# Flutter Clean Architecture: Per-Feature Structure

## Overview

This guide explains how to structure a Flutter project using Clean Architecture with a per-feature organization. The goal is to ensure separation of concerns, maintainability, testability, and scalability.

## Folder Structure (Per Feature)

```
lib/
 core/              # Shared logic
 features/
   auth/
      data/
      domain/
      presentation/
 injection_container.dart
 main.dart
```

## Presentation Layer

Handles UI and state management. Includes pages, widgets, and BLoC/Cubit logic. No business logic.

## Domain Layer

Pure business logic. Includes entities, use cases, and abstract repositories. Framework-agnostic.

## Data Layer

Responsible for data fetching and persistence. Includes models, data sources, and repository implementations.

## Core Layer

Holds reusable utilities such as error handling, constants, theming, and network helpers.

# Flutter Clean Architecture: Per-Feature Structure

## Injection Layer

Manages dependency injection using tools like GetIt or Riverpod.

## Best Practices

- Follow Dependency Inversion Principle
- Keep UI, business logic, and data separate
- Avoid bloated files and promote modularity
- Use DTOs for data transfer
- Test each layer independently
- Place shared logic in the core folder