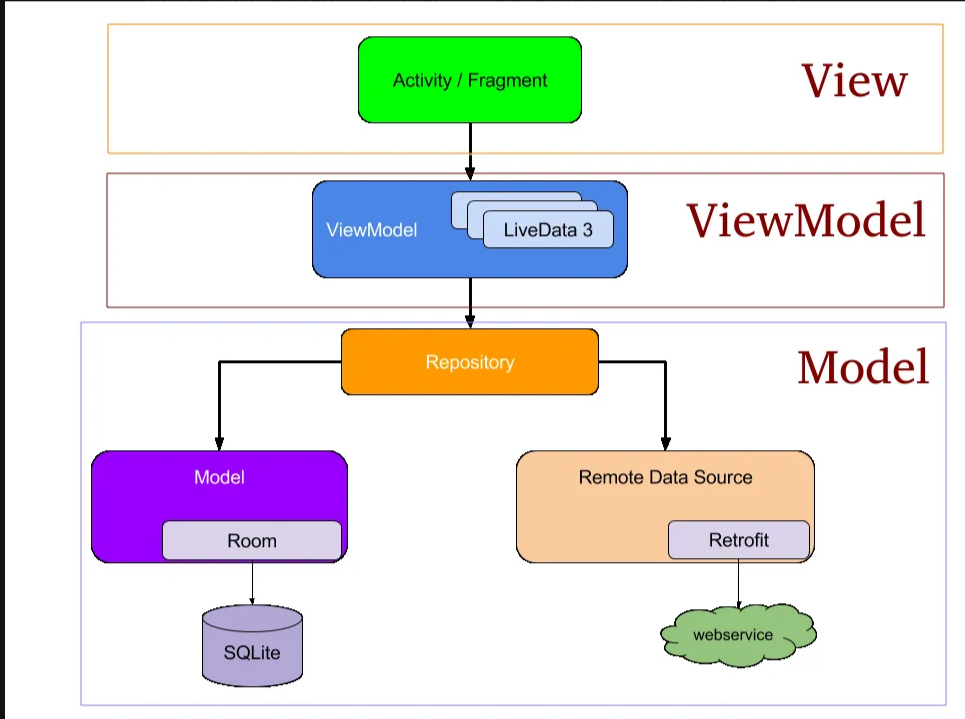
**Chapter No 4**

**Architecture**

**4. Architecture**

The primary static and dynamic components of our application are represented by the architecture. By highlighting the key components and excluding extraneous information, it offers a clear picture of the overall system. Because the Model-View-ViewModel (MVVM) architecture aids in the construction of a neat, orderly, and maintainable structure, we are using it.



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

Model-View-ViewModel, or MVVM, is a design pattern that helps maintain code organization and cleanliness. This pattern aids in creating manageable, testable, and maintainable apps in Flutter. It divides the application into three sections: View (the user interface), Model (the data and logic), and ViewModel (which links Model and View). This increases the code's flexibility and ease of use, particularly in collaborative projects.

**4.1.1 Model**

The data for your app is kept in the Model. To store and handle this data, you can make a Dart class in Flutter.

**4.1.2 View**

What the user sees on the screen is called the View. Widgets that show the data supplied by the ViewModel are used in Flutter.

**4.1.3 ViewModel**

The View and the Model are connected via the ViewModel. It manages user actions and the logic of the application. It is written in Flutter as a Dart class that prepares data for the View and controls the state of the application.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_­\_\_\_\_\_\_**