Link --→ [https://github.com/sharlijune/recipe.git](https://github.com/sharlijune/recipe.git)

Login

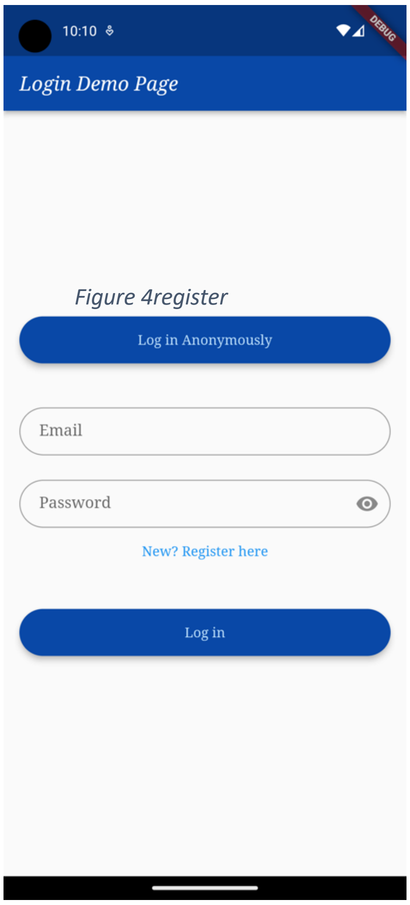
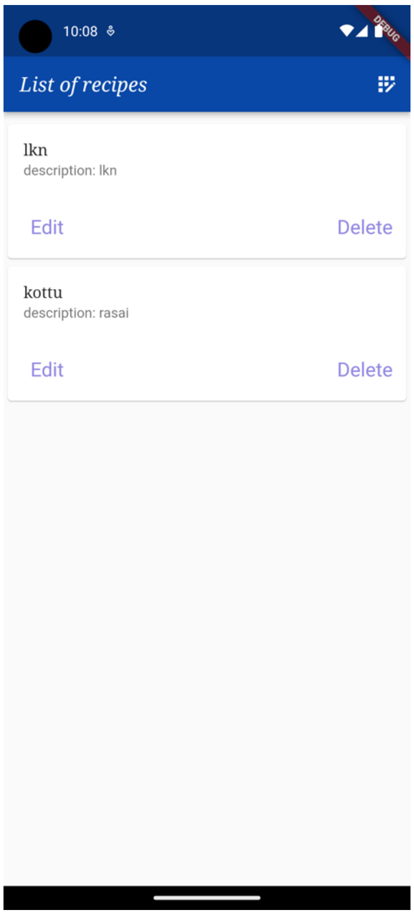


*Figure 4register*

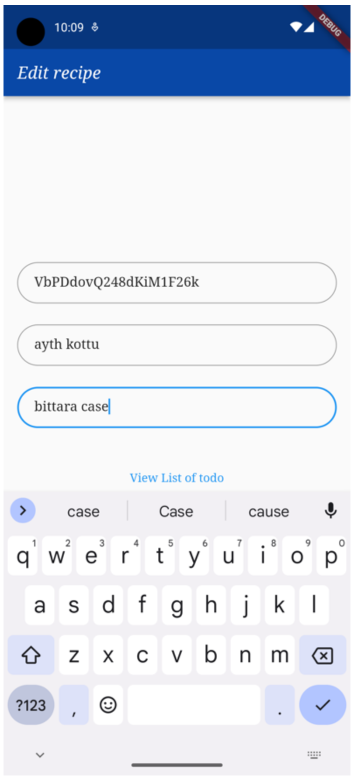
*Figure 2 login*



lkn
description: lkn

kottu
description: rasai

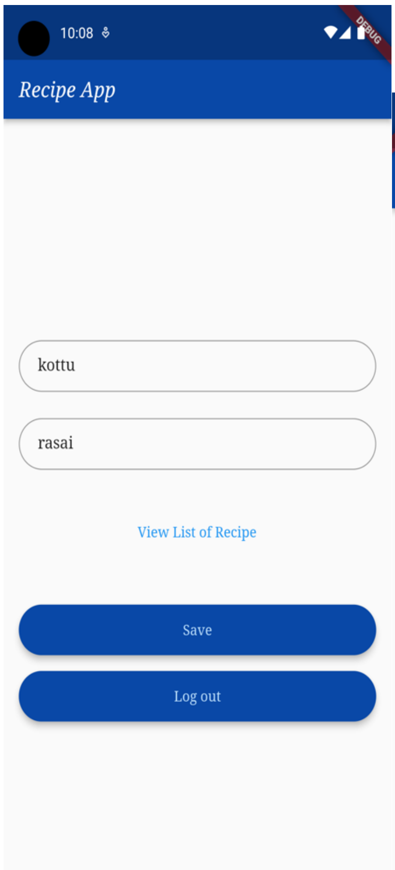
*Figure 1list page*



VbPDdovQ248dKiM1F26k

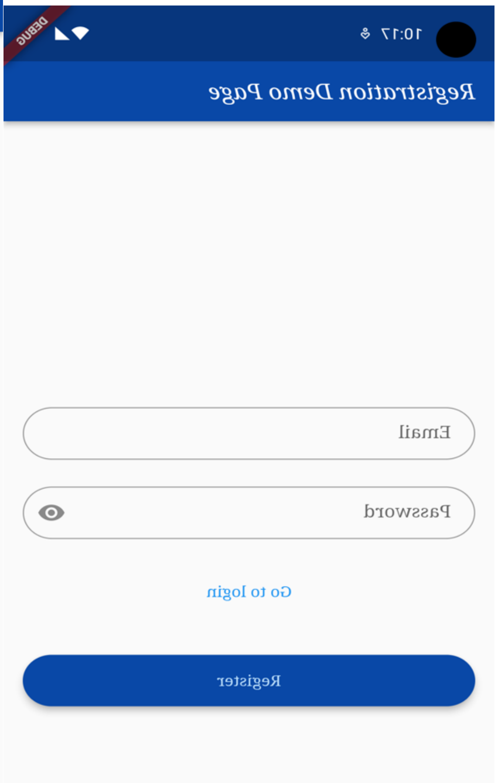
ayth kottu

bittara case



kottu

rasai



Firebase auth

```
import 'package:firebase_auth/firebase_auth.dart';
import '../models/loginuser.dart';
import '../models/FirebaseUser.dart';

class AuthService {
  final FirebaseAuth _auth = FirebaseAuth.instance;


  FirebaseUser? _firebaseUser(User? user) {
    return user != null ? FirebaseUser(uid: user.uid) : null;
  }


  Stream<FirebaseUser?> get user {
    return _auth.authStateChanges().map(_firebaseUser);
  }


  Future signInAnonymous() async {
    try {
      UserCredential userCredential = await _auth.signInAnonymously();
      User? user = userCredential.user;
      return _firebaseUser(user);
    } catch (e) {
     return FirebaseUser(code: e.toString(), uid: null);
    }
  }


  Future signInEmailPassword(LoginUser _login) async {
    try {
      UserCredential userCredential = await FirebaseAuth.instance
          .signInWithEmailAndPassword(
              email: _login.email.toString(),
              password: _login.password.toString());
      User? user = userCredential.user;
      return _firebaseUser(user);
    } on FirebaseAuthException catch (e) {
      return FirebaseUser(code: e.code, uid: null);
```

```dart
    }
  }


  Future registerEmailPassword(LoginUser _login) async {
    try {
      UserCredential userCredential = await FirebaseAuth.instance
          .createUserWithEmailAndPassword(
              email: _login.email.toString(),
              password: _login.password.toString());
      User? user = userCredential.user;
      return _firebaseUser(user);
    } on FirebaseAuthException catch (e) {
      return FirebaseUser(code: e.code, uid: null);
    } catch (e) {
      return FirebaseUser(code: e.toString(), uid: null);
    }
  }


  Future signOut() async {
    try {
      return await _auth.signOut();
    } catch (e) {
      return null;
    }
  }
}
```

Firebase crud

```dart
import 'package:cloud_firestore/cloud_firestore.dart';
import '../models/response.dart';

final FirebaseFirestore _firestore = FirebaseFirestore.instance;
final CollectionReference _Collection = _firestore.collection('Recipes');
class FirebaseCrud {

static Future<Response> addRecipe({
    required String title,
    required String description,

  }) async {


    Response response = Response();
    DocumentReference documentReferencer =
        _Collection.doc();



    Map<String, dynamic> data = <String, dynamic>{
      "titleName": title,
      "description": description,
      "isComplete":false

    };
    var result = await documentReferencer

        .set(data)
        .whenComplete(() {
          response.code = 200;
          response.message = "Sucessfully added to the database";
        })
        .catchError((e) {
            response.code = 500;
            response.message = e;
        });

        return response;
  }
static Stream<QuerySnapshot> readToDo() {


    CollectionReference notesItemCollection =
        _Collection;

    return notesItemCollection.snapshots();
  }





static Future<Response> updateTitle({
    required String title,
    required String description,
     required String docId,

  }) async {
    Response response = Response();
    DocumentReference documentReferencer =
        _Collection.doc(docId);

    Map<String, dynamic> data = <String, dynamic>{
      "titleName": title,
      "description": description,

    };




    await documentReferencer
        .update(data)
        .whenComplete(() {
           response.code = 200;
          response.message = "Sucessfully updated recipe";
        })
        .catchError((e) {
            response.code = 500;
            response.message = e;
        });

        return response;
  }
  static Future<Response> complete({

     required String docId,
     required bool iscomplete,


  }) async {
    Response response = Response();
    DocumentReference documentReferencer =
        _Collection.doc(docId);

    Map<String, dynamic> data = <String, dynamic>{
      "iscomplete": iscomplete,



    };





    await documentReferencer
        .update(data)
        .whenComplete(() {
           response.code = 200;
          response.message = "completed";
        })
        .catchError((e) {
            response.code = 500;
            response.message = e;
        });

        return response;
  }






  static Future<Response> deletetodo({
    required String docId,
  }) async {
     Response response = Response();
    DocumentReference documentReferencer =
        _Collection.doc(docId);
    await documentReferencer
        .delete()
        .whenComplete((){
          response.code = 200;
          response.message = "Sucessfully Deleted recipe";
        })
        .catchError((e) {
           response.code = 500;
            response.message = e;
        });
   return response;
  }

}
```