Layering

When using layering each part of the system relies on some lower layer of the system. Lower layers are unaware of higher layers Layers usually hide their own lower layers from the higher layers that use it.

- You can understand a single layer as a coherent whole without knowing much about the other layers
- You can substitute layers with alternative implementations of the same basic services. E.g. The underlying ISP infrastructure can change without the FTP service being affected.
- You minimize dependencies between layers. Changes on lower layers should only affect those directly dependent on it.
- Layers are a good place for standardization
- One layer can be used in multiple higher-level services. E.g. TCP/IP is used by FTP, telnet, SSH, and HTTP. This makes it so these services don't have to write their own lower level protocols.

Downsides of layering

- Layers encapsulate some, but not all, things well. Sometimes this can cause cascading changes. E.g. A field in the UI needs to be stored in the DB and thus needs to exist in every layer in between.
- Extra layers can harm performance. Data typically needs to be transformed from one representation to another between layers. This is typically more than offset by the optimization that takes place within the layer. These optimizations will speed everything higher-level that depends on it up.

How to decide what should be layers? - The most difficult part of implementing layering.

The Three Principal Layers

1. Presentation (Frontend)
2. Domain (Backend)
3. Data (Databases, Data Sources)

Presentation logic - how interaction between the user and the software is handled.

Responsibilities:

Layer	Responsibilities
Presentation	Displaying of information, provisioning of services, handling user requests (mouse clicks, keyboard actions), HTTP requests, command-line invocations, batch API
Domain	Houses the logic of the application
Data Source	Manages communications with databases, messaging systems, transaction managers, and other packages

Domain/Business Logic - work and rules that need to be done specifically for a particular domain of a business.

The domain and data source layers should NEVER depend on the presentation layer.

Example: Products that sold greater than 10% this quarter than the last were to be highlighted in red.

DON'T: Put logic in the presentation to calculate last quarters sales and color it red if greater than 10%.

DO: Add data to the response indicating a significant increase in sales.

This breaks the problem into two parts, Deciding whether something should be highlighted red and choosing how to highlight.