<Source: https://www.youtube.com/watch?v=aSvOThsVe5w&list=PLTCrU9sGyburBw9wNOHebv9SjlE4Elv5a&index=2>

1. Components of a system design

This can be divided into two:

1. Logical entities : databases, Applications layer, communication protocol, requests eg rest, grpc etc., presentation layer i.e. mobile apps, desktop appse,ect NB:(‘some systems won’t have eg loggers)
2. Tangible entities:

Client server architecture

There are three things.

1. Presentation
2. Logic
3. Data

Depending on where your logic+data stay or where the processing is happening then that can determine if the system is thin client or thick client.

2 Tier, 3 Tier and N tier architecture.

In the world of system design, "tier" refers to the logical separation of components within an application or system, often based on their functionality and the way they interact with each other. Each tier represents a layer of the application, with each layer responsible for specific tasks. Let's break down each term:

1. \*\*1 Tier Architecture:\*\* Also known as a single-tier architecture, this model consists of a single software application in which all the components, including the user interface, business logic, and data storage, are housed on a single platform or machine. It's typically used for small-scale applications where simplicity is prioritized over scalability and maintainability. For instance, a simple desktop application that doesn't require interaction with other systems might follow a 1-tier architecture.

2. \*\*2 Tier Architecture:\*\* Also called two-tier architecture, this model divides the application into two main components: the client-side and the server-side. The client-side, which is usually the user interface, is responsible for presenting information to users and accepting their inputs. The server-side, on the other hand, handles business logic and data storage. This architecture is commonly seen in client-server applications where the client communicates directly with the server to perform tasks, such as accessing a database. Examples include desktop applications interacting with a database server.

3. \*\*3 Tier Architecture:\*\* Also known as three-tier architecture, this model extends the 2-tier architecture by introducing a middle layer between the client-side and the server-side. The three tiers are:

- \*\*Presentation Tier (Client-side):\*\* This tier handles the user interface and user interactions. It's responsible for displaying information to users and capturing their inputs.

- \*\*Application Tier (Middle Tier):\*\* Also called the business logic tier or logic tier, this layer contains the application logic responsible for processing requests from the presentation tier, applying business rules, and coordinating data access. It acts as an intermediary between the presentation tier and the data tier.

- \*\*Data Tier (Server-side):\*\* This tier, often referred to as the data storage or data access tier, is responsible for managing and storing data. It typically involves database systems where data is stored and accessed by the application logic tier.

4. \*\*N Tier Architecture:\*\* N-tier architecture extends the concept of 3-tier architecture by adding more tiers/layers as needed to handle additional complexity or scalability requirements. It allows for greater flexibility, scalability, and separation of concerns. Each additional tier can be dedicated to specific functions such as security, caching, messaging, etc. N-tier architectures are common in large-scale enterprise applications where there's a need to distribute functionality across multiple layers to achieve better performance, scalability, and maintainability.

1. What is a proxy
2. What is the significance of proxies in system design on the client server architecture

In client server architecture there exists two types of proxies:

1. Forward proxy
2. Rever proxy

* Merits and dimerits of reverse proxy and forward proxy
* Scenarios where you can make good use of proxies