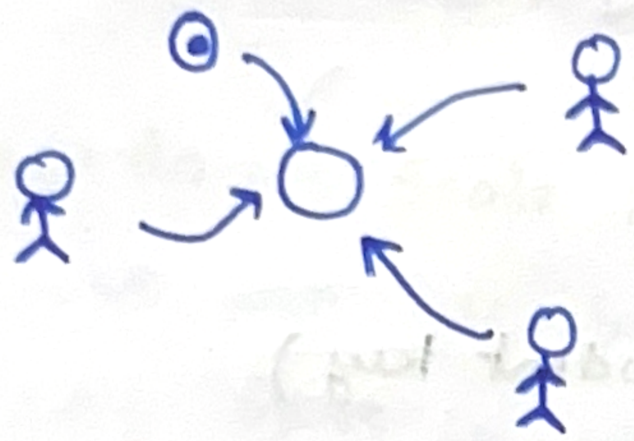
Idempodency Handler :

Concurrency	Idempodency
one resource is being accessed by multiple users	helps to take care of duplicate request

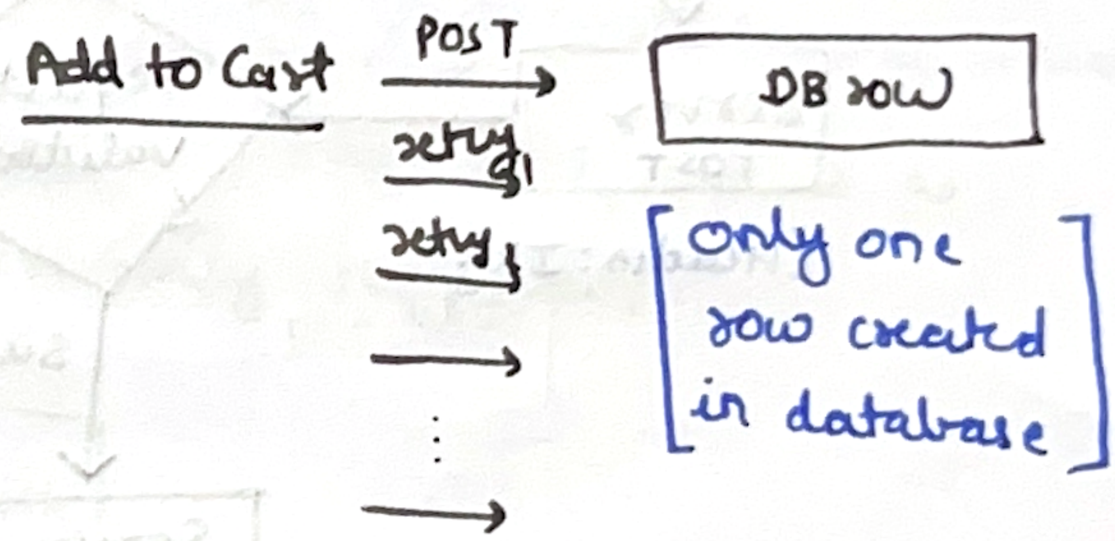


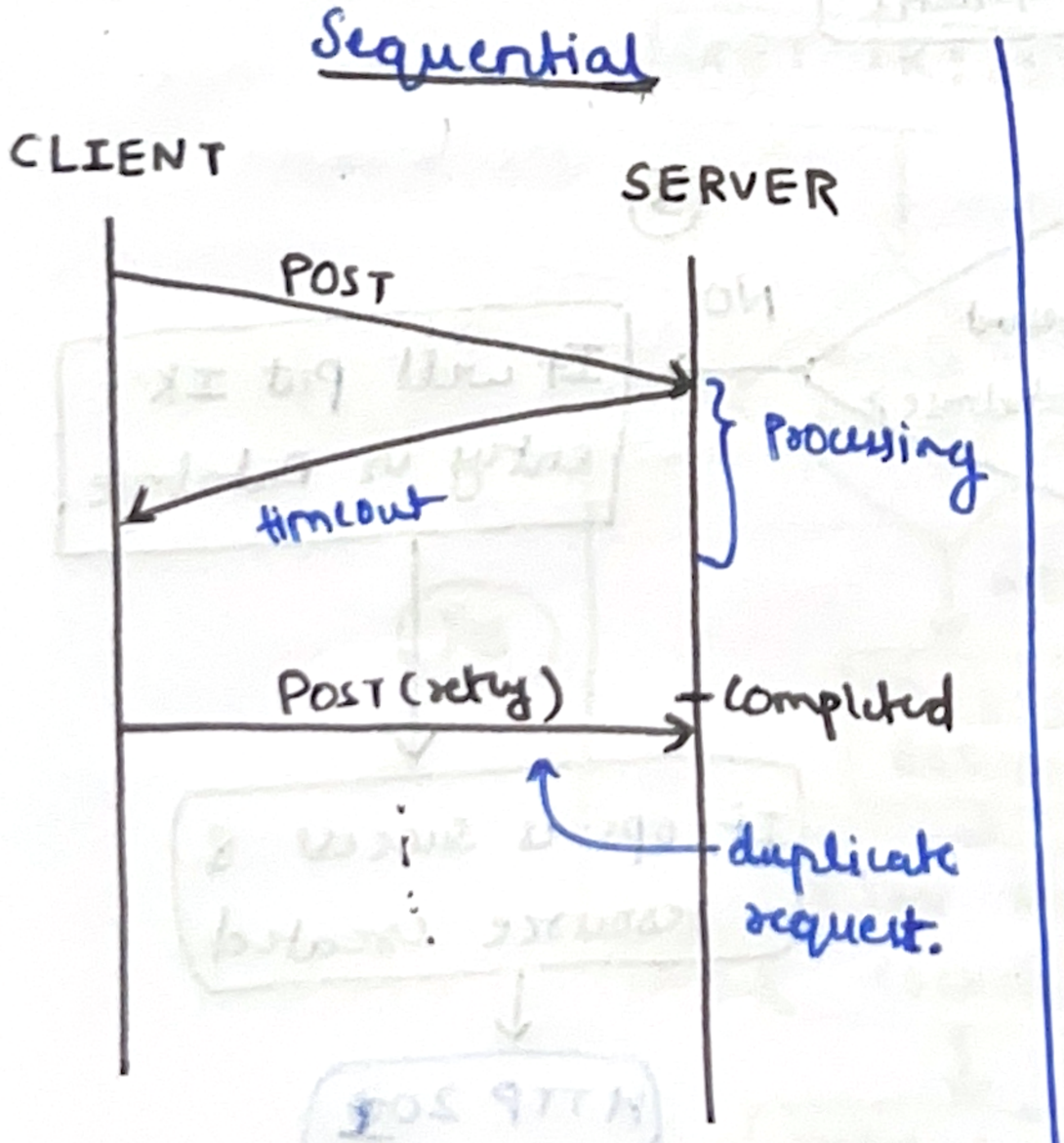
It enables client to safely (retry) on operation without worrying about side-effect of operation can cause

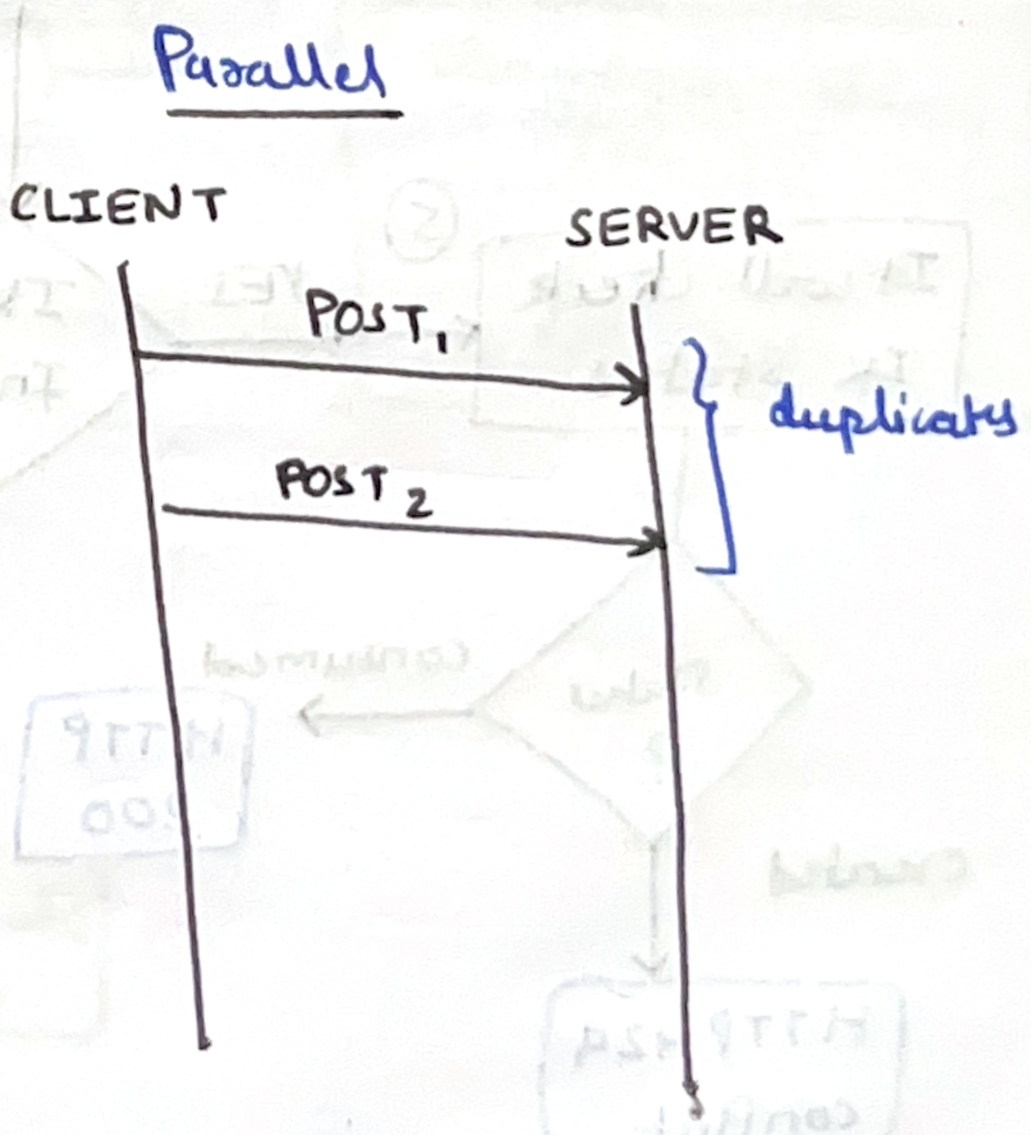
① by default get, put and delete are idempodent in nature.

② Post needs to be made idempodent explicitly

operation :





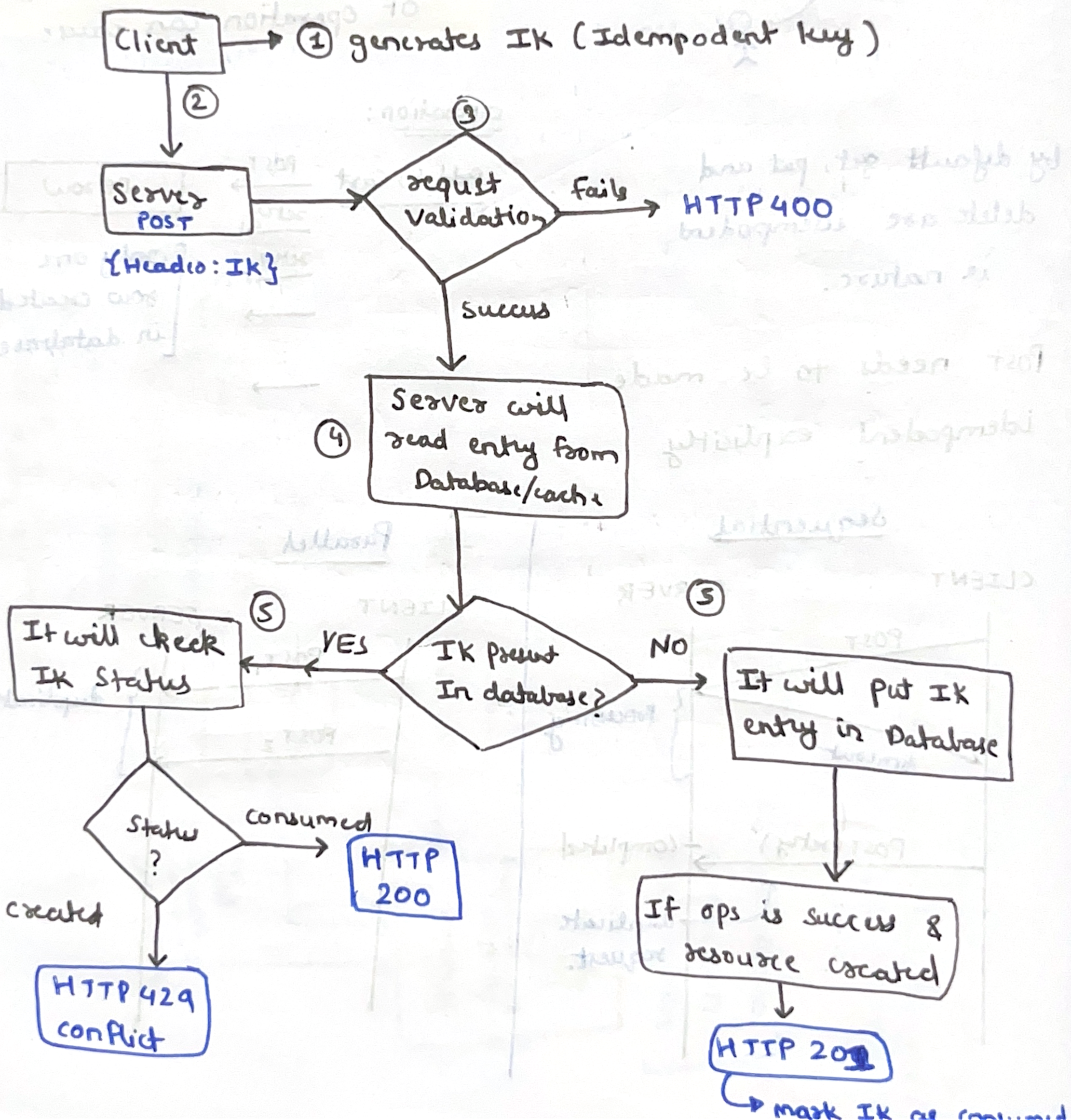


✳ Approach :

① Agreement with Client :

i) Client should generate the Idempodent key

ii) for each different operation, new idempodent key should be generated.



* Above design deals with Sequential request efficiently.
* To deal with Parallel duplicate requests; we need to have a lock on database operation using mutex
But NOTE that Mutex lock would not be per database but per idempodent key ⇒ this would lead to more scalable server and more efficient one.

How do we Scale IH in case of multiple servers:
we can use something like redis cache as Idempodent Store which can scale as per needs, or for simpler use case we can use database Uniqueness Atomic insert.
⇒ Redis distributed locking is extremly fast, scales as per needs, locking is natively supported.