NAME : HASEEB UR REHMAN

SAP ID : 55859

SECTION : SE 5-2

ASSIGNMENT : 04

GIT LINK :

https://github.com/Haseeb55859/Mobile-Application/tree/main/Assignment_04/assignment_04

SCREEN SHOTS :

main.dart > MyApp > MyApp

```dart
// lib/main.dart
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'models/activity.dart';
import 'services/api_service.dart';
import 'services/location_service.dart';
import 'repositories/activity_repository.dart';
import 'providers/activity_provider.dart';
import 'utils/hive_helper.dart';
import 'screens/home_screen.dart';

Run | Debug | Profile
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await HiveHelper.init();
  final api = ApiService();
  final repo = ActivityRepository(api: api);
  runApp(MyApp(repo: repo));
}

class MyApp extends StatelessWidget {
  final ActivityRepository repo;
  const MyApp({Key? key, required this.repo}) : super(key: key);
```

```dart
// lib/utils/hive_helper.dart
import 'package:hive/hive.dart';
import 'package:hive_flutter/hive_flutter.dart';

class HiveHelper {
  static const String kBoxName = 'smarttrackerBox';

  static Future<void> init() async {
    await Hive.initFlutter();
    await Hive.openBox(kBoxName);
  }
}
```

```dart
// lib/widgets/map_widget.dart
import 'package:flutter/material.dart';
import 'package:google_maps_flutter/google_maps_flutter.dart';

class SimpleMap extends StatelessWidget {
  final double latitude;
  final double longitude;
  final double zoom;

  const SimpleMap({Key? key, required this.latitude, required this.longitude, this.z
      : super(key: key);

  @override
  Widget build(BuildContext context) {
    final initial = CameraPosition(target: LatLng(latitude, longitude), zoom: zoom);
    return GoogleMap(
      initialCameraPosition: initial,
      markers: {
        Marker(markerId: MarkerId('me'), position: LatLng(latitude, longitude)),
      },
      myLocationEnabled: true,
      zoomControlsEnabled: false,
    ); // GoogleMap
  }
}
```

```dart
import 'dart:convert';
import 'package:http/http.dart' as http;
import '../models/activity.dart';

class ApiService {
  // For Android emulator use 10.0.2.2, for real device replace with server host
  static const String baseUrl = 'http://10.0.2.2:3000';

  Future<List<Activity>> fetchActivities() async {
    final resp = await http.get(Uri.parse('$baseUrl/activities'));
    if (resp.statusCode == 200) {
      final List<dynamic> list = jsonDecode(resp.body);
      return list.map((e) => Activity.fromJson(e)).toList();
    }
    throw Exception('Failed to fetch activities');
  }

  Future<Activity> createActivity(Activity a) async {
    final resp = await http.post(
      Uri.parse('$baseUrl/activities'),
      headers: {'Content-Type': 'application/json'},
      body: jsonEncode(a.toJson()),
```

```dart
import 'dart:convert';
import 'dart:io';
import 'package:camera/camera.dart';
import 'package:path_provider/path_provider.dart';

class CameraService {
  CameraController? controller;
  List<CameraDescription>? cameras;

  Future<void> init() async {
    cameras = await availableCameras();
    if (cameras != null && cameras!.isNotEmpty) {
      controller = CameraController(cameras!.first, ResolutionPreset.medium);
      await controller!.initialize();
    } else {
      throw Exception('No cameras available');
    }
  }

  Future<String> takePictureAsBase64() async {
    if (controller == null || !controller!.value.isInitialized) {
      throw Exception('Camera not initialized');
```

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

class LocationService {
  Future<Position> getCurrentLocation() async {
    bool serviceEnabled = await Geolocator.isLocationServiceEnabled();
    if (!serviceEnabled) {
      throw Exception('Location services are disabled.');
    }

    LocationPermission permission = await Geolocator.checkPermission();
    if (permission == LocationPermission.denied) {
      permission = await Geolocator.requestPermission();
      if (permission == LocationPermission.denied) {
        throw Exception('Location permissions are denied');
      }
    }
    if (permission == LocationPermission.deniedForever) {
      throw Exception('Location permissions are permanently denied.');
    }

    return await Geolocator.getCurrentPosition(desiredAccuracy: LocationAccuracy.high
  }
}
```