មជ្ឈមណ្ឌលកូរ៉េ សហ្វវែរ អេច អរ ឌី

Korea Software HRD Center

MVP Android Clean Architecture Design Pattern

1. Problem
2. Why MVP?
3. MVP Responsibility

1. Problem

- Activity / Fragment with millions lines of codes
- Single Responsibility Principle

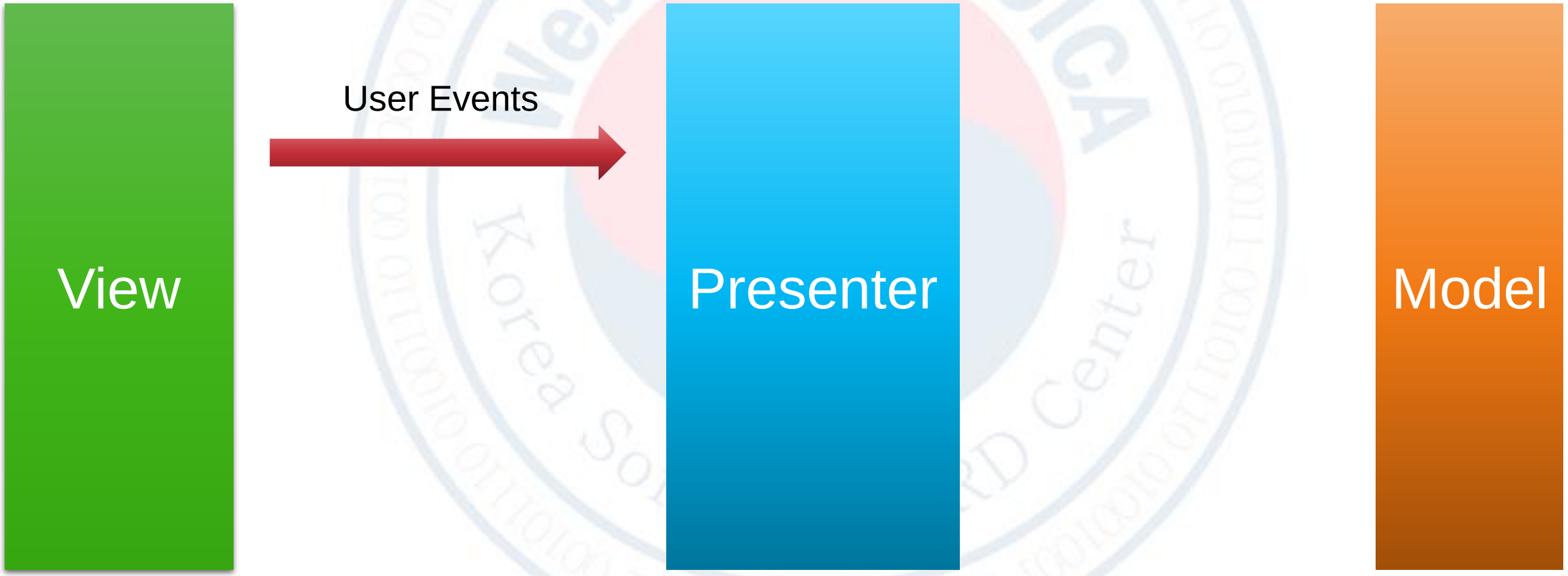
1. Why MVP?

- Totally decoupled
- Each Class has Single Responsibility
- Each layer can easily testable separately
- Easy to maintain

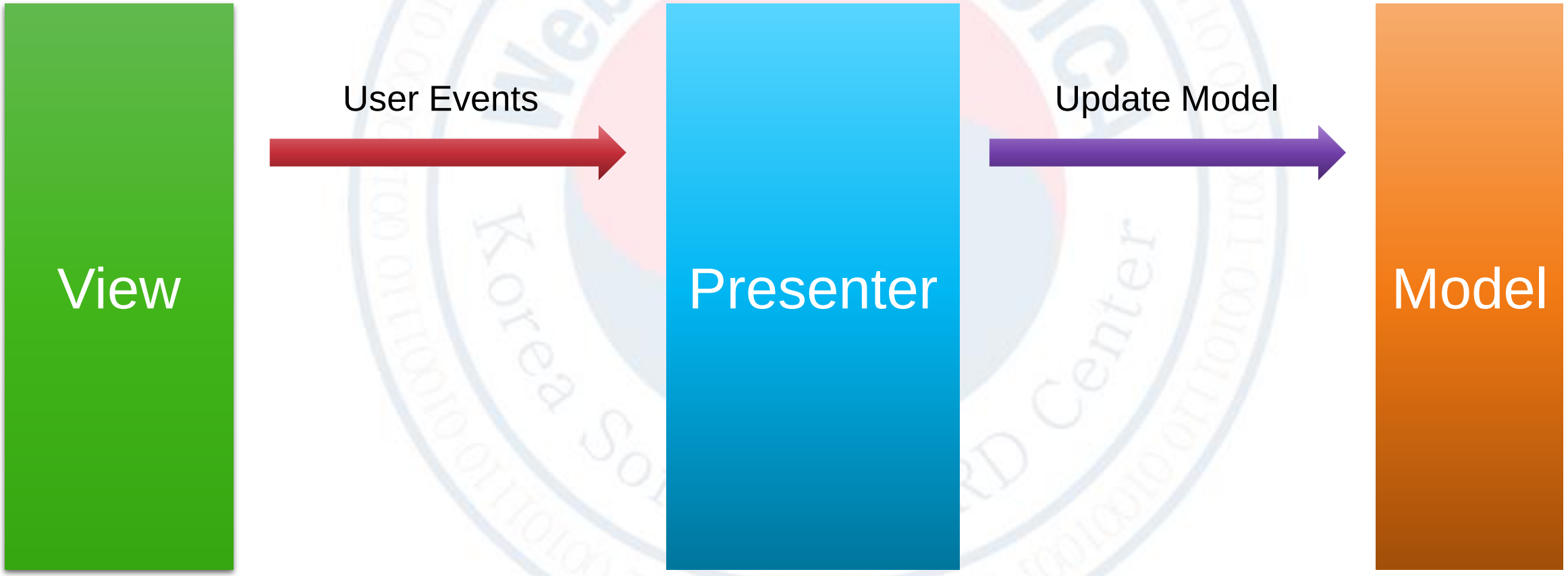
3. MVP Responsibility

- Model defines the data to display
- View is a passive interface that displays data and route user events
- Presenter retrieves data from Model and notify the view to display it ☺

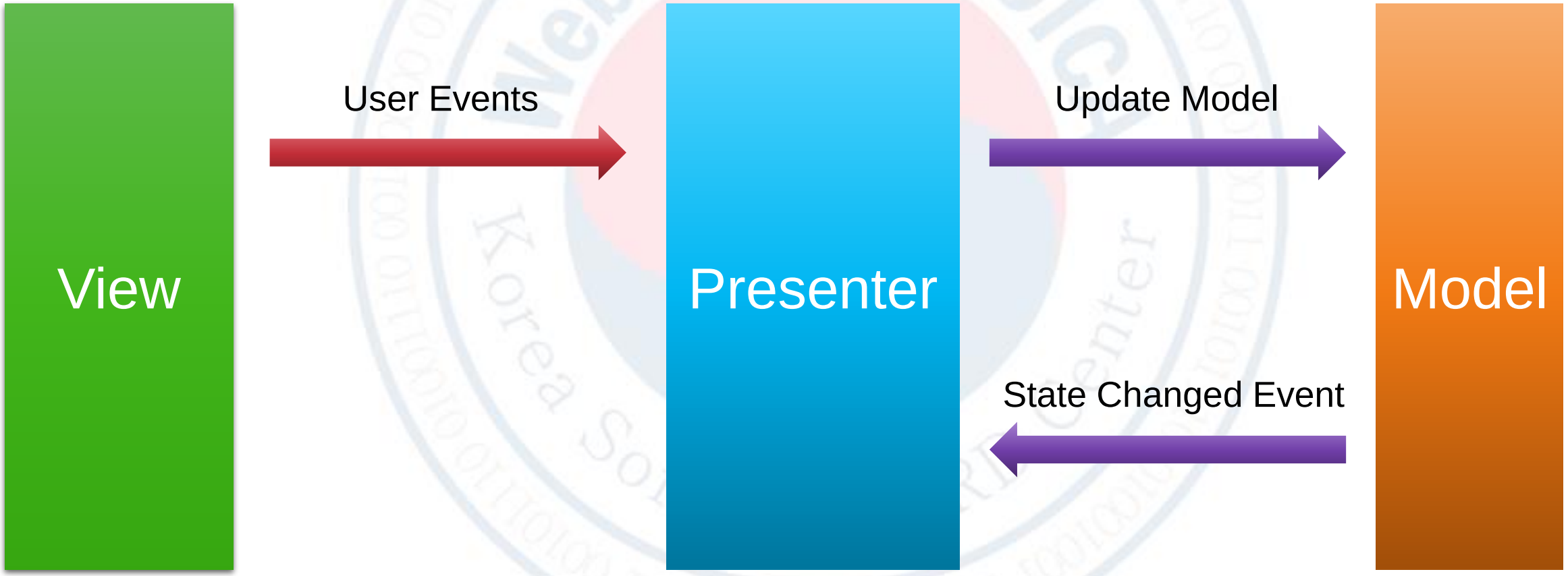
3. MVP Responsibility



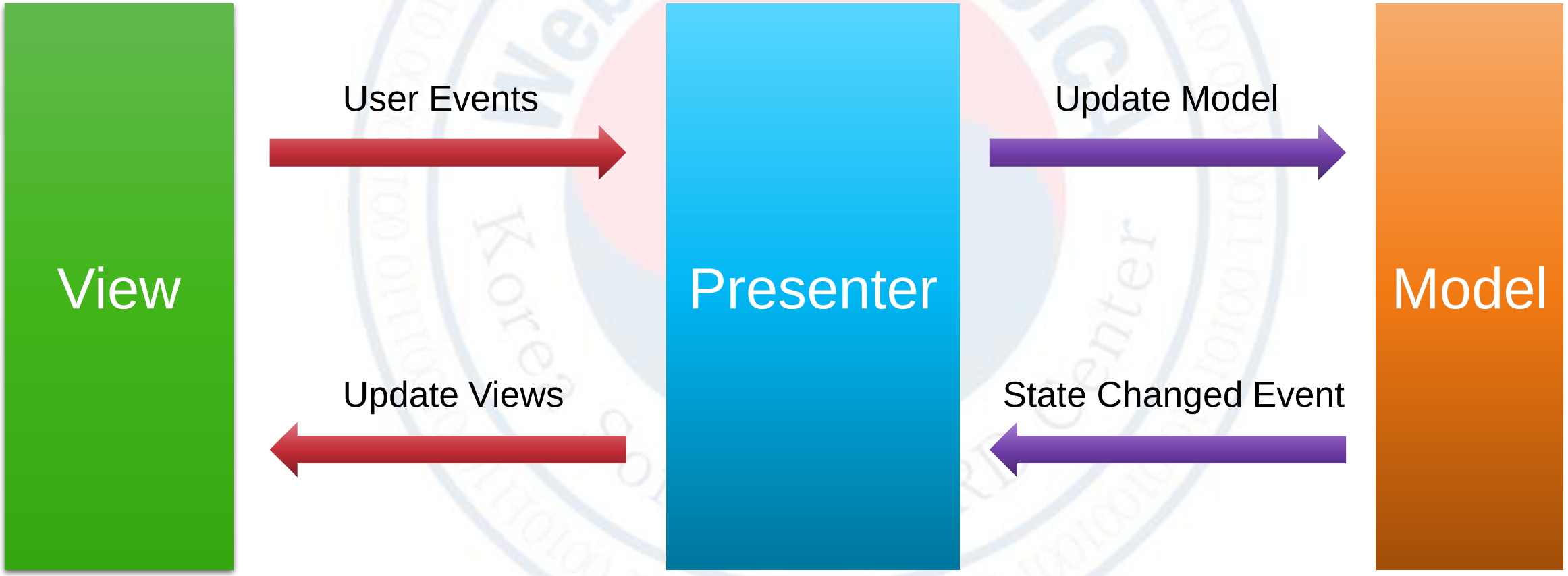
3. MVP Responsibility



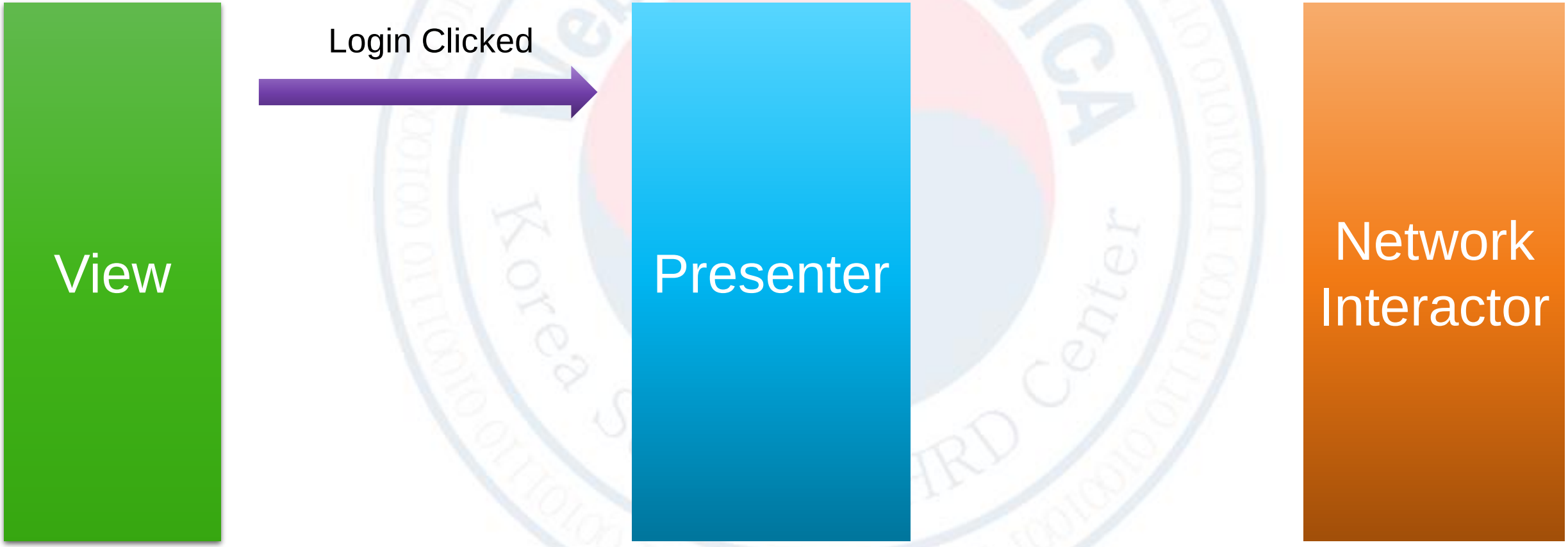
3. MVP Responsibility



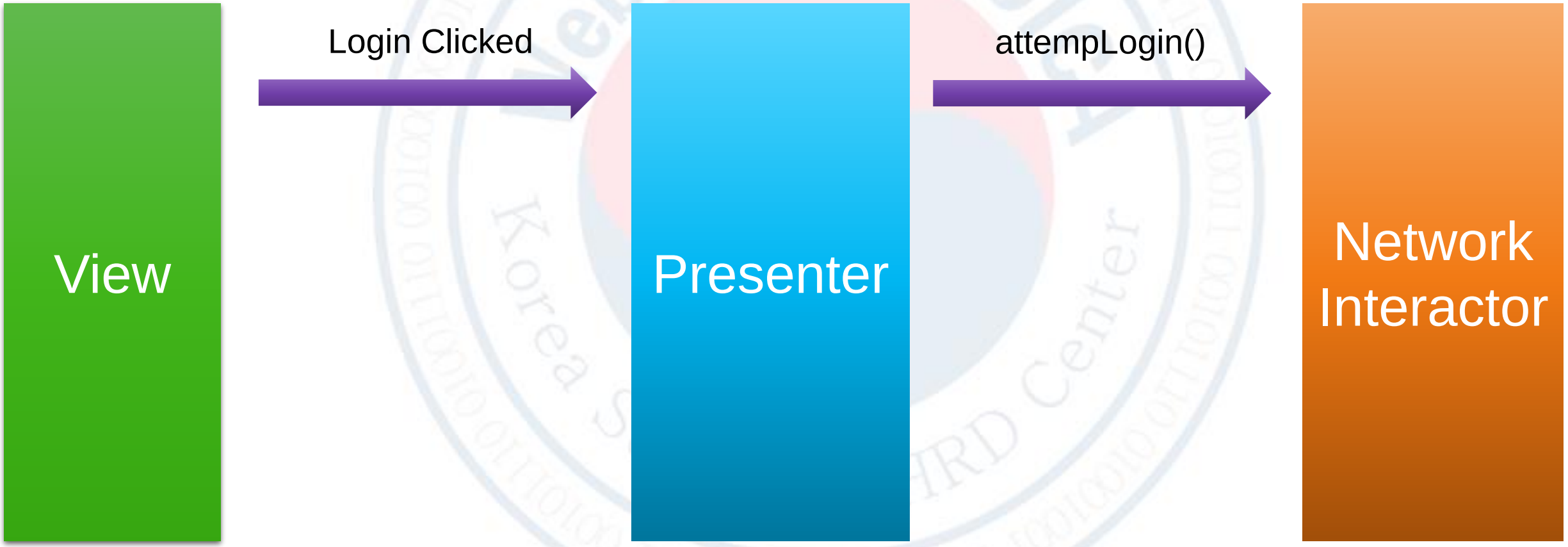
3. MVP Responsibility



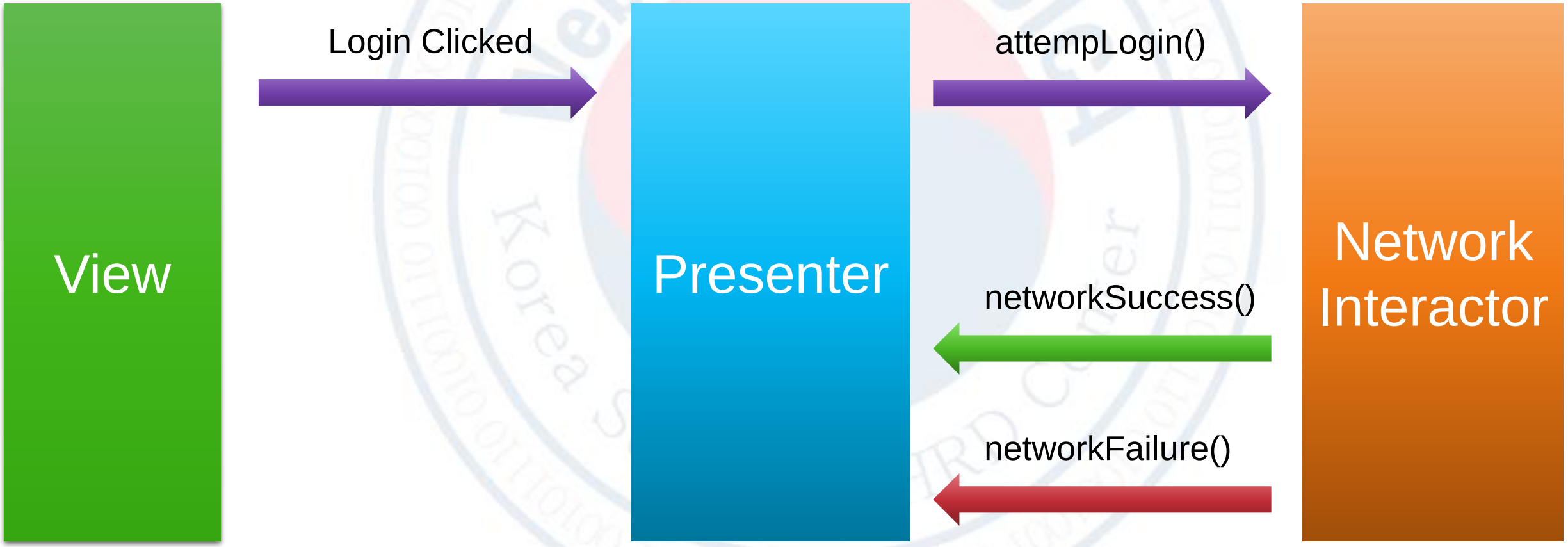
Login MVP



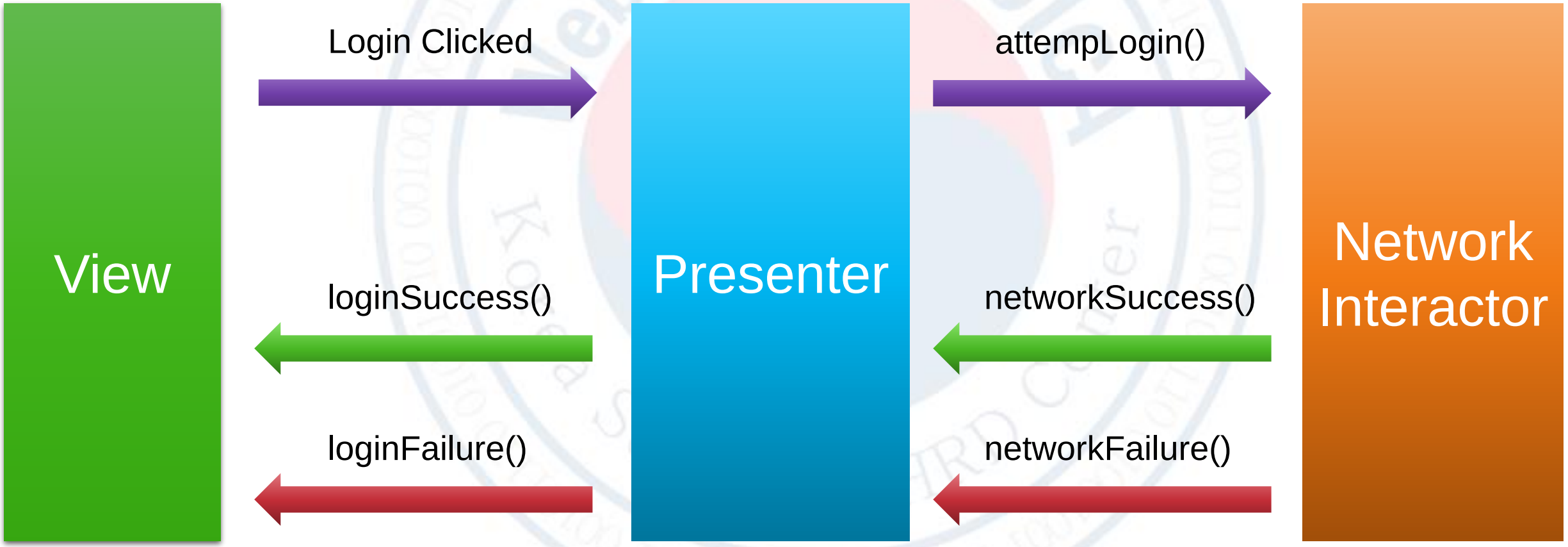
Login MVP


View
Login Clicked
Presenter
attempLogin()
Network
Interactor

Login MVP


View
Login Clicked
Presenter
attempLogin()
networkSuccess()
networkFailure()
Network Interactor

Login MVP


View
Login Clicked
loginSuccess()
loginFailure()
Presenter
attempLogin()
networkSuccess()
networkFailure()
Network Interactor

4. Rule

View

- Activity, Fragment
- Contain a reference of Presenter
- Delegate events from UI to Presenter (OnClick, lifecycle events…)
- Contain methods which control the presentation of data

    ex: Show / Hide loading layout, update RecyclerView, etc…

4. Rule

Presenter

- Is a simple Java class
- Must not contain any Android Dependency
- Is the Middle-man between View and Model
- Contain a reference of View and Model
- Update UI By calling the View
- Connection from / to Presenter are done via Interface

4. Rule

Interactor (Model)

- Is the gateway towards the business logic
- Must not contain any Android Dependency
- Contain methods for data retrieval