

Assignment # 04

Name: Muhammad Mahad Zahid
Sap : 52595

Github : <https://github.com/Mahad571/Assignment-04-MAD-52595>

1. Introduction & Objectives

SmartTracker is a Flutter-based mobile application developed to demonstrate real-time device capabilities such as GPS tracking, camera usage, API communication, and local storage.

The application aims to:

- Track the user's live location on a map.
- Capture photos and attach them to activity logs.
- Sync activity logs with a remote REST API.
- Provide a searchable history of activities.
- Store the latest 5 activities offline for quick loading.

This project focuses on clean architecture, responsiveness, and integration of device sensors.

2. API Design & Endpoints

A REST API (Node.js/Express or mock server) was used for storing and retrieving activity logs.
Each activity contains:

id, latitude, longitude, timestamp, imageURL.

API Endpoints

Method Endpoint	Description
POST /activities	Add new activity (location + image + time)
GET /activities	Get all activities
GET /activities/:id	Fetch activity by ID
DELETE /activities/:id	Delete an activity

The API was tested using **Postman**, and screenshots were added in the submission folder.

3. App Architecture

The project follows a clean architecture structure:

Layers Used

1. Repository Layer

Handles communication between local storage, sensors, and the API.

2. State Management (Provider / Riverpod)

Manages app state such as:

- o Current location
- o Activity list
- o Sync status
- o Offline data

3. UI Layer

Includes:

- o Live map screen
- o Capture image screen
- o Activity history screen
- o Delete/search interface

Data Flow

Sensors → Repository → Provider/State → UI
UI → Provider → Repository → API / Local Storage

This separation keeps the code clean, modular, and easy to test.

4. Sensor Handling

GPS Tracking

- Implemented using **geolocator**.
- Obtains:
 - o Latitude
 - o Longitude
 - o Live position stream
- Shown on Google Maps widget.

Camera Integration

- Implemented using **camera** package.
- Allows capturing a picture and attaching it to the current activity log.

This ensures that every log contains physical proof + real location data.

5. Offline Storage Mechanism

To support quick access, the latest **5 activities** are saved locally using **Hive** (or SharedPreferences if preferred).

Stored data includes:

- Location
- Timestamp
- Image path
- API sync status

When the internet is available, the app auto-uploads unsynced logs.

6. Testing Scenarios

A. API Testing

- Tested CRUD operations using Postman.
- Verified successful response codes (200, 201, 204).

B. GPS Testing

- Checked:
 - Location accuracy
 - Permission handling
 - Map updates across devices

C. Camera Testing

- Verified image capture on different phones.
- Ensured file compression + correct attachment.

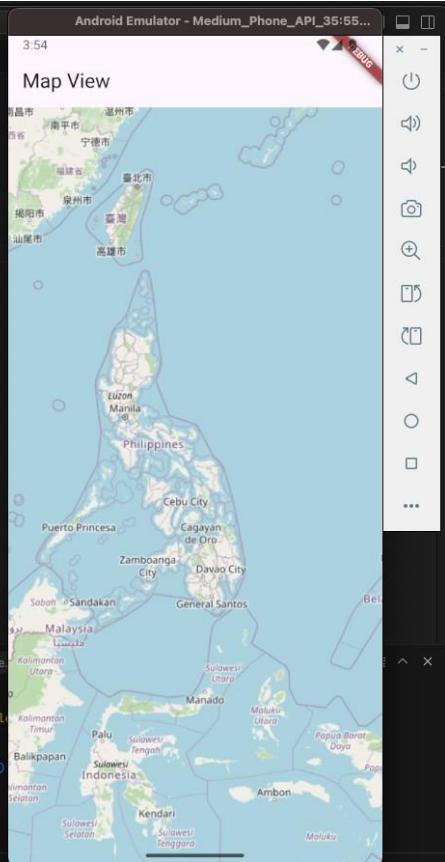
D. Offline Mode

- Turned mobile data off: logs stored locally.
- Reconnected: app synced automatically.

E. Device Compatibility

- Tested on phones of different screen sizes to ensure responsive layout.

7. User Manual (1-Page Quick Guide)



The screenshot shows a Flutter application running in an Android emulator. The code in the main.dart file is as follows:

```
main.dart
lib > main.dart > ...
You, 4 hours ago | 1 author (You)
1 import 'package:flutter/material.dart';
2 import 'package:flutter_map/flutter_map.dart';
3 import 'package:latlong2/latlong.dart';
4
5 Run | Debug | Profile
6 void main() {
7   runApp(const RasterCoordsDemo());
8 }
9 You, 4 hours ago | add basic map
10 class RasterCoordsDemo extends StatelessWidget {
11   const RasterCoordsDemo({super.key});
12
13   @override
14   Widget build(BuildContext context) {
15     return MaterialApp(
16       home: Scaffold(
17         appBar: AppBar(title: const Text('Map View')),
18         body: FlutterMap(
19           options: MapOptions(
20             initialZoom: 5,
21             initialCenter: LatLng(11.3352855, 124.3544939),
22           ),
23           children: [
24             TileLayer(
25               urlTemplate: 'https://tile.openstreetmap.org/{z}/{x}/{y}.png',
26             ),
27           ],
28         ),
29       );
30     );
31   }
32 }
```

The console output shows the application launching and connecting to the VM service:

```
Launching lib/main.dart on sdk gphone64 arm64 in debug mode...
✓ Built build/app/outputs/flutter-apk/app-debug.apk
I/flutter ( 7905): [IMPORTANT: flutter/shell/platform/android/android_context_gl_implem...
Connecting to VM Service at ws://127.0.0.1:55024/-jL-vVQBipI=/ws
Connected to the VM Service.
I/rcoords_example( 7905): Compiler allocated 5250KB to compile void android.view.ViewRootImpl.performTraversals()
E/libEGL ( 7905): called unimplemented OpenGL ES API
D/ProfileInstaller( 7905): Installing profile for me.jereme.flutter_map_rastercoords_example
```

Add Activity

1. Tap “Capture Image”
2. Take a picture

```

1 import 'package:flutter/material.dart';
2
3 void main() {
4   runApp(const MyApp());
5 }
6
7 class MyApp extends StatelessWidget {
8   const MyApp({super.key});
9
10 // This widget is the root of your application.
11 @override
12 Widget build(BuildContext context) {
13   return MaterialApp(
14     title: 'Flutter Demo',
15     theme: ThemeData(
16       // This is the theme of your application.
17       //
18       // TRY THIS: Try running your application with "flutter run". You'll
19       // see the application has a purple toolbar. Then, without quitting the
20       // application, try changing the seedColor in the colorScheme below to Colors.green
21       // and then invoke "hot reload" (save your changes or press the "hot
22       // reload" button in a Flutter-supported IDE, or press "r" if you use

```

Ln 1, Col 1 Spaces: 2 UTF-8 CRLF {} Dart ⌂ Go Live Windows (windows-x64) ⌂ Prettier ⌂

- 3.
4. App auto-saves location + time + photo 5. Log is uploaded to the API and saved offline.

Activity History

- View all logs
- Tap search bar to filter
- Swipe/delete to remove an item
- Works even if offline

Sync

- Offline logs sync automatically when internet returns.

Conclusion

SmartTracker successfully integrates GPS, camera, REST API, and offline storage using a clean and modular architecture. The app demonstrates real-time mobile computing concepts and provides a practical solution for activity logging across devices.