The MVC architecture stands for Model-View-Controller and is a software design pattern commonly used for developing user interfaces, particularly in web and mobile apps. It separates an application into three interconnected components:

1. Model:

1. The Model represents the data or business logic of the application.

2. It is responsible for directly managing the data, logic, and rules of the application. For example, if you're working with a database, the Model handles communication with the database and retrieves the necessary data.

3. The Model is independent of the user interface, meaning it doesn’t care how data is displayed to the user.

Example: In a shopping app, the Model could be the data representing the product list, the user's cart, and the operations for adding or removing items from the cart.

2. View:

1. The View is responsible for the presentation of data (UI or User Interface).

2. It defines how the data from the Model is displayed to the user. However, the View does not contain logic on how the data is manipulated—only how it’s shown.

3. The View listens to the Controller for instructions and updates the UI based on the data received from the Model.

Example: In a shopping app, the View could be the screens that display the product details, cart items, and the checkout process.

3. Controller:

1.The Controller acts as an intermediary between the Model and the View.

2. It receives user inputs (like clicks or form submissions), processes those inputs, interacts with the Model to retrieve or update data, and then updates the View to reflect any changes.

3. The Controller doesn’t deal with UI details but controls how data flows between the Model and View based on user actions.

Example: In a shopping app, the Controller could handle the logic for adding an item to the cart when a user clicks "Add to Cart." It interacts with the Model to update the cart and then instructs the View to refresh and display the updated cart.

Summary of Workflow:

View: Displays the data (UI).

Controller: Handles the user input and interacts with the Model.

Model: Manages the data and business logic.

In an Android app using MVC, you might have:

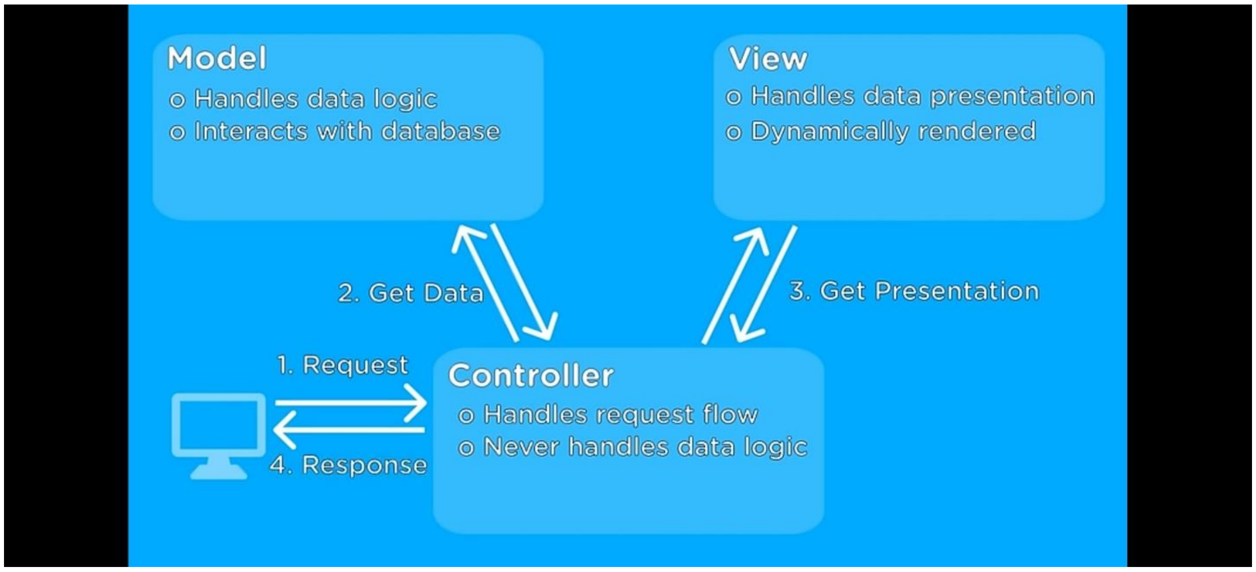
The Model as Java classes that manage app data (such as the user's profile).

The View as XML layouts or Java/Jetpack Compose code that define the UI (like the screen for displaying a user's profile).

The Controller as activity/fragment classes that handle user interactions (like buttons or forms) and update the UI based on data from the Model.

This architecture helps keep your code modular and easier to maintain.

Example:


Model
o Handles data logic
o Interacts with database
View
o Handles data presentation
o Dynamically rendered
2. Get Data
3. Get Presentation
1. Request
Controller
o Handles request flow
o Never handles data logic
4. Response