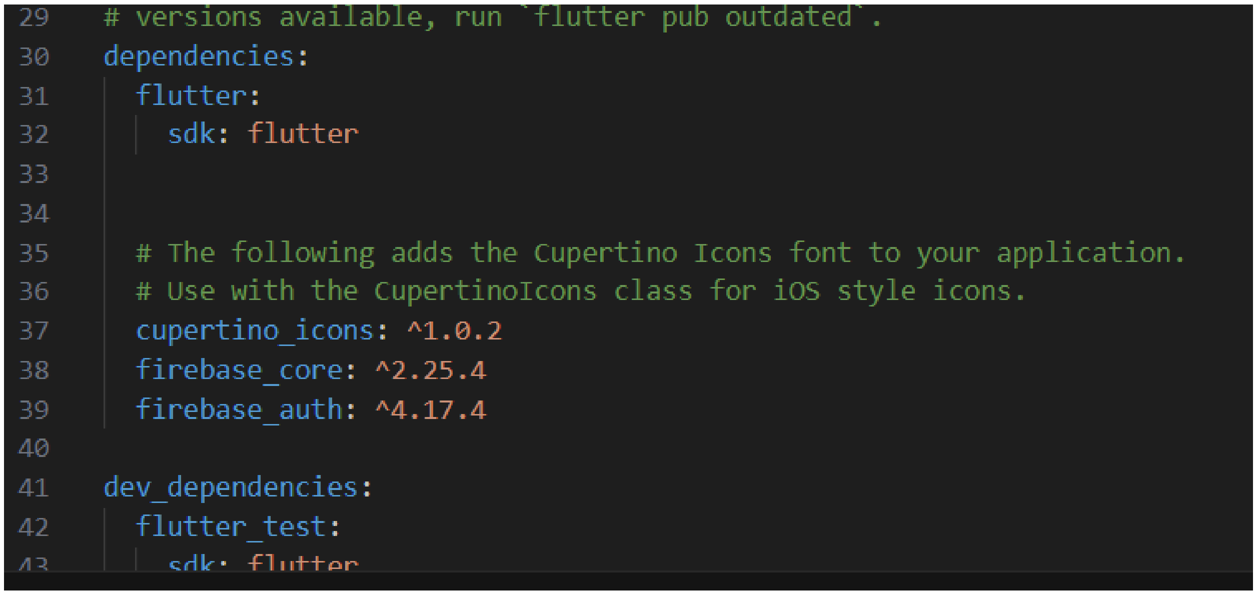
App nickname (optional): read-note

Debug signing certificate SHA-1 (optional): 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:(

Required for Dynamic Links, and Google Sign-In or phone number support in Auth. Edit SHA-1s in Settings.

Yaml

Dependencies:
Flutter:
sdk: flutter
firebase_core: ^latest_version
firebase_database: ^latest_version
- Run flutter pub get to install the dependencies.



```
# versions available, run `flutter pub outdated`.
dependencies:
  flutter:
    sdk: flutter


  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.2
  firebase_core: ^2.25.4
  firebase_auth: ^4.17.4

dev_dependencies:
  flutter_test:
    sdk: flutter
```

4. Initialize Firebase in Flutter:
- Import the Firebase packages in your main.dart file:dart

```
import 'package:firebase_core/firebase_core.dart';
```

- Initialize Firebase in the main function:
dart

```
    void main() async {
        WidgetsFlutterBinding.ensureInitialized();
            await Firebase.initializeApp();
        runApp(MyApp());

    }
```

5. Set Up Firebase Realtime Database:
- In the Firebase Console, go to "Database" and click on "Create Database."
- Choose "Start in test mode" and click "Next."
- Set up your database rules.

6. Create Firebase Database Reference:
- Import the necessary package in your Dart file:dart

```
    import 'package:firebase_database/firebase_database.dart';
```

- Create a reference to your Firebase database:dart

```
final databaseReference = FirebaseDatabase.instance.reference();
```

7. Read Data from Firebase:

- To read data from Firebase, you can use methods like once() or onValue() :

Dart

```
DatabaseReference reference = FirebaseDatabase.instance.reference();
          // Read data once
DataSnapshot snapshot = await reference.once();
          // Access the data
print('Data: ${snapshot.value}');
```

8. Write Data to Firebase:

- To write data to Firebase, you can use methods like set() or push() :

Dart

```
DatabaseReference reference = FirebaseDatabase.instance.reference();
          // Set data
reference.child('users').set({'username': 'JohnDoe', 'email': 'john@example.com'});
```

9. Update or Delete Data:

- Use the update() or remove() methods to update or delete data:

Dart

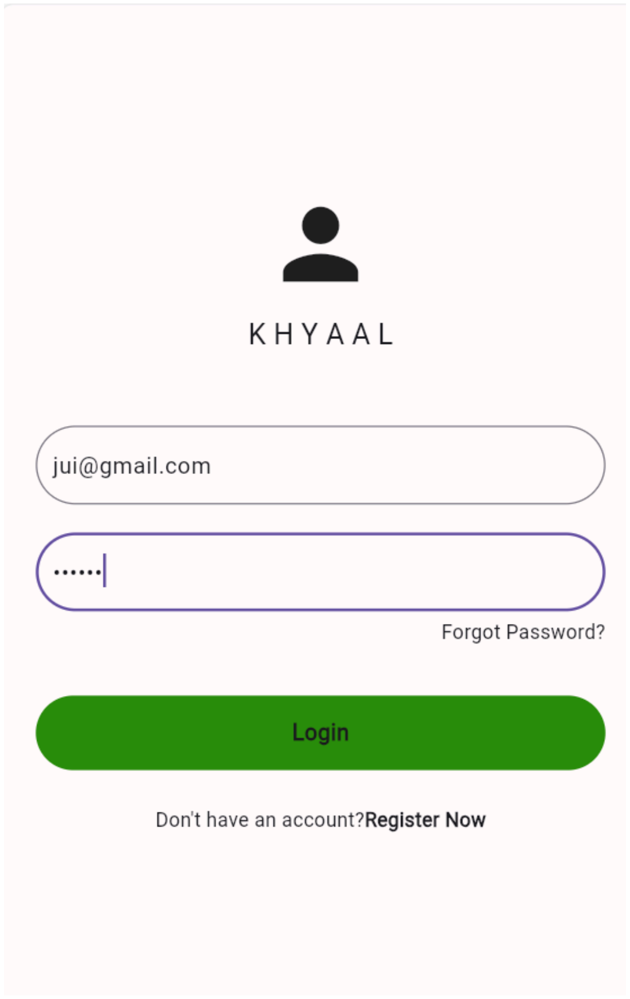
```
atabaseReference reference = FirebaseDatabase.instance.reference();
          // Update data
reference.child('users').child('userId').update({'username': 'NewUsername'});
          // Delete data
          reference.child('users').child('userId').remove();
```

10. Handle Asynchronous Operations:

- Since Firebase operations are asynchronous, make sure to handle them using async/await or .then() .

11. Display Data in Flutter UI:

- Use Flutter widgets to display the data retrieved from Firebase in your app's UI.


KHYAAL

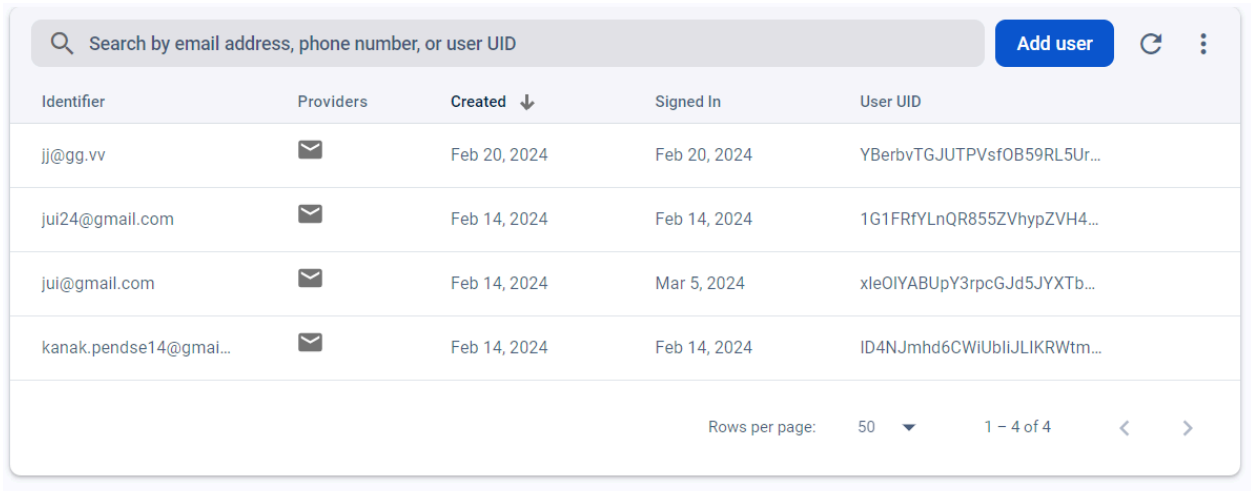
jui@gmail.com

••••••

Forgot Password?

Login

Don't have an account?**Register Now**


Search by email address, phone number, or user UID

Add user

Identifier	Providers	Created	Signed In	User UID
jj@gg.vv	✉	Feb 20, 2024	Feb 20, 2024	YBerbvTGJUTPVsfOB59RL5Ur...
jui24@gmail.com	✉	Feb 14, 2024	Feb 14, 2024	1G1FRfYLnQR855ZVhypZVH4...
jui@gmail.com	✉	Feb 14, 2024	Mar 5, 2024	xleOIYABUpY3rpcGJd5JYXTb...
kanak.pendse14@gmai...	✉	Feb 14, 2024	Feb 14, 2024	ID4NJmhd6CWiUbliJLIKRWtm...

Rows per page: 50 1 – 4 of 4

CONCLUSION : Hence, we have integrated Flutter with Firebase , which offers a robust solution for building powerful cross-platform applications. By combining Flutter's UI capabilities with Firebase's authentication, real-time database, and storage features, developers can create scalable, feature-rich apps. This streamlined process involves project setup, dependency configuration, initialization, and thorough testing, ultimately culminating in a seamless deployment across iOS and Android platforms.