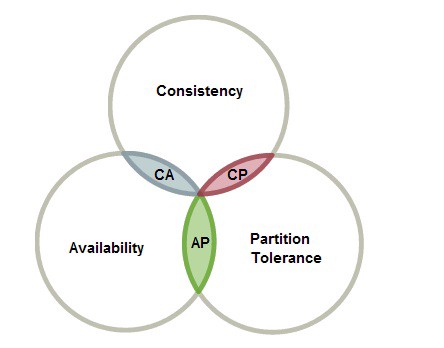
**What is CAP theorem ?**



<https://towardsdatascience.com/cap-theorem-and-distributed-database-management-systems-5c2be977950e>

**Ticket booking ER diagram**

[**https://creately.com/diagram/example/hs5vby2t/online%20movie%20ticket%20booking%20System**](https://creately.com/diagram/example/hs5vby2t/online%20movie%20ticket%20booking%20System)

**Which data structure will you use to implement a Limit Order Book? and Why?**

https://javarevisited.blogspot.com/2017/03/2-practical-data-structure-algorithm-interview-questions-java.html  
**Which data structure will you choose to store Market data? and Why?**

https://javarevisited.blogspot.com/2017/03/2-practical-data-structure-algorithm-interview-questions-java.html

**Upload and download a huge file more than 5GB**

1. The client sends a request to initiate a multipart upload, the API responds with an upload id
2. The client uploads each file chunk with a part number (to maintain ordering of the file), the size of the part, the md5 hash of the part and the upload id; each of these requests is a separate HTTP request. The API validates the chunk by checking the md5 hash received chunk against the md5 hash the client supplied and the size of the chunk matches the size the client supplied. The API responds with a tag (unique id) for the chunk. If you deploy your API across multiple locations you will need to consider how to store the chunks and later access them in a way that is location transparent.
3. The client issues a request to complete the upload which contains a list of each chunk number and the associated chunk tag (unique id) received from API. The API validates there are no missing chunks and that the chunk numbers match the correct chunk tag and then assembles the file or returns an error response.

Amazon also supplies methods to abort the upload and list the chunks associated with the upload. You may also want to consider a timeout for the upload request in which the chunks are destroyed if the upload is not completed within a certain amount of time.

**How to limit number of requests in java rest api ?**

**Process 1 :**

There is [bucket4j-spring-boot-starter](https://github.com/MarcGiffing/bucket4j-spring-boot-starter) project which uses [bucket4j](https://github.com/vladimir-bukhtoyarov/bucket4j) library with token-bucket algorithm to rate-limit access to the REST api.

As an example simple setup which allows a maximum of 5 requests within 10 seconds independently from the user:

bucket4j:

enabled: true

filters:

- cache-name: buckets

url: .\*

rate-limits:

- bandwidths:

- capacity: 5

time: 10

unit: seconds

**Process 2 : Using webfilter**

public class LimitFilter implements Filter {

private int limit = 5;

private int count;

private Object lock = new Object();

public void doFilter(ServletRequest request, ServletResponse response,

FilterChain chain) throws IOException, ServletException {

try {

boolean ok;

synchronized (lock) {

ok = count++ < limit;

}

if (ok) {

// let the request through and process as usual

chain.doFilter(request, response);

} else {

// handle limit case, e.g. return status code 429 (Too Many Requests)

// see http://tools.ietf.org/html/rfc6585#page-3

}

} finally {

synchronized (lock) {

count--;

}

}

}

}

**Process 3 : Using API gateway**

Create an api gateway which can limit api rate like aws api gateway, Netflix Zul etc.

Different alogorithms to implement Api rate limiter

<https://www.youtube.com/watch?v=mhUQe4BKZXs>

**Design shortning url**

[**Tech**](https://www.educative.io/courses/grokking-the-system-design-interview/m2ygV4E81AR) **dummies**

**Design a pastebin**

[**https://www.youtube.com/watch?v=j1CZ5vSU3CQ**](https://www.youtube.com/watch?v=j1CZ5vSU3CQ)

**Design book my show**

[**https://www.youtube.com/watch?v=lBAwJgoO3Ek&t=1275s**](https://www.youtube.com/watch?v=lBAwJgoO3Ek&t=1275s)

**Other designs**

Custom thread pool

Custom LRU cache

External sort to sort a huge file

Vending machine

Atm machine

Traffic signal

Elevetor design