**System Design:**

DB First vs API First:

<https://softwareengineering.stackexchange.com/questions/264379/code-first-vs-database-first/264389#264389?s=21ca9403dc494760bcc06ecb5a16980d>

<https://www.ryadel.com/en/code-first-model-first-database-first-vs-comparison-orm-asp-net-core-entity-framework-ef-data/>

<https://dzone.com/articles/an-api-first-development-approach-1>

[API](https://dzone.com/articles/versioning-restful-services-with-spring-boot) [Versioning](http://www.springboottutorial.com/spring-boot-versioning-for-rest-services) – [Should](https://dzone.com/articles/versioning-rest-api-with-spring-boot-and-swagger) be avoided as much as possible. There are various of doing API versioning – URI Versioning, Header Versioning, Request Parameter Versioning, Media Type Versioning

Top 10 Design Questions – [Link1](https://hackernoon.com/top-10-system-design-interview-questions-for-software-engineers-8561290f0444) [Link2](https://www.hackingnote.com/en/interview/system-design-interview-questions)

Design Cheat Sheet – [Link2](https://gist.github.com/vasanthk/485d1c25737e8e72759f)

Consistent Hashing – [Link1](https://www.youtube.com/watch?v=viaNG1zyx1g) [Link2](https://www.youtube.com/watch?v=bBK_So1u9ew)

All About RabbitMQ - [here](https://www.cloudamqp.com/blog/2017-12-29-part1-rabbitmq-best-practice.html) [here](https://www.rabbitmq.com/confirms.html)

**DB Related Theory:**

**SQL** – ideal for Transactional Storage (booking, payment etc.), **NoSQL** – Non-Transactional Storage (comments, reviews, history data etc.)

Surrogate key vs Natural business key - [here](https://stackoverflow.com/questions/2186260/when-to-use-an-auto-incremented-primary-key-and-when-not-to) [here](https://stackoverflow.com/questions/63090/surrogate-vs-natural-business-keys) [here](https://stackoverflow.com/questions/707657/picking-the-best-primary-key-numbering-system) (very nice)

If you are using surrogate keys then don’t forget to key create unique constraint on the natural business key - [here](https://stackoverflow.com/questions/337503/whats-the-best-practice-for-primary-keys-in-tables)

Usually the rule of thumb is that never use meaningful information in primary keys (like Social Security number or barcode). Just plain auto incremented integer. However constant the data seems - it may change at one point (new legislation comes and all SSNs are recalculated). Integer comparisons are much cheaper than string comparisons, and it will occupy less space in the DB- [here](https://stackoverflow.com/questions/506164/use-item-specific-prefixes-and-autonumber-for-primary-keys)

**Common Mistakes on DB Design** - [here](https://stackoverflow.com/questions/621884/database-development-mistakes-made-by-application-developers?page=1&tab=votes#tab-top)

Load Balancers – Software and Hardware

Layer 4 vs Layer 7 load balancing

JVM Profilers - [here](https://www.baeldung.com/java-profilers), [Uber JVM Profiler](https://github.com/uber-common/jvm-profiler/tree/master/src/main/java/com/uber/profiling/profilers)

Courses to do:

Apache Kafka course (Udemy)

Useful Links:

<http://highscalability.com/>

Approach:

1. Ask for Requirements
2. State assumptions if any and confirm from them
3. Think out loud