**MVVM (Model-View-ViewModel)**

**Components:**

1. **Model**: Represents the data and business logic. It is similar to the Model in MVC.
2. **View**: Represents the UI and is responsible for displaying data. It binds to the ViewModel to get data and commands.
3. **ViewModel**: Acts as a bridge between the Model and the View. It holds the UI logic and state. The ViewModel communicates with the Model to fetch or update data and exposes properties and commands to the View.

**Pros:**

* **Data Binding**: Provides a more straightforward way to bind data to the UI
* **Testability**: ViewModel is easier to test since it doesn’t directly depend on the View.
* **Decoupling**: Better separation between UI and business logic compared to MVC.

**Cons:**

* **Difficult to understand**: More complex than MVC, especially for developers new to data binding.
* **Overhead**: May involve more boilerplate code and setup.

**Usage in Flutter:**

* MVVM can be implemented in Flutter using packages like provider, riverpod, or flutter\_bloc for state management.
  + **Model**: Defines the data structure and business logic.
  + **View**: Flutter Widgets that display data.
  + **ViewModel**: Dart classes that manage the state and logic, typically using ChangeNotifier or similar.