Caching – System Design Concept

Last Updated : 12 Feb, 2024

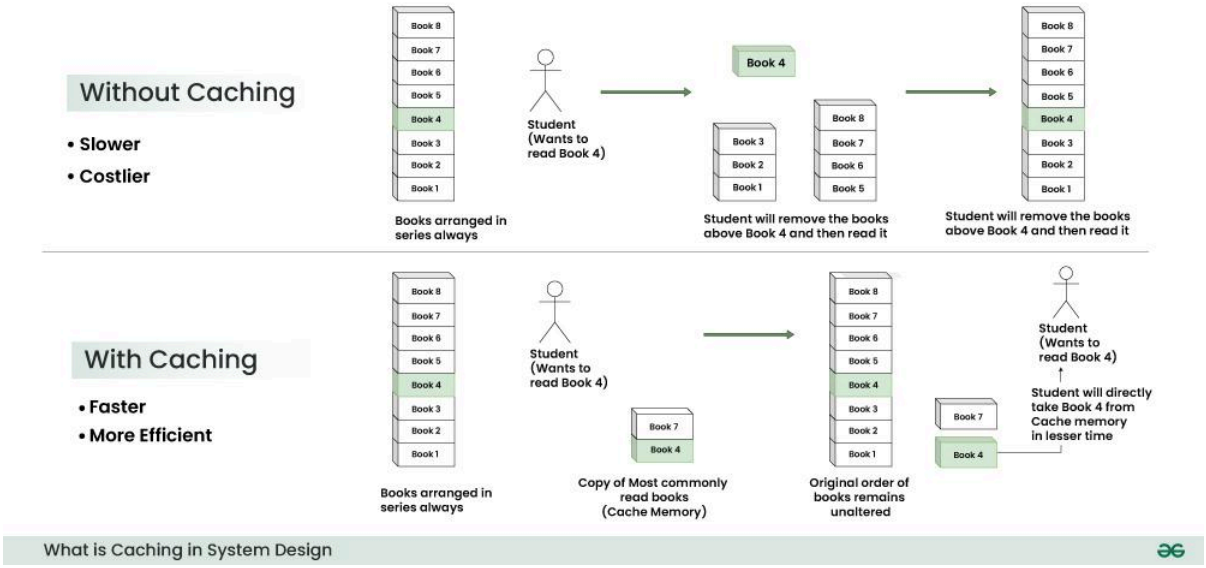
Caching is a system design concept that involves storing frequently accessed data in a location that is easily and quickly accessible. The purpose of caching is to improve the performance and efficiency of a system by reducing the amount of time it takes to access frequently accessed data.

Important Topics for Caching in System Design

- What is Caching
- How Does Cache Work?
- Where Cache can be added?
- key points to understand Caching
- Types of Cache
- Applications of Caching

- Eviction Policies of Caching

1. What is Caching



Imagine a library where books are stored on shelves. Retrieving a book from a shelf takes time, so a librarian decides to keep a small table near the entrance. This table is like a cache, where the librarian places the most popular or recently borrowed books.

Now, when someone asks for a frequently requested book, the librarian checks the table first. If the book is there, it's quickly provided. This saves time compared to going to the shelves each time. The table acts as a cache, making popular books easily accessible.

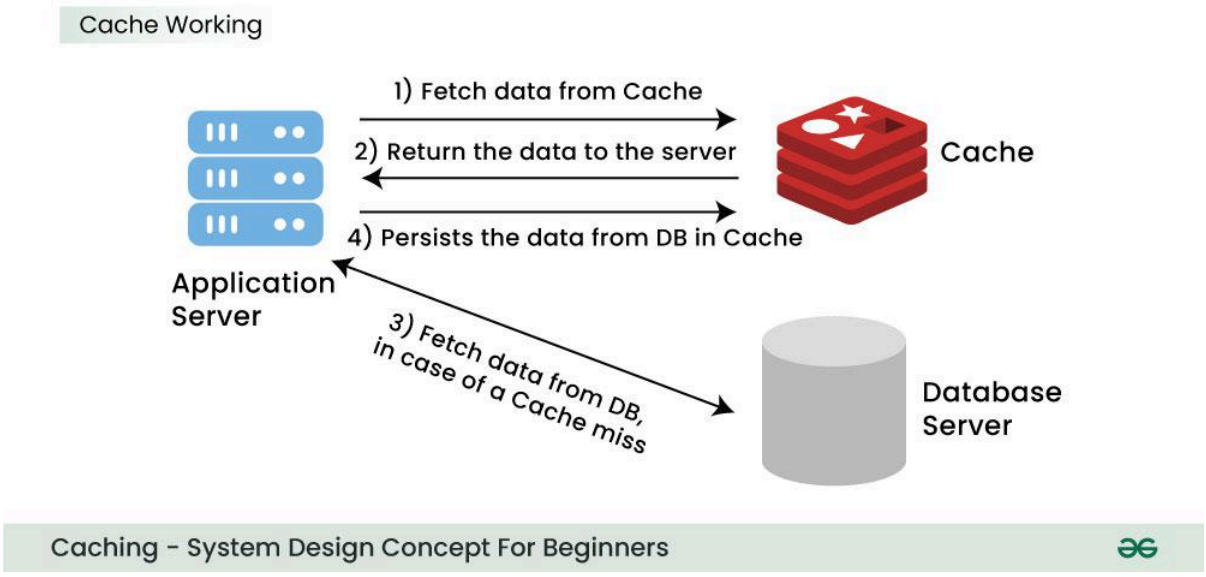
- The same things happen in the system. In a system accessing data from primary memory (RAM) is faster than accessing data from secondary memory (disk).
- Caching acts as the local store for the data and retrieving the data from this local or temporary storage is easier and faster than retrieving it from the database.

- So If you need to rely on a certain piece of data often then cache the data and retrieve it faster from the memory rather than the disk.

As you know there are many benefits of the cache but that doesn't mean we will store all the information in your cache memory for faster access, we can't do this for multiple reasons, such as:

- Hardware of the cache which is much more expensive than a normal database.
- Also, the search time will increase if you store tons of data in your cache.
- So in short a cache needs to have the most relevant information according to the request which is going to come in the future.

2. How Does Cache Work?



Typically, web application stores data in a database. When a client requests some data, it is fetched from the database and then it is returned to the user. Reading data from the database needs network calls and I/O operation which is a time-consuming process. Cache reduces the network call to the database and speeds up the performance

Twitter: when a tweet becomes viral, a huge number of clients request the same tweet. Twitter is a gigantic website that has millions of users. It is inefficient to read data from the disks for this large volume of user requests.

Here is how using cache helps to resolve this problem:

- To reduce the number of calls to the database, we can use cache and the tweets can be provided much faster.
- In a typical web application, we can add an application server cache, and an **in-memory store** like Redis alongside our application server.
- When the first time a request is made a call will have to be made to the database to process the query. This is known as a **cache miss**.
- Before giving back the result to the user, the result will be saved in the cache.
- When the second time a user makes the same request, the application will check your cache first to see if the result for that request is cached or not.
- If it is then the result will be returned from the in-memory store. This is known as a **cache hit.**
- The response time for the second time request will be a lot less than the first time.

3. Where Cache Can be Added?

Caching is used in almost every layer of computing.

- In hardware, for example, you have various layers of cache memory.
- You have layer 1 cache memory which is the CPU cache memory, then you have layer 2 cache memory and finally, you would have the regular RAM (random access memory)

- You also have caching in a web browser to decrease the load time of the website.

4. Key points to understand Caching

Caching can be used in a variety of different systems, including web applications, databases, and operating systems. In each case, caching works by storing data that is frequently accessed in a location that is closer to the user or application. This can include storing data in memory or on a local hard drive.

- **How it works:**
    - When data is requested, the system first checks if the data is stored in the cache.
    - If it is, the system retrieves the data from the cache rather than from the original source.
    - This can significantly reduce the time it takes to access the data.
- **Types of caching:**
    - There are several types of caching, including in-memory caching, disk caching, and distributed caching.
    - In-memory caching stores data in memory, while disk caching stores data on a local hard drive.
    - Distributed caching involves storing data across multiple systems to improve availability and performance.
- **Cache eviction:**
    - Caches can become full over time, which can cause performance issues.
    - To prevent this, caches are typically designed to automatically evict older or less frequently accessed data to make room for new data.

- Caching can introduce issues with data consistency, particularly in systems where multiple users or applications are accessing the same data.
- To prevent this, systems may use cache invalidation techniques or implement a cache consistency protocol to ensure that data remains consistent across all users and applications.

5. Types of Cache

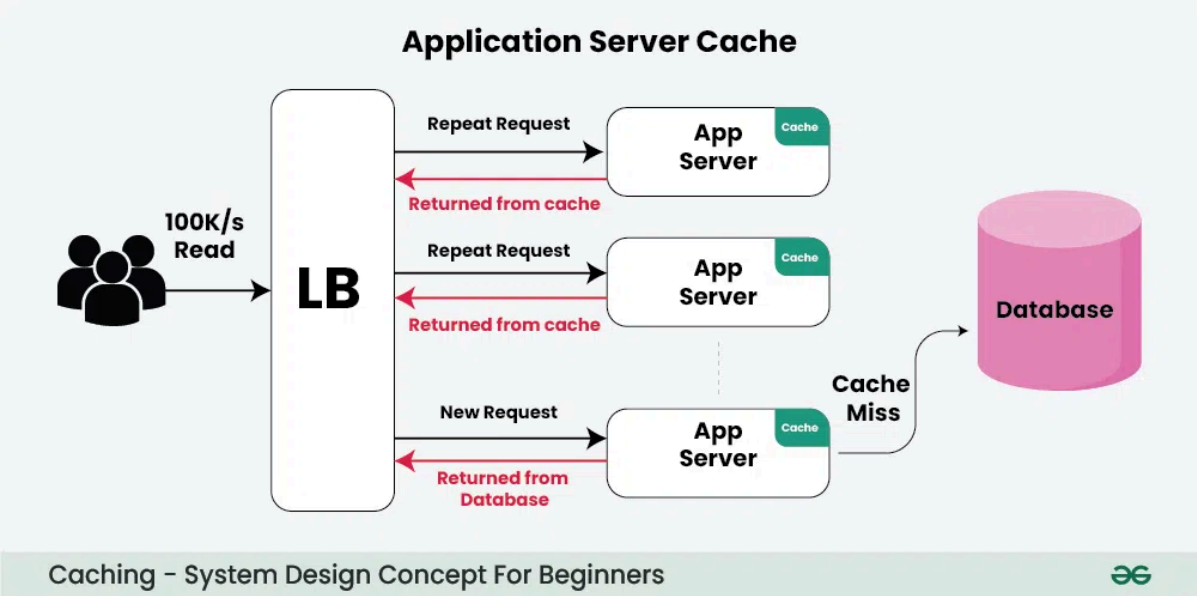
In common there are four types of Cache...

5.1. Application Server Cache:

In the “How does Cache work?” section we discussed how application server cache can be added to a web application.

- A cache can be added in in-memory alongside the application server.
- The user’s request will be stored in this cache and whenever the same request comes again, it will be returned from the cache.
- For a **new request**, data will be fetched from the disk and then it will be returned.
- Once the new request will be returned from the disk, it will be stored in the same cache for the next time request from the user.

Note: When you place your cache in memory ,the amount of memory in the server is going to be used up by the cache. If the number of results you are working with is really small then you can keep the cache in memory.



Drawbacks of Application Server Cache:

- The problem arises when you need to scale your system. You add multiple servers in your web application (because one node can not handle a large volume of requests) and you have a load balancer that sends requests to any node.
- In this scenario, you'll end up with a lot of cache misses because each node will be unaware of the already cached request.
- This is not great and to overcome this problem we have two choices: Distribute Cache and Global Cache. Let's discuss that...

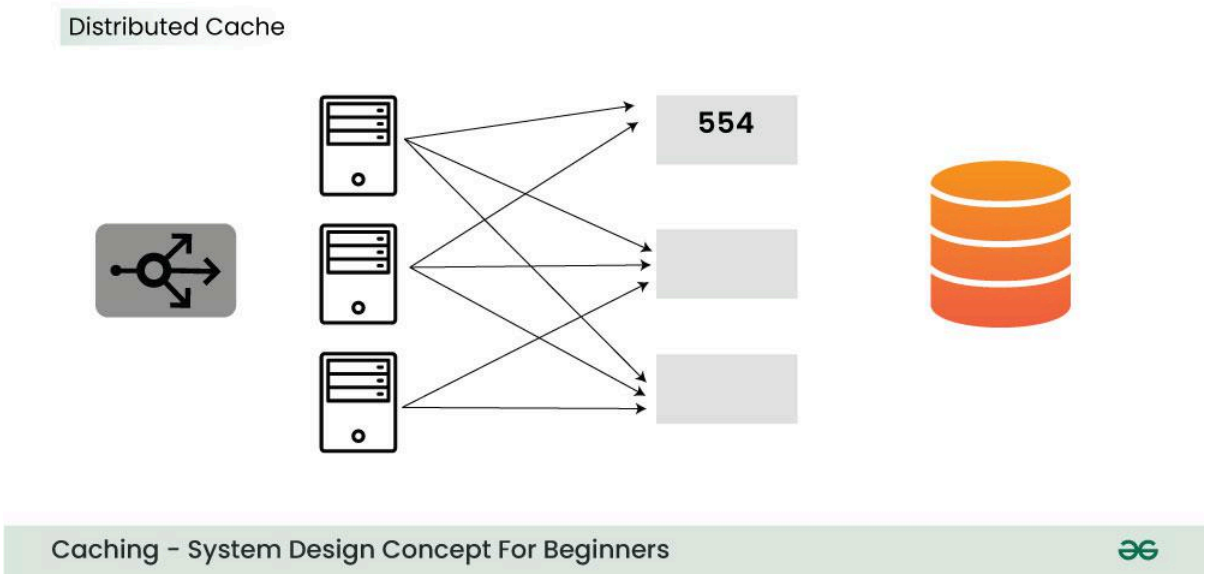
5.2. Distributed Cache:

In the distributed cache, each node will have a part of the whole cache space, and then using the consistent hashing function each request can be routed to where the cache request could be found. Let's suppose we have 10 nodes in a distributed system, and we are using a load balancer to route the request then...

- Each of its nodes will have a small part of the cached data.
- To identify which node has which request the cache is divided up using a consistent hashing function each request can be routed to

looking for a certain piece of data, it can quickly know where to look within the distributed cache to check if the data is available.

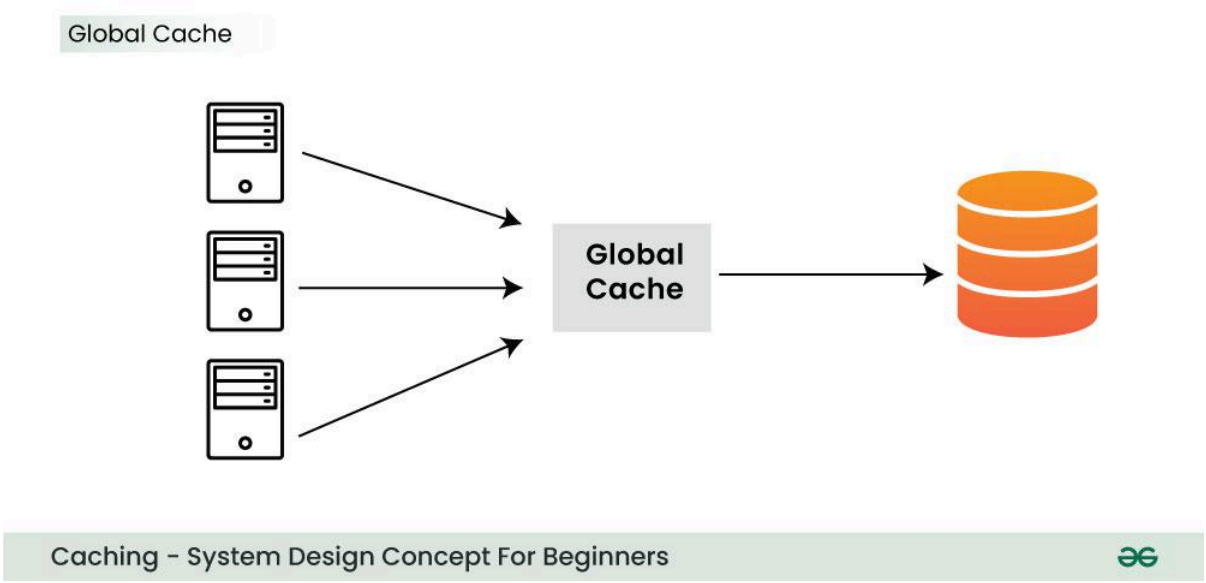
- We can easily increase the cache memory by simply adding the new node to the request pool.



5.3. Global Cache:

As the name suggests, you will have a single cache space and all the nodes use this single space. Every request will go to this single cache space. There are two kinds of the global cache

- First, when a cache request is not found in the global cache, it's the responsibility of the cache to find out the missing piece of data from anywhere underlying the store (database, disk, etc).
- Second, if the request comes and the cache doesn't find the data then the requesting node will directly communicate with the DB or the server to fetch the requested data.

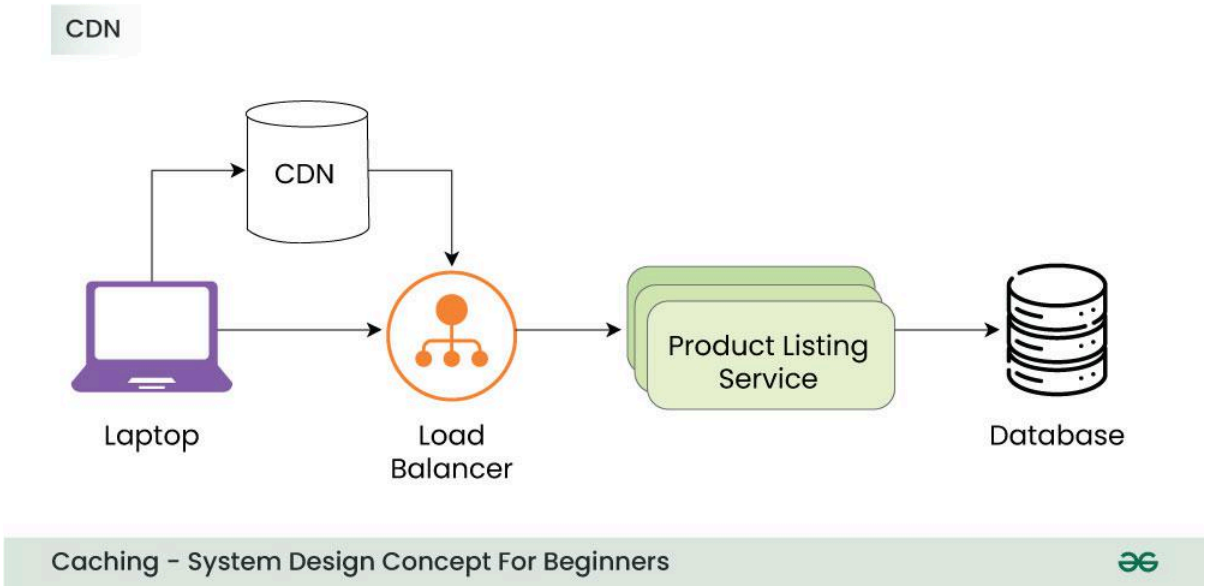


5.4. CDN (Content Distribution Network)

A CDN is essentially a group of servers that are strategically placed across the globe with the purpose of accelerating the delivery of web content. A CDN-

- Manages servers that are geographically distributed over different locations.
- Stores the web content in its servers.
- Attempts to direct each user to a server that is part of the CDN so as to deliver content quickly.

CDN is used where a large amount of static content is served by the website. This can be an HTML file, CSS file, JavaScript file, pictures, videos, etc. First, request ask the CDN for data, if it exists then the data will be returned. If not, the CDN will query the backend servers and then cache it locally.



6. Applications of Caching

Facebook, Instagram, Amazon, Flipkart....these applications are the favorite applications for a lot of people and most probably these are the most frequently visited websites on your list.

> *Have you ever noticed that these websites take less time to load than brand-new websites? And have you noticed ever that on a slow internet connection when you browse a website, texts are loaded before any high-quality image? Why does this happen?*

The answer is Caching.

- If you check your Instagram page on a slow internet connection you will notice that the images keep loading but the text is displayed. For any kind of business, these things matter a lot.
- A better customer/user experience is the most important thing and you may lose a lot of customers due to the poor user experience with your website.
- A user immediately switches to another website if they find that the

You can take the example of watching your favorite series on any video streaming application. How would you feel if the video keeps buffering all the time? Chances are higher that you won't stick to that service and you discontinue the subscription. All the above problems can be solved by improving retention and engagement on your website and by delivering the best user experience. And one of the best solutions is Caching.

7. What are the Advantages of using Caching?

Caching optimizes resource usage, reduces server loads, and enhances overall scalability, making it a valuable tool in software development.

- **Improved performance:** Caching can significantly reduce the time it takes to access frequently accessed data, which can improve system performance and responsiveness.
- **Reduced load on the original source:** By reducing the amount of data that needs to be accessed from the original source, caching can reduce the load on the source and improve its scalability and reliability.
- **Cost savings:** Caching can reduce the need for expensive hardware or infrastructure upgrades by improving the efficiency of existing resources.

8. What are the Disadvantages of using Caching?

Despite its advantages, caching comes with drawbacks also and some of them are:

- **Data inconsistency:** If cache consistency is not maintained properly, caching can introduce issues with data consistency.
- **Cache eviction issues:** If cache eviction policies are not designed properly, caching can result in performance issues or data loss.
- **Additional complexity:** Caching can add additional complexity to a system, which can make it more difficult to design, implement, and

Overall, caching is a powerful system design concept that can significantly improve the performance and efficiency of a system. By understanding the key principles of caching and the potential advantages and disadvantages, developers can make informed decisions about when and how to use caching in their systems.

9. Cache Invalidation Strategies

Cache invalidation is crucial in systems that use caching to enhance performance. When data is cached, it's stored temporarily for quicker access. However, if the original data changes, the cached version becomes outdated. Cache invalidation mechanisms ensure that outdated entries are refreshed or removed, guaranteeing that users receive up-to-date information.

- Common strategies include time-based expiration, where cached data is discarded after a certain time, and event-driven invalidation, triggered by changes to the underlying data.
- Proper cache invalidation optimizes performance and avoids serving users with obsolete or inaccurate content from the cache.

10. Eviction Policies of Caching

Eviction policies are crucial in caching systems to manage limited cache space efficiently. When the cache is full and a new item needs to be stored, an eviction policy determines which existing item to remove.

- One common approach is the Least Recently Used (LRU) policy, which discards the least recently accessed item. This assumes that recently used items are more likely to be used again soon.
- Another method is the Least Frequently Used (LFU) policy, removing the least frequently accessed items.
- Alternatively, there's the First-In-First-Out (FIFO) policy, evicting the oldest cached item.

Each policy has its trade-offs in terms of computational complexity and adaptability to access patterns. Choosing the right eviction policy depends on the specific requirements and usage patterns of the application, balancing the need for efficient cache utilization with the goal of minimizing cache misses and improving overall performance.

11. Conclusion

- Caching is becoming more common nowadays because it helps make things faster and saves resources.
- The internet is witnessing an exponential growth in content, including web pages, images, videos, and more.
- Caching helps reduce the load on servers by storing frequently accessed content closer to the users, leading to faster load times.
- Real-time applications, such as online gaming, video streaming, and collaborative tools, demand low-latency interactions.
- Caching helps in delivering content quickly by storing and serving frequently accessed data without the need to fetch it from the original source every time.

What is Content Delivery Network(CDN) in System Design

Message Queues | System Design

Similar Reads

Server-side Caching and Client-side Caching

Caching is a temporary technique that stores frequently accessed data for faster retrieval. There are two main types of caching in web development:...

8 min read

Negative Caching - System Design

Negative caching refers to storing failed results or errors to avoid redundant requests. It plays a major role in enhancing system performanc...

11 min read

Caching Design Pattern

In today's digital world, speed and efficiency matter a lot. When we use apps and websites, we want things to happen quickly. But making...

10 min read

Asynchronous Caching Mechanisms to Overcome Cache Stampede…

Caching is a critical component in modern software systems, serving as an effective means to reduce latency and improve system performance....

4 min read

What is Pre-Caching?

Pre-caching is like getting ready for something before it happens. Imagine you're going on a trip and you pack your bag the night before so you're all...

13 min read

10 min read

Why Caching Does not Always Improve Performance?

Caching is widely regarded as a key technique for enhancing the performance of systems by reducing data retrieval times and alleviating t...

8 min read

Design a system that counts the number of clicks on YouTube videos |...

Designing a Click Tracking System for YouTube Videos involves architecting a comprehensive and efficient solution to monitor and analyze user...

15+ min read

Design Restaurant Management System | System Design

In the modern restaurant industry, delivering exceptional dining experiences requires more than just good cuisine. Restaurant Managemen...

15 min read

Design a Picture-Sharing System - System Design

In the present day, the need for good tools to exchange and organize images has never been this much higher. As for social networking, e-...

11 min read

Article Tags : GBlog System Design System-Design