Đã suy nghĩ trong 12s

Tuyệt — mình sẽ giúp bạn triển khai **In Old Days** (ứng dụng di động Flutter + SQLite + MapLibre) theo hướng toàn diện: kiến trúc, schema DB, các màn chính, flow đăng nhập/đăng ký, lưu sự kiện yêu thích, kiểm tra/generate mật khẩu, và ví dụ code để bạn paste vào VS Code chạy thử. Mình cũng cho thêm lưu ý quan trọng khi tích hợp MapLibre trong Flutter (tham khảo chính thức). [Dart packages+1](https://pub.dev/packages/maplibre_gl?utm_source=chatgpt.com" \t "_blank)

Mình trình bày theo các phần: overview → data model/DB → cấu trúc project → màn & flow → code mẫu quan trọng → tích hợp MapLibre → checklist triển khai.

# 1) Tổng quan / yêu cầu kỹ thuật

Mục tiêu: hiển thị danh sách sự kiện lịch sử Việt Nam trong 1 năm, dạng danh sách và trên bản đồ; chi tiết sự kiện; đăng nhập/đăng ký; quản lý tài khoản; thay đổi/forget password; lưu sự kiện yêu thích.

Công nghệ:

Flutter (single codebase), viết bằng VS Code.

Local DB: SQLite (gói sqflite + path\_provider).

Map: MapLibre (Flutter plugin maplibre\_gl hoặc maplibre tùy cách bạn tiếp cận; có plugin chính thức maplibre\_gl). [Dart packages+1](https://pub.dev/packages/maplibre_gl?utm_source=chatgpt.com" \t "_blank)

State management: provider (đơn giản), hoặc riverpod nếu bạn muốn.

Local auth: lưu user bằng SQLite + bcrypt/scrypt-like hashing (hoặc crypto để hash + salt).

Optional: export/import data JSON để admin nhập list events.

# 2) Data model & SQLite schema (gợi ý)

SQL schema (SQLite):

-- users tableCREATE TABLE users (

id INTEGER PRIMARY KEY AUTOINCREMENT,

username TEXT UNIQUE NOT NULL,

email TEXT UNIQUE NOT NULL,

password\_hash TEXT NOT NULL,

display\_name TEXT,

created\_at TEXT DEFAULT CURRENT\_TIMESTAMP

);

-- events table (sự kiện lịch sử)CREATE TABLE events (

id INTEGER PRIMARY KEY AUTOINCREMENT,

title TEXT NOT NULL,

description TEXT,

date TEXT NOT NULL, -- ISO date like "YYYY-MM-DD"

year INTEGER NOT NULL,

month INTEGER NOT NULL,

day INTEGER NOT NULL,

lat REAL, -- optional geo

lng REAL,

summary TEXT,

source TEXT,

created\_at TEXT DEFAULT CURRENT\_TIMESTAMP

);

-- favorites table: user lưu event yêu thíchCREATE TABLE favorites (

id INTEGER PRIMARY KEY AUTOINCREMENT,

user\_id INTEGER NOT NULL,

event\_id INTEGER NOT NULL,

added\_at TEXT DEFAULT CURRENT\_TIMESTAMP,

UNIQUE(user\_id, event\_id),

FOREIGN KEY(user\_id) REFERENCES users(id),

FOREIGN KEY(event\_id) REFERENCES events(id)

);

-- password\_reset\_tokens (simple)CREATE TABLE password\_resets (

id INTEGER PRIMARY KEY AUTOINCREMENT,

user\_id INTEGER NOT NULL,

token TEXT NOT NULL,

expires\_at TEXT NOT NULL,

used INTEGER DEFAULT 0

);

Lưu ý: field date ở định dạng ISO giúp filter theo ngày/tháng/năm nhanh.

# 3) Cấu trúc project (gợi ý)

lib/

main.dart

app.dart

models/

event.dart

user.dart

services/

db\_helper.dart // sqlite open, migration

auth\_service.dart // register/login/hash, password reset

event\_service.dart // CRUD events, queries by date

providers/

auth\_provider.dart

event\_provider.dart

screens/

login\_screen.dart

register\_screen.dart

home\_screen.dart // list of days/events

event\_detail.dart

map\_screen.dart // MapLibre with markers

profile\_screen.dart

favorites\_screen.dart

widgets/

event\_card.dart

date\_picker\_row.dart

password\_strength\_bar.dart

assets/

styles, icons, map styles (if offline)

pubspec.yaml

# 4) Các màn chính & flow người dùng

Màn khởi động: kiểm tra local session → nếu chưa đăng nhập show Login.

Login/Register: username/email/password, password strength bar, password generator button.

Home: chọn năm hoặc ngày (calendar/day list) → hiển thị danh sách events của ngày đó.

Event Detail: show tiêu đề, ngày, full description, nguồn, vị trí (nếu có) và nút Lưu/Yêu thích.

Map view: Hiển thị các events (theo ngày/ tháng/ năm) như markers; marker tóm tắt (tooltip); click vào marker → mở detail.

Profile: cập nhật thông tin, đổi mật khẩu (cần xác thực mật khẩu cũ).

Forgot password: tạo token, dùng token để reset (có thể implement local-only: security question hoặc email nếu tích hợp backend).

# 5) Ví dụ pubspec.yaml (phần dependencies)

dependencies:

flutter:

sdk: flutter

provider: ^6.0.0

sqflite: ^2.2.0

path\_provider: ^2.0.0

path: ^1.8.0

crypto: ^3.0.0 # hash password

maplibre\_gl: ^0.20.0 # MapLibre Flutter plugin (tham khảo).

intl: ^0.18.0

mình khuyến nghị maplibre\_gl — plugin Flutter chính thức cho MapLibre (xem docs/quickstart để biết cách thêm style/API key). [Dart packages+1](https://pub.dev/packages/maplibre_gl?utm_source=chatgpt.com" \t "_blank)

# 6) Code mẫu quan trọng (copy — paste)

## a) Model Event (Dart)

// lib/models/event.dart

class Event {

final int? id;

final String title;

final String? description;

final String date; // YYYY-MM-DD

final int year;

final int month;

final int day;

final double? lat;

final double? lng;

final String? summary;

Event({

this.id,

required this.title,

this.description,

required this.date,

required this.year,

required this.month,

required this.day,

this.lat,

this.lng,

this.summary,

});

Map<String, dynamic> toMap() => {

'id': id,

'title': title,

'description': description,

'date': date,

'year': year,

'month': month,

'day': day,

'lat': lat,

'lng': lng,

'summary': summary,

};

factory Event.fromMap(Map<String, dynamic> m) => Event(

id: m['id'],

title: m['title'],

description: m['description'],

date: m['date'],

year: m['year'],

month: m['month'],

day: m['day'],

lat: m['lat'] == null ? null : (m['lat'] as num).toDouble(),

lng: m['lng'] == null ? null : (m['lng'] as num).toDouble(),

summary: m['summary'],

);

}

## b) DB helper (sqflite)

// lib/services/db\_helper.dart

import 'package:sqflite/sqflite.dart';

import 'package:path/path.dart';

import 'dart:async';

class DBHelper {

static final DBHelper \_instance = DBHelper.\_internal();

factory DBHelper() => \_instance;

DBHelper.\_internal();

static Database? \_db;

Future<Database> get db async {

if (\_db != null) return \_db!;

\_db = await initDb();

return \_db!;

}

Future<Database> initDb() async {

final databasesPath = await getDatabasesPath();

final path = join(databasesPath, 'in\_old\_days.db');

return await openDatabase(path, version: 1, onCreate: \_onCreate);

}

Future \_onCreate(Database db, int version) async {

await db.execute('''

CREATE TABLE users (

id INTEGER PRIMARY KEY AUTOINCREMENT,

username TEXT UNIQUE,

email TEXT UNIQUE,

password\_hash TEXT,

display\_name TEXT,

created\_at TEXT DEFAULT CURRENT\_TIMESTAMP

);

''');

await db.execute('''

CREATE TABLE events (

id INTEGER PRIMARY KEY AUTOINCREMENT,

title TEXT,

description TEXT,

date TEXT,

year INTEGER,

month INTEGER,

day INTEGER,

lat REAL,

lng REAL,

summary TEXT,

source TEXT,

created\_at TEXT DEFAULT CURRENT\_TIMESTAMP

);

''');

await db.execute('''

CREATE TABLE favorites (

id INTEGER PRIMARY KEY AUTOINCREMENT,

user\_id INTEGER,

event\_id INTEGER,

added\_at TEXT DEFAULT CURRENT\_TIMESTAMP,

UNIQUE(user\_id, event\_id)

);

''');

await db.execute('''

CREATE TABLE password\_resets (

id INTEGER PRIMARY KEY AUTOINCREMENT,

user\_id INTEGER,

token TEXT,

expires\_at TEXT,

used INTEGER DEFAULT 0

);

''');

}

}

## c) Password hashing (simple with salt) + strength & generator

// lib/services/auth\_service.dart (snippets)

import 'dart:convert';

import 'dart:math';

import 'package:crypto/crypto.dart';

String generateSalt([int length = 16]) {

final rand = Random.secure();

final bytes = List<int>.generate(length, (\_) => rand.nextInt(256));

return base64Url.encode(bytes);

}

String hashPassword(String password, String salt) {

final bytes = utf8.encode(password + salt);

final digest = sha256.convert(bytes);

return '${digest.toString()}:$salt'; // store salt after colon

}

bool verifyPassword(String password, String stored) {

final parts = stored.split(':');

if (parts.length != 2) return false;

final hash = parts[0];

final salt = parts[1];

final newHash = sha256.convert(utf8.encode(password + salt)).toString();

return newHash == hash;

}

// password generator

String generatePassword({int length = 12}) {

const lower = 'abcdefghijklmnopqrstuvwxyz';

const upper = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';

const digits = '0123456789';

const symbols = '!@#\$%^&\*()-\_=+[]{};:,.<>?';

final all = '$lower$upper$digits$symbols';

final rand = Random.secure();

return List.generate(length, (\_) => all[rand.nextInt(all.length)]).join();

}

// simple strength estimator: returns 0..4

int passwordStrength(String pw) {

int score = 0;

if (pw.length >= 8) score++;

if (pw.length >= 12) score++;

if (RegExp(r'[A-Z]').hasMatch(pw) && RegExp(r'[a-z]').hasMatch(pw)) score++;

if (RegExp(r'\d').hasMatch(pw) && RegExp(r'[^A-Za-z0-9]').hasMatch(pw)) score++;

return score; // 0 weak ... 4 very strong

}

Gợi ý: bạn có thể dùng zxcvbn port nếu muốn đánh giá mạnh mẽ hơn, nhưng trên mobile local simple estimator thường đủ cho UI.

# 7) Map screen (MapLibre) — ví dụ cơ bản

Mình dùng widget từ plugin maplibre\_gl (cách sử dụng tham khảo docs/plugin). Ví dụ hiển thị map và thêm marker:

// lib/screens/map\_screen.dart

import 'package:flutter/material.dart';

import 'package:maplibre\_gl/mapbox\_gl.dart'; // plugin maplibre\_gl reuses mapbox\_gl types

class MapScreen extends StatefulWidget {

final List<Event> events;

const MapScreen({Key? key, required this.events}) : super(key: key);

@override

State<MapScreen> createState() => \_MapScreenState();

}

class \_MapScreenState extends State<MapScreen> {

MaplibreMapController? mapController;

void \_onMapCreated(MaplibreMapController controller) {

mapController = controller;

\_addEventMarkers();

}

void \_addEventMarkers() {

for (final e in widget.events) {

if (e.lat != null && e.lng != null) {

mapController?.addSymbol(SymbolOptions(

geometry: LatLng(e.lat!, e.lng!),

iconSize: 1.0,

textField: e.title,

textOffset: Offset(0, 1.5),

));

}

}

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(title: Text('Bản đồ sự kiện')),

body: MaplibreMap(

initialCameraPosition: CameraPosition(

target: LatLng(14.0583, 108.2772), // trung tâm VN

zoom: 5.0,

),

styleString: 'https://demotiles.maplibre.org/style.json', // hoặc custom style

onMapCreated: \_onMapCreated,

),

);

}

}

Ghi chú: bạn có thể dùng styleString đến style JSON (MapTiler, StadiaMaps, hoặc style lưu trên server). MapLibre hỗ trợ clustering, layer filtering theo thời gian nếu bạn muốn hiển thị sự kiện theo ngày/ tháng (xem plugins & examples). [docs.stadiamaps.com](https://docs.stadiamaps.com/native-multiplatform/flutter-maplibre-gl/?utm_source=chatgpt.com" \t "_blank)

# 8) Tối ưu hiển thị & UX

Hiển thị danh sách theo ngày: dùng ListView grouped by date; hỗ trợ tìm kiếm (title/summary).

Trên map: hiển thị marker cluster khi nhiều markers gần nhau (plugin hỗ trợ clustering).

Khi người dùng bấm marker: hiển thị bottom sheet tóm tắt (title + date + nút Xem chi tiết).

Cho phép offline: lưu tiles (offline) hoặc cache GeoJSON; MapLibre có hướng dẫn offline caching (cần cấu hình thêm).

Sync/backup: export events to JSON hoặc (nếu muốn) sync lên backend (future).

# 9) Lưu favorite & local auth flow

Khi user add favorite: insert vào favorites với user\_id.

Khi logout: clear session token (local secure storage hoặc SharedPreferences).

Reset password: nếu app local-only, bạn có thể implement "security question" hoặc gửi token tới email (nếu có backend). Nếu muốn local reset, lưu token trong password\_resets table và validate expires\_at.

# 10) Triển khai & checklist trong VS Code

Tạo Flutter project: flutter create in\_old\_days

Thêm dependencies vào pubspec.yaml (xem ở trên).

Implement DB helper, models, services.

Tạo màn đăng ký/login; test register → db record created; test login -> verify hashed password.

Tạo import script JSON để nạp danh sách event (admin).

Map: lấy API/style URL để styleString; test map on Android/iOS emulator (MapLibre cần config native on iOS/Android per plugin docs).

Test marker click → mở detail.

Add favorites, profile update UIs.

Build release APK / test on device.

# 11) Các lưu ý kỹ thuật & tài liệu tham khảo nhanh

Plugin Flutter MapLibre: dùng maplibre\_gl hoặc maplibre — xem pub.dev và ví dụ để biết cách cài native config (Android/iOS). [Dart packages+1](https://pub.dev/packages/maplibre_gl?utm_source=chatgpt.com" \t "_blank)

Quickstart & examples (MapTiler / StadiaMaps) có hướng dẫn cụ thể cách thêm style và offline caching. [docs.stadiamaps.com+1](https://docs.stadiamaps.com/native-multiplatform/flutter-maplibre-gl/?utm_source=chatgpt.com" \t "_blank)

Nếu bạn cần clustering, look up MapLibre clustering or cluster plugin (nhiều ví dụ có sẵn).

# 12) Muốn mình làm gì tiếp theo?

Mình có thể (chọn 1 hoặc nhiều):

Viết sẵn một **skeleton Flutter project** (những file Dart chính: main.dart, db\_helper, auth\_service, home\_screen, map\_screen) mà bạn chỉ cần clone & chạy trong VS Code.

Viết script **import JSON** để bạn nạp sẵn danh sách events theo năm (mình sẽ cho format JSON mẫu).

Viết UI màn Login/Register + full auth flow + password strength bar + generator (hoàn chỉnh).

Hướng dẫn chi tiết cấu hình Android/iOS cho maplibre\_gl (manifest, permissions, keys).

Bạn muốn mình làm **(1)**, **(2)**, **(3)** hay **(4)** trước? Hoặc nếu muốn, mình có thể **tạo luôn skeleton code** ngay bây giờ (mình sẽ dán bộ file chính để bạn copy vào VS Code).