**Model-View-Controller (MVC):**

* **Structure:**
  + **Model:** Manages application data and business logic.
  + **View:** Displays the data and interacts with the user.
  + **Controller:** Handles user input, updates the Model, and selects the appropriate View.
* **Characteristics:**
  + The Controller acts as the intermediary between the View and Model.
  + Minimal support for two-way data binding.
* **Use Case:**
  + Best for simple applications with well-defined interactions between UI and logic.
  + **Example:** A blogging platform where:
    - **Model:** Manages post data.
    - **View:** Displays posts.
    - **Controller:** Handles user actions like creating or editing posts.

**Model-View-ViewModel-Controller (MVVC):**

* **Structure:**
  + **Model:** Contains the application's core logic.
  + **View:** Presents the user interface.
  + **ViewModel:** Acts as a middle layer for two-way data binding and logic for the View.
  + **Controller:** Coordinates high-level application flow and invokes ViewModel or Model logic.
* **Characteristics:**
  + Enables two-way data binding between the View and ViewModel.
  + The Controller focuses on orchestration, while the ViewModel manages UI state.
* **Use Case:**
  + Ideal for complex, dynamic applications with interactive UIs.
  + **Example:** A stock trading app where:
    - **Model:** Manages stock data.
    - **ViewModel:** Formats stock prices and updates the UI in real-time.
    - **View:** Displays a dashboard with interactive charts.
    - **Controller:** Handles routing and user actions like placing trades.