Web Application Architectures

* We have already seen that modern web applications involve a significant amount of complexity, particularly on the server side.
* A typical web application involves numerous protocols, programming languages and technologies spread throughout the web stack.
* Developing, maintaining and extending a complex web application is extremely difficult – but, building it using a foundation of solid design principles can simplify each of these tasks.
* Software engineers use abstraction to deal with this type of complexity.
* Design patterns provide useful design abstractions for object-oriented systems.

Design Patterns:

A design pattern is a description of interacting objects and classes that interact to solve a general design problem within a particular context.

* A design pattern is an abstract template that can be applied over and over again.
* The idea is apply abstract design patterns in order to solve specific design problems that occur while building real systems.
* NBU: particular context: context is a circumstances under which we solve a particular problem.
* Design patterns provide a way to communicate the parts of a design, i.e., it’s the vernacular software engineers use to talk about designs.

Client-Server Model

* The whole point of a client-server architecture is to distribute the components of an application between the client and the server in some way. E.g., this makes sense if we trying to share a database or files among some users, share printers, etc.
* What gets put where determines the particular type of the client-server architecture.
* In order to build complex web applications, we’re going to make use of numerous design patterns that will help us organize how pieces are placed within the client-server architecture.

N-Tier Architecture

* The n-tier architecture is a highly useful design pattern that maps to the client-server model.
* This design pattern is based on the concept of breaking a system into different pieces or tiers that can be physically separated:
* Each tier is responsible for providing a specific functionality.
* A tier only interacts with the tiers adjacent to it through a well-defined interface.
* Ex.
* Print server – 2-tier architectural pattern.
* Early web applications – 2-tier client-server architecture:
* User interface (browser) functionality resided on the (thin) client.
* Server provided static web pages (HTML).
* Interface between the two via the hypertext transfer protocol (HTTP).