Heart App — Flutter MVVM Architecture

Heart App is a Flutter application that demonstrates clean architectural design using the **MVVM pattern** and **Dependency Injection (DI)** through `get_it`.

It visualizes a *heart filling animation* (0 % → 100 %) that represents progress toward a goal. Users can **tap to start, pause, or resume**, and progress **persists across sessions** via `SharedPreferences`.This project highlights best practices for testable, scalable Flutter apps with separation of concerns and reactive UI updates.

Code Organization

```
lib/
│
├── app.dart                    → Root MaterialApp (UI entry)
├── main.dart                   → Initializes DI & Provider
│
├── model/                      → Data layer
│   ├── heart.dart                  → Core Heart entity (immutable
model)
│   ├── repository/
│   │   ├── heart_repository.dart      → Repository interface
│   │   └── heart_repository_impl.dart → Implementation via local driver
│   └── services/
│       ├── local_storage.dart         → SharedPreferences wrapper
│       ├── heart_local_driver.dart    → Adapter for persistence
│       └── heart_fill_service.dart    → Timer-based filling service
│
├── viewmodels/
│   └── heart_view_model.dart     → Application logic, state &
persistence
│
└── ui/
    ├── screens/
    │   ├── heart_screen.dart     → Main screen (progress + interaction)
    │   └── success_screen.dart   → Post-completion screen
    └── widgets/
        ├── heart_painter_twoIcon+ClipRect.dart
        ├── HeartPainterFill.dart
        ├── HeartPathFill_ClipPathHeart.dart
        └── _LiquidHeartChartState.dart
```

Design Patterns

1) MVVM (Model–View–ViewModel) https://docs.flutter.dev/app-architecture/guide

- **Model:** Business logic and data (`Heart`, Repository layer).
- **ViewModel:** Exposes reactive state, manages timer and persistence.
- **View:** Stateless UI that observes ViewModel through Provider.

2) Repository Pattern

https://www.geeksforgeeks.org/system-design/repository-design-pattern/

- `HeartRepository` defines abstract data operations.
- `HeartRepositoryImpl` delegates to `HeartLocalDriver`, allowing future replacement (Firebase, REST API, Hive, etc.) without ViewModel changes.

3) Dependency Injection (GetIt)

https://medium.com/%40ibrahimtalhahurata/dependency-injection-in-flutter-with-get-it-5f657e068395
https://medium.com/@lanresamuel2002/mastering-dependency-injection-in-flutter-with-get-it-a-comprehensive-guide-944a7ac57df5
https://pub.dev/packages/get_it

Centralized object graph for testability and scalability:

```
sl.registerLazySingleton<HeartRepository>(
  () => HeartRepositoryImpl(sl<HeartLocalDriver>()),
);
sl.registerFactory<HeartViewModel>(
  () => HeartViewModel(repo: sl(), filler: sl()),
);
```

State Management

State	Description
empty	Initial or cleared.
progressing	Timer running, filling the heart.
paused	User paused midway.
completed	Heart fully filled.

`Provider` + `ChangeNotifier` drive reactive updates between `HeartViewModel` and UI.

References : https://docs.flutter.dev/data-and-backend/state-mgmt/simple

https://medium.com/flutter-community/providers-with-changenotifiers-752593f082a5

https://stackoverflow.com/questions/77762643/minimal-example-using-changenotifierprovider

https://www.geeksforgeeks.org/flutter/flutter-provider-package/

ViewModel Responsibilities

- Starts/stops the timer every second.
- Increments progress by **10 % per tick**.
- Saves state + progress persistently.
- Calculates derived values:

```
double get percent => (progress / capacity) * 100;
```

Data Persistence Flow

1. **ViewModel → Repository:** `repo.save(heart, stateIndex)`.
2. **Repository → Driver:** delegates to `HeartLocalDriver`.
3. **Driver → LocalStorage:** persists `progress`, `capacity`, `stateIndex`.
4. **Startup:** ViewModel restores previous values and adjusts state (paused/resumed).

References (drawing heart)

1. https://github.com/JordanADavies/liquid_progress_indicator
2. https://flutterexperts.com/custom-progress-indicator-in-flutter
3. https://stackoverflow.com/questions/57756692/flutter-drawing-a-heart-shape-with-custom painter
4. https://api.flutter.dev/flutter/rendering/CustomPainter-class.html
5. https://api.flutter.dev/flutter/widgets/CustomPaint-class.html
6. https://api.flutter.dev/flutter/dart-ui/Canvas-class.html
7. https://api.flutter.dev/flutter/dart-ui/Canvas/clipPath.html