System Design

**Short timeline** - Aim for breadth with system design topics. Practice by solving some interview questions.

**Medium timeline** - Aim for breadth and some depth with system design topics. Practice by solving many interview questions.

**Long timeline** - Aim for breadth and more depth with system design topics. Practice by solving most interview questions.

Everything is a trade-off.

**Step 1: Outline use cases, constraints, and assumptions**

Gather requirements and scope the problem. Ask questions to clarify use cases and constraints. Discuss assumptions :

eg: Who is going to use it? How are they going to use it? How many users are there? What does the system do? What are the inputs and outputs of the system? How much data do we expect to handle? How many requests per second do we expect? What is the expected read to write ratio?

**Step 2: Create a high level design**

Outline a high level design with all important components.

* Sketch the main components and connections.
* Justify your ideas

**Step 3: Design core components**

Dive into details for each core component.

For example, if you were asked to design a url shortening service, discuss:

* + - Generating and storing a hash of the full url
      * MD5 and Base62
      * Hash collisions
      * SQL or NoSQL
      * Database schema
    - Translating a hashed url to the full url
      * Database lookup
    - API and object-oriented design

**Step 4: Scale the design**

Identify and address bottlenecks, given the constraints.

For example, do you need the following to address scalability issues?

* + - Load balancer
    - Horizontal scaling
    - Caching
    - Database sharding

**Step 5: Discuss potential solutions and trade-offs**.

Everything is a trade-off. Address bottlenecks using principles of scalable system design.

**Step 6: Provide some estimate calculation**

An estimate is a numerical representation of how the designed system will perform as compared to another design.

Refer to following for providing an estimate of the system design:

* [Use back of the envelope calculations](http://highscalability.com/blog/2011/1/26/google-pro-tip-use-back-of-the-envelope-calculations-to-choo.html)
* [Powers of two table](https://github.com/donnemartin/system-design-primer#powers-of-two-table)
* [Latency numbers every programmer should know](https://github.com/donnemartin/system-design-primer#latency-numbers-every-programmer-should-know)

Scalability