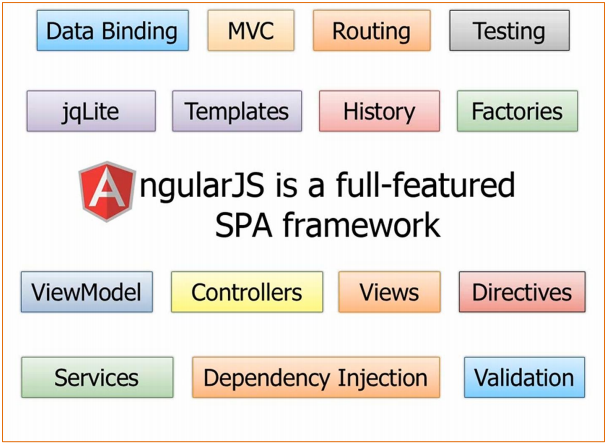
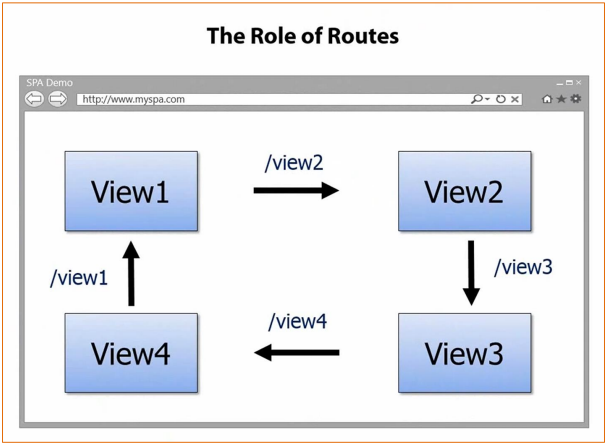
Problem 3

## Design a client-side architecture for a white-label web application that is used by thousands of unique clients. Explain and/or diagram how you would go about architecting this application. How would you design this system? What are the pros and cons to your approach? Tradeoffs? What design patterns would you use and why?

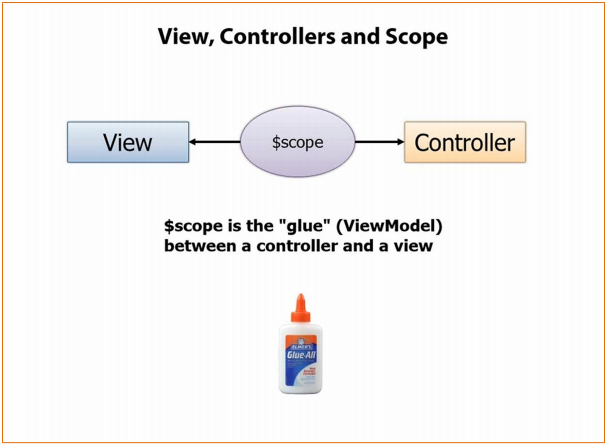
* Using AngularJS I would implement a Single Page Application in which we can have a shell page which can load multiple views representing our different clients.



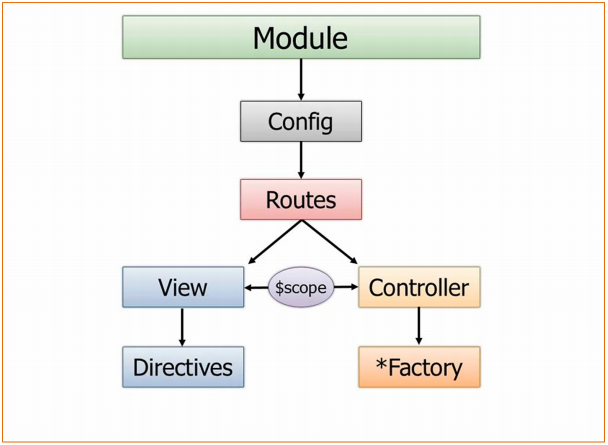
* Implemented using Routes, you can load different view for each individual client as needed with the necessary styles in terms of typography, colors, borders, etc.



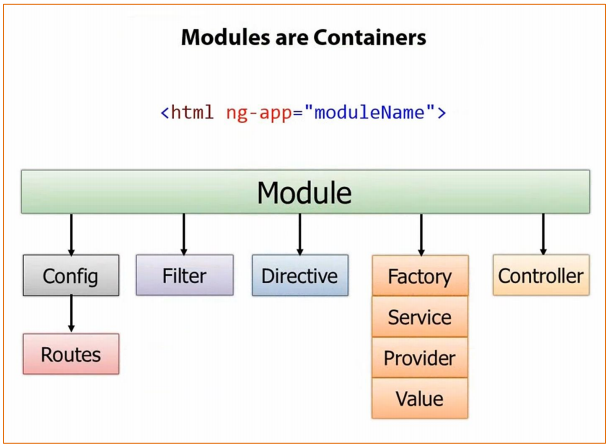
* Imagine multiple interchangeable views on a single page which can be switched in and out give a specific client or scenario.
* There would be minimal coding involved as Views are simply partials or small segments of code that you can add or remove from a shell page as needed.



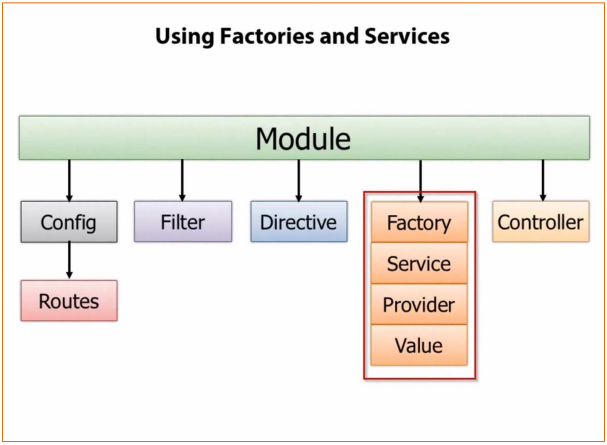
* My design would utilize the use of Modules and Controllers to config the Routes to show the specific view’s as needed.



* You can filter out the views as needed based on different views for mobile, maybe one specific to iPad or Surface or something else.



* Using Factory’s, Services or Provider’s you can even encapsulate data functionality
  + With the **factory** you actually create an object inside of the factory and return it.
  + With the **service** you just have a standard function that uses the **this** keyword to define function.
  + With the **provider** there’s a **$get** you define and it can be used to get the object that returns the data.



* The key would be code organization, a major downside of this design patter would be not making sure our directory structure is sound and modular.
  + It gets messy and we start losing times searching for files.
* Keep controllers simple, by implement inheritance pattern using Class.js utility. Due to the fact that Angular is missing two critical things. Structured class approach and class inheritance.
* By using the 3 frameworks below, our design can come together very efficiently.

