IOS Module Architecture → MVC

Weather Model

In this case, they don't have face to face connection

Get Data

Update Data

Call Native Methods by Click on Buttons

View (Flutter)

Weather Controller

Sending Random Data to Flutter App
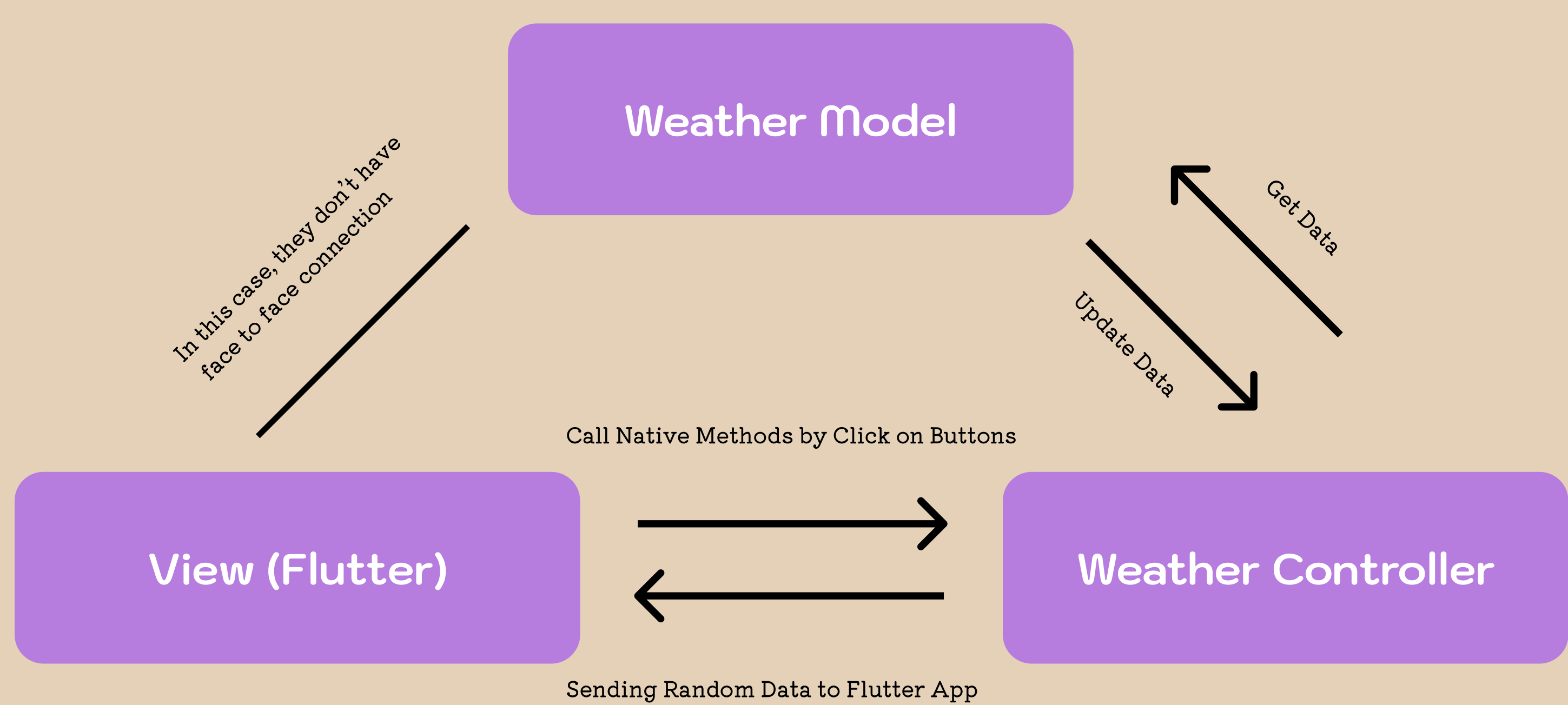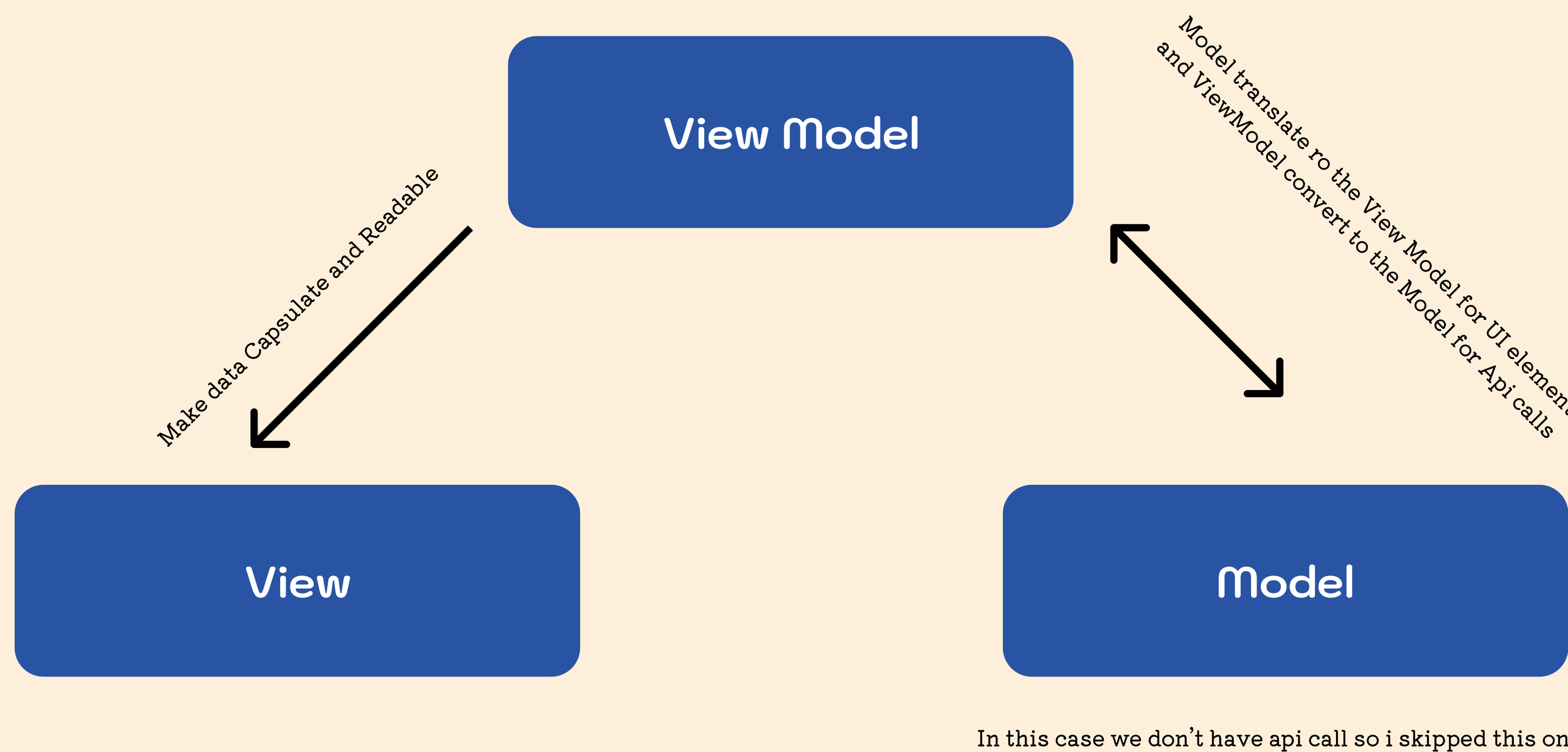
I chose the Model-View-Controller (MVC) architecture for my Swift code because it provides a clear separation of concerns, making the code more maintainable, scalable and testable.
- Model → Handles data and business logic (e.g., fetching or processing data).
- View → Manages UI elements and user interactions.
- Controller → Acts as a mediator between Model and View

Flutter Architecture → MVVM

View Model

Make data Capsulate and Readable

Model translate ro the View Model for UI elements and ViewModel convert to the Model for Api calls

View

Model

In this case we don't have api call so i skipped this one

I chose MVVM (Model-View-ViewModel) along with GetX in Flutter to enhance state management, dependency injection, and navigation handling.
- MVVM: Provides a clean separation of concerns, making the code more structured and testable.
- GetX: A lightweight and powerful package for reactive state management, dependency injection, and simplified navigation.

Note: WeatherController in flutter is just a mixin for make code cleaner it's not actually MVC Archivecture.