**MVC VS MVP**

**MVC (Model View Controller)**

Design pattern that divides the application into three layers: model, view controller

**Model:**

The model is the brain of the application .it is represented by set of class responsible for the data and business logic

**View:**

Responsible for rendering the UI interface .it is completely stateless and usually without any logic .it is only responsible for the data that is received from the controller

**Controller:**

It acts as a mediator between view and model. It receives input from users via the View, then process the user’s data with the help of Model than update the view based on the results. The controller is responsible for determining which view to be displayed

For example if a user click on a button ,the view tells the controller that the user clicked a button than the controller decides how to interact with the model ,than based on the data retrieved from the model controller decides how to return the view

**MVP (Model View Presenter)**

Design pattern that divides the application into three layers: model, view, presenter

**Model:**

Same as MVC

**View:**

Contains almost zero logic , it is only responsible for displaying data it receive from presenter .the view holds only reference of presenter and doesn’t know about model.

The view communicates with presenter through an interface

**Presenter:**

The presenter receives all the UI events on behalf of the view .the presenter communicates with model and based on the data received it sends the result back to view

Presenter communicates with the view through an interface. Unlike the controller in MVC where it is tied to view, the presenter and view are completely decoupled from each other and there only way of communication is through interfaces. This way we can easily change the view without any effect on the other layers.

**Conclusion:**

1-Both MVC and MVP does a great job separating the model from the view .When it comes to unit testing the model can be easily test in both cases since it’s not dependent on view however testing the controller might be harder since its tied to the view .on the other hand we can easily test the presenter code since it’s not tied to any android APIs.

2-For small apps using MVP for instance could add more complexity for the app .However for large project in android MVP can provide a cleaner, better maintained and more testable code