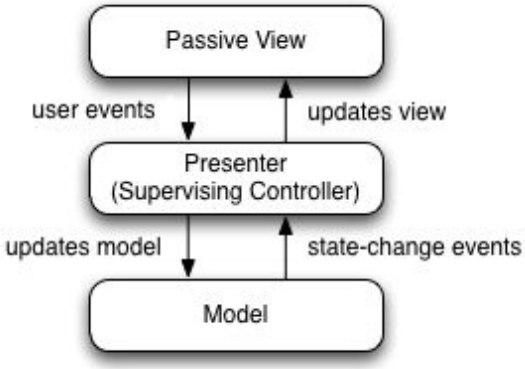
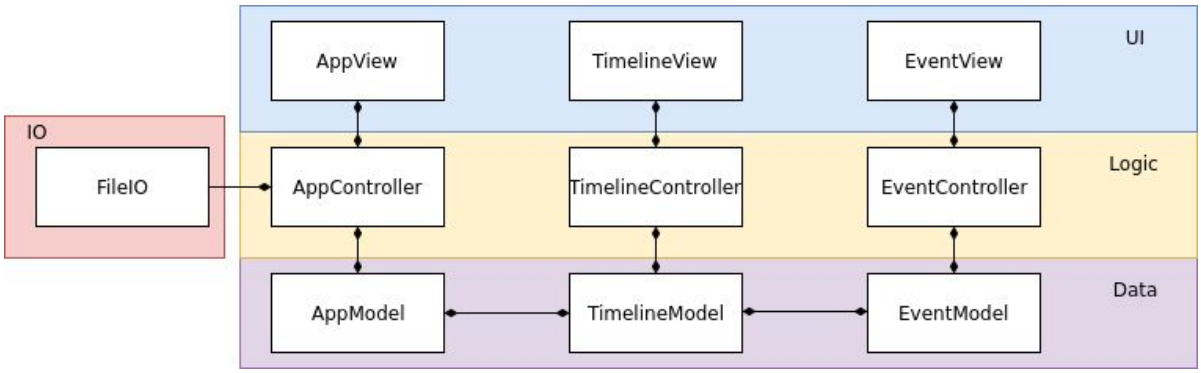
## High Level Design

The app will use the MVP pattern to handle interactions between the user interface and the data. This separates logic and data (Timelines and Events) from the UI, and as such it is easier to work on as a group and more or less the same code can be used independently of what the UI looks like.




Source: https://en.wikipedia.org/wiki/Model–view–presenter

## Classes/modules




- AppView represents the entire UI and contains all the visual elements that do not belong to a specific timeline or event.
- AppController is responsible for creating and deleting TimelineViews, TimelineControllers and TimelineModels
- AppModel contains a list of TimelineModels

- TimelineViews represents the UI elements associated with the timelines
- TimelineControllers is responsible for creating and deleting EventModels, EventControllers and EventViews as well as updating the TimelineModel based on changes in the TimelineView and the other way around (See MVP diagram)
- TimelineModels contains a startDate, an endDate and a list of EventModels

- EventViews represents the UI elements associated with EventModel properties, such as the name, startDate, endDate, and so on
- EventController handles the communication and logic between EventViews and EventModel in the same way as TimelineController does for timelines
- EventModels contains the event properties specified in the requirements (startDate, endDate, name, ...)

- FileIO has only been loosely defined, it is responsible for serializing/deserializing the AppModel as well as saving this to disk in a CSV file

## Example UI, annotated

*not the actual UI that will be used