```
# tracker

A new Flutter project.

## Getting Started

This project is a starting point for a Flutter application.

A few resources to get you started if this is your first Flutter project:

- [Lab: Write your first Flutter app](https://docs.flutter.dev/get-
started/codelab)
- [Cookbook: Useful Flutter samples](https://docs.flutter.dev/cookbook)

For help getting started with Flutter development, view the
[online documentation](https://docs.flutter.dev/), which offers tutorials,
samples, guidance on mobile development, and a full API reference.
```

```
import 'package:bloc/bloc.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:tracker/shared/bloc_observer.dart';
import 'package:tracker/shared/cache_helper.dart';
import 'package:tracker/screens/splash_screen.dart';


import 'shared/consteant.dart';
import 'cubit/cubit.dart';
import 'screens/home_screen.dart';
import 'screens/layout_screen.dart';
import 'screens/login_screen.dart';

main()async{
  Bloc.observer = const SimpleBlocObserver();
  WidgetsFlutterBinding.ensureInitialized();
  try {
    await Firebase.initializeApp();
  } catch (e) {
    print('Firebase initialization error: $e');
  }


  await CacheHelper.init();


  Widget widget;
  uid=CacheHelper.getData(key:'userId');

  if(uid != null){
    widget = LayoutScreen();
  }else{
    widget = SplashScreen();
  }



  runApp(
      MultiBlocProvider(
        providers: [
          BlocProvider(
              create:(BuildContext context)=>SokarCubit()..getUserData()
          ),
        ],
        child: MyApp(
          startWidget :widget,
        ),
      ));
  //runApp(const MyApp());
}

class MyApp extends StatefulWidget {
  MyApp({Key? key,required this.startWidget}) : super(key: key);
  final Widget startWidget;

  @override
  State<MyApp> createState() => _MyAppState();
}

class _MyAppState extends State<MyApp> {
  // This widget is the root of your application.
  @override
  void initState() {
    FirebaseAuth.instance
        .authStateChanges()
        .listen((User? user) {
      if (user == null) {
        print('User is currently signed out!');
      } else {
        print('User is signed in!');
      }
    });
    // TODO: implement initState
    super.initState();
  }
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      home:  widget.startWidget,
    );
  }
}
```

```yaml
name: tracker
description: A new Flutter project.
# The following line prevents the package from being accidentally published
to
# pub.dev using `flutter pub publish`. This is preferred for private
packages.
publish_to: 'none' # Remove this line if you wish to publish to pub.dev

# The following defines the version and build number for your application.
# A version number is three numbers separated by dots, like 1.2.43
# followed by an optional build number separated by a +.
# Both the version and the builder number may be overridden in flutter
# build by specifying --build-name and --build-number, respectively.
# In Android, build-name is used as versionName while build-number used as
versionCode.
# Read more about Android versioning at
https://developer.android.com/studio/publish/versioning
# In iOS, build-name is used as CFBundleShortVersionString while build-number
is used as CFBundleVersion.
# Read more about iOS versioning at
#
https://developer.apple.com/library/archive/documentation/General/Reference/I
nfoPlistKeyReference/Articles/CoreFoundationKeys.html
# In Windows, build-name is used as the major, minor, and patch parts
# of the product and file versions while build-number is used as the build
suffix.
version: 1.0.0+1

environment:
  sdk: '>=3.0.3 <4.0.0'

# Dependencies specify other packages that your package needs in order to
work.
# To automatically upgrade your package dependencies to the latest versions
# consider running `flutter pub upgrade --major-versions`. Alternatively,
# dependencies can be manually updated by changing the version numbers below
to
# the latest version available on pub.dev. To see which dependencies have
newer
# 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
  shared_preferences: ^2.0.15
  connectivity_plus: ^3.0.2
  http: ^0.13.5
  fluttertoast: ^8.0.0
  geolocator: ^8.2.1
  geocoding: ^2.0.4
  intl: ^0.17.0


  flutter_polyline_points: ^1.0.0


  #new


  bloc: ^8.1.2
  flutter_bloc: ^8.1.2
  animated_conditional_builder: ^0.0.5
  image_picker: ^1.0.4

  firebase_core: ^2.17.0
  firebase_auth: ^4.10.1
  modal_progress_hud_nsn: ^0.3.0
  cloud_firestore: ^4.9.3

  firebase_storage: ^11.2.8
  percent_indicator: ^4.2.3
  email_validator: ^2.1.17




  awesome_bottom_bar: ^1.2.4


#map

  permission_handler: ^11.0.1
  location: ^6.0.1
  google_maps_flutter: ^2.6.1

  flutter_map: ^4.0.0
  latlong2: ^0.8.0


  dio: ^5.4.3+1
  firebase_database: ^10.5.4







dev_dependencies:
  flutter_test:
    sdk: flutter

  # The "flutter_lints" package below contains a set of recommended lints to
  # encourage good coding practices. The lint set provided by the package is
  # activated in the `analysis_options.yaml` file located at the root of your
  # package. See that file for information about deactivating specific lint
  # rules and activating additional ones.
```

```yaml
  flutter_lints: ^2.0.0

# For information on the generic Dart part of this file, see the
# following page: https://dart.dev/tools/pub/pubspec

# The following section is specific to Flutter packages.
flutter:

  # The following line ensures that the Material Icons font is
  # included with your application, so that you can use the icons in
  # the material Icons class.
  uses-material-design: true

  # To add assets to your application, add an assets section, like this:
  assets:
     - assets/
     - assets/onboarding/

  #   - images/a_dot_ham.jpeg

  # An image asset can refer to one or more resolution-specific "variants",
see
  # https://flutter.dev/assets-and-images/#resolution-aware

  # For details regarding adding assets from package dependencies, see
  # https://flutter.dev/assets-and-images/#from-packages

  # To add custom fonts to your application, add a fonts section here,
  # in this "flutter" section. Each entry in this list should have a
  # "family" key with the font family name, and a "fonts" key with a
  # list giving the asset and other descriptors for the font. For
  # example:
  # fonts:
  #   - family: Schyler
  #     fonts:
  #       - asset: fonts/Schyler-Regular.ttf
  #       - asset: fonts/Schyler-Italic.ttf
  #         style: italic
  #   - family: Trajan Pro
  #     fonts:
  #       - asset: fonts/TrajanPro.ttf
  #       - asset: fonts/TrajanPro_Bold.ttf
  #         weight: 700
  #
  # For details regarding fonts from package dependencies,
  # see https://flutter.dev/custom-fonts/#from-packages
```

```dart




import 'package:bloc/bloc.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';

import '../model/user_model.dart';
import '../screens/login_screen.dart';
import '../screens/signup_screen.dart';
import '../shared/cache_helper.dart';
import '../shared/consteant.dart';
import 'states.dart';

class SokarCubit extends Cubit<SokarState>{

  SokarCubit() : super(SokarInitialState());

  static SokarCubit get(context) => BlocProvider.of(context);

  //indecator progress
  bool showSpinner = false;

  void playSpinner(){
    showSpinner=true;

  }
  void stopSpinner(){
    showSpinner=false;
  }

  bool obsecuer = true;

  void openObsecuer(){
    obsecuer = false;
    emit(OpenObsecuerState());
  }

  void closeObsecuer(){
    obsecuer = true;
    emit(OpenObsecuerState());
  }




  //register user
  Future<void> userRegister({
    required String name,
    required String email,
    required String password,
    BuildContext? context

  }) async {
    emit(RegisterLoadingState());
    try {
      var value = await FirebaseAuth.instance.createUserWithEmailAndPassword(
        email: email,
        password: password,
      );

      uid =value.user!.uid; //gameddddddddd ya abdoooooo wallahyyyyyyyyyy
      await userCreate(
        uid: value.user!.uid,
        email: email,
        password: password,
        name: name,
      );
      emit(RegisterSuccessState());
    } on FirebaseAuthException catch (e) {
      if(e.code == 'weak-password'){
        print('the password is too weak');
        _showErrorDialog('the password is too weak',context!,SignupScreen());
      }
      else if(e.code == 'email-already-in-use'){
        print('the account already exists for that email');
        _showErrorDialog('the account already exists for that
email',context!,SignupScreen());
      }
      print("Error during user registration: $e");
      emit(RegisterErrorState());
    }
  }

  //use it in userRegister above
  Future<void> userCreate({
    required String email,
    required String name,
    required String password,
    required String uid,
    String? kidName,
    String? gender,
  }) async {
    UserModel model = UserModel(
      email: email,
      password: password,
      userId: uid,
      name: name,
      kidName: kidName,
      gender: gender
    );
    try {
      await FirebaseFirestore.instance
          .collection('users')
          .doc(uid)
          .set(model.toMap());
      emit(CreateUserSuccessState());
    } catch (error) {
      print("Error during user creation: $error");
      emit(CreateUserErrorState());
    }
  }
  // Future<void> userLogin1({
  //   required String email,
  //   required String password,
  // }) async {
  //   emit(LoginLoadingState());
  //   await
FirebaseFirestore.instance.collection('users').doc(uid).snapshots().forEach((
element) {
  //     if(element.data()?['emil'] ==email &&
element.data()?['password']==password){
  //       print('i user');
  //
  //     }else{
  //
  //     }
  //   });
  // }
  Future<void> userLogin({
    required String email,
    required String password,
    BuildContext? context
  }) async {
    emit(LoginLoadingState());
    try {
      var value = await FirebaseAuth.instance.signInWithEmailAndPassword(
        email: email,
        password: password,
      );
      emit(LoginSuccessState(value.user!.uid));
    } on FirebaseAuthException catch (e) {
      if(e.code == 'user-not-found'){
        print('no user found for that email ');
        _showErrorDialog('no user found for that
email',context!,LoginScreen());

      }
      else if(e.code == 'wrong-password'){
        print('wrong password provided for that user ');
        _showErrorDialog('wrong password provided for that
user',context!,LoginScreen());

      }

      print("Error during user login: $e");
      emit(LoginErrorState());
    }
  }

  UserModel? model;

  Future<void> getUserData() async {
    emit(GetUserLoadingState());
    FirebaseFirestore.instance
        .collection('users')
        .doc(uid)
        .get()
        .then((value) {
      print(value.data()); // is map
      print(uid);
      print('dddddddddddddddd');
      print('${value.id}');
      uid=value.id;
      model = UserModel.fromJson(value.data()!);
      emit(GetUserSuccessState());
    }).catchError((error) {
      print(error.toString());
      emit(GetUserErrorState());
    });
  }



  void signOut(context) {
    CacheHelper.removeDate(key: 'userId')
        .then((value) {
      if (value) {
        Navigator.pushReplacement(
            context, MaterialPageRoute(builder: (context) => LoginScreen()));
      }
    });
  }

  void _showErrorDialog(String errorMessage,BuildContext context,Widget
screen) {
    showDialog(
      context: context,
      builder: (BuildContext context) {
        return AlertDialog(
          title: Text('Error'),
          content: Text(errorMessage),
          actions: <Widget>[
            GestureDetector(
              child: Text('OK'),
              onTap: () {
                Navigator.of(context).push(MaterialPageRoute(builder:
(context)=>screen));
              },
            ),
          ],
        );
      },
    );
  }


}
```

```
import 'package:awesome_bottom_bar/awesome_bottom_bar.dart';
import 'package:flutter/material.dart';
```

```dart
import 'package:awesome_bottom_bar/tab_item.dart';


import 'add_device_screen.dart';
import 'get_data.dart';
import 'home_screen.dart';

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

  @override
  State<LayoutScreen> createState() => _LayoutScreenState();
}

class _LayoutScreenState extends State<LayoutScreen> {
  List<TabItem> tabItems = [
    TabItem(icon: Icons.home, title: 'Home'),
    TabItem(icon: Icons.spatial_tracking_outlined, title: 'location'),
    TabItem(icon: Icons.device_unknown_outlined, title: 'add device'),
  ];

  List<Widget>screens=[
    HomeScreen(),
    MapScreen(),
    // FaceBlurPage(),
    AddDeviceScreen()
  ];

  int currentIndex = 0;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body:screens[currentIndex], // Your main content goes here
      bottomNavigationBar: Padding(
        padding: const EdgeInsets.fromLTRB(10,10,10,10),
        child: Stack(
          children: [
            Padding(
              padding: const EdgeInsets.fromLTRB(0,0,0,0),
              child: BottomBarFloating(
                items: tabItems,
                backgroundColor: Colors.white, // Customize your background
color
                boxShadow: [
                  BoxShadow(color: Colors.black.withOpacity(0.2), blurRadius:
5, offset: Offset(0, -3))
                ], // Add a custom box shadow
                borderRadius: BorderRadius.circular(20), // Make it circular
                color: Colors.grey, // Customize default icon color
                colorSelected: Colors.blue, // Customize selected icon color
                indexSelected: currentIndex, // Specify the initially
selected index
                onTap: (index) {
                  // Handle tap event
                  setState(() {
                    currentIndex = index;
```

```
                   screens[index];
                  });
                  print('Tab $currentIndex tapped');
                },
                iconSize: 30, // Change icon size
                titleStyle: TextStyle(fontSize: 14, fontWeight:
FontWeight.bold), // Change title style
                paddingVertical: 12, // Adjust vertical padding
                top: 16, // Adjust top spacing
                bottom: 16, // Adjust bottom spacing
                pad: 0, // Adjust spacing between icon and title
                duration: Duration(milliseconds: 600), // Change animation
duration
                curve: Curves.easeInOut, // Change animation curve
              ),
            ),
            if (currentIndex != null && currentIndex >= 0 && currentIndex <
tabItems.length)
              Positioned(
                top: 0,
                left: MediaQuery.of(context).size.width / tabItems.length *
currentIndex,
                child: Padding(
                  padding: const EdgeInsets.fromLTRB(25, 0, 0, 0),
                  child: Center(
                    child: Container(
                      decoration: BoxDecoration(
                        borderRadius: BorderRadius.circular(2.5),
                        color: Colors.blue, // Color of the line
                      ),
                      height: 5,
                      width: MediaQuery.of(context).size.width /
tabItems.length/2,
                    ),
                  ),
                ),
              ),
          ],
        ),
      ),
    );
  }
}
```

```dart
import 'package:flutter/material.dart';
class MapScreen extends StatelessWidget {
  const MapScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Map'),
      ),
    );
  }
}
```