8 weeks <u>System Design</u>

week 01 (20 Nov, 2025)

Agenda:

- Design online/offline indicators
- How to approach System Design
- Designing a multi user bloging plateform
  - Database
  - Cashing

Online Offline Indicator

user 1	online
user 2	online
user 3	offline
user 4	offline

Approaching System Design: (Big Orginazation)

Decide the core &
building your system around it.

Core is user-case specific

<u>Database</u>, <u>comunication</u>

↓ Design DB ↓ websockets

→ where we know who will be do So go this like know about DB, sockets.

Incremental building

1. Start with a Day zero <u>architechure</u>
   ↳ simple
2. See how each component would behave
   ↳ under Load
   ↳ at scale
3. Identify the bottenneck
4. re-architect
5. repeat

Point to remember:

- Understanding the <u>core property</u> & access pattern
  ↳ pick that <u>technology</u> that are required over system
  ↓ DB Framework etc
- Affinity towards a tech comes seconds
- Build an intuition towards building systems

When we [illegible]

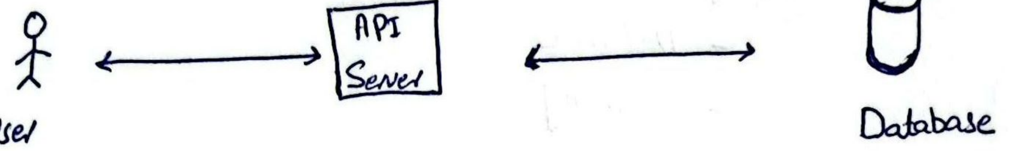
Storage:

$$\begin{tikzcd} \text{user} \arrow[r] \arrow[d] & \text{online/offline} \arrow[d] \\ \text{int} & \text{bool} \end{tikzcd}$$

Access: key values

[not yet decided DB]
we still have not
picked which DB to use
only decide which
type of data we want
to store

Interfacing API

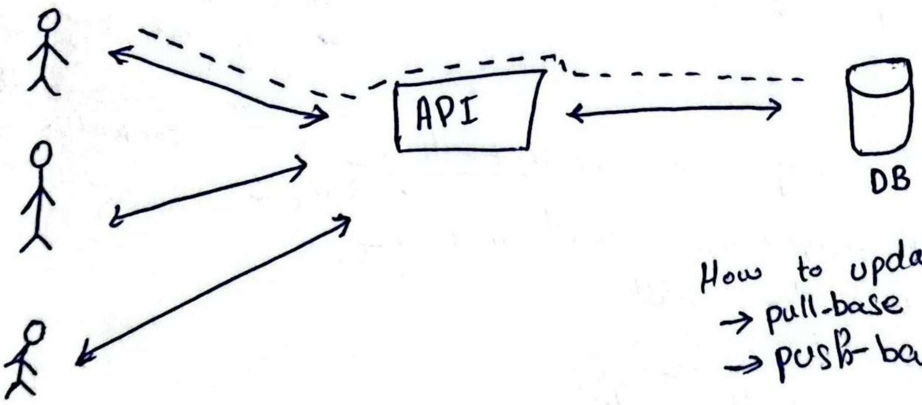


GET /status/users ? ids = $\underbrace{U_1, U_2, U_3}$

User ids from
whom status to be
fixed

[why we exposed All
users because if we want
Show in UI to status Not
call App for each user.]

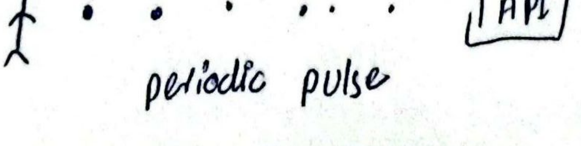
Updating the database



How to update data?
→ pull-base model
→ push-base model

Push based model: User push their status periodically
Our API servers cannot "pull" from client because we cannot
proactively talk to client * unless there is a persistent connection.
Every user periodically send "heartbeat' to the service

post /heartbeat → the authenticated user will be marked
as 'alive'



[why we need send periodically
send because it tell server
I'm alive if it not send
server mark offline.]

Offline : when we do not recived heartbeat long time enough
↳ Handle through bussiness logic.

Say 30sec

In database store " time you received the last heartbeat"

pulse

user_id	last_hb
U1	1000
U2	1050
U3	1060

when you recived the heartbeat

```
Update pulse
Set last_hb = Now()
Where user_id = U1
```

→ its not decide using SQL its easy way to understandable

* User sends heartbeat every 10 Seconds.

→ Now our GET Method change:

GET /status/<user_id>

→ If no entry in the database for user → offline

→ if entry and entry. last_hb < $\underset{\text{current time}}{\underline{\text{Now() - 30 sec}}}$ → offline

online

let's estimate the scale

B → Byte

100 users → 100 entries
1000 users → 1000 entries
1B users → 1B entries

Each entry has 2 columns
user_id → int(4B) , last_hb → int(4B)

Total storage required for 1B entries = 8GB

Size of each entry = 8B

Can we do better on Storage? [always think what i do possible way to better]

Requirement :
user 1 → offline
user 2 → online
user 3 → online
} we all is online/offline

what if absence == offline?

Idea: if user not present in the DB. we return offline.
So, lets expire (→ delete) the entries after 30 seconds

→ we can do for every system step

[→ every system step we make ask question can we change this can I go alternative]

if we delete entries → we save a bunch of space by not storing data of inactive users

Total entries = Active users
If 1B total users and 100k active then total entries = 100k
total size = 800KB

How to auto delete?

Approach 01: Write a CRON job that deletes expired entries $\rightarrow$ Implement
- Not a robust solution
- We need to handle edge case in the business logic

Approach 02: Can we not offload this to out datastore?

[A corn job is an auto scheduled task that run at specific times (daily, hourly, weekly etc) using the system's corn scheduler]

$\boxed{\text{Never re-invent the wheel!}}$

DB with KV + expiration → Redis
↳ key-value → DynamoDB

Upon receiving an heartbeat
- update entry in redis/DynamoDB with ttl = 30 sec
↳ Time to Live

Every heartbeat move the expiration time forward!

Which one would you pick & Why?

Redis - key-Value Store with Expiration
SET user_id "123" EX 3600

→ Database taking care of deleting & Storage

Redis	DynamoDB
	✓ —— persistance
	✓ —— Managed by AWS — why we need?
	✓ —— (team size & expertize)
—— ✓ Vender Locking (cost/issue) ↓ why vender Locking?	
→ Redis persistance ↓ How they give us?	[walmart not on AWS because it used there data Amzon. Not used completor of Cloud used]
→ we can change one provider to another.	
✓ time sensitivity	✗ Stateless

→ GCP → Google cloud platform
→ Vender locking means you depende become on one cloud provide, and it become difficult or expensive to move another provider.
e.g AWS DynamoDB

Note: In real world websocket are used in such.

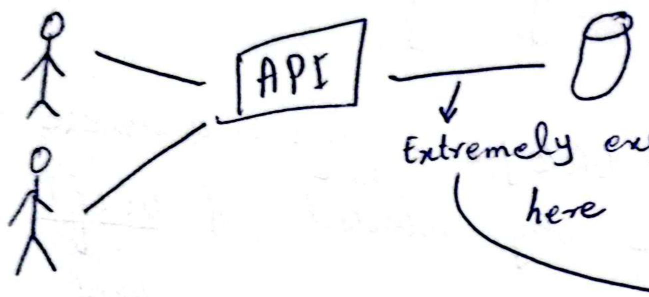
How is our DB doing?

Heartbeat is send every 10sec

So, one user in 1 min sends 6 heartbeats

If there are 1m active user, our system will get 6m req/min

Each heartbeat request result in DB call



Our DB needs to handle 6m updates per minute!!

↓ Extremely expensive

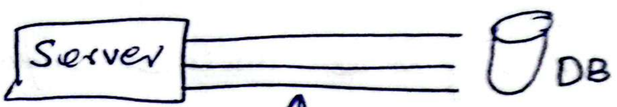
here computation is problem

→ Here established TCP Connect ↳ 3 way Hand Shake

How to make it better?

what's hectic? creating a new connect everytime

Connection Pool (always used)



↓ is a technique where a set of initial connections is established and mantain for resused.

↑ pre-established connections

↙ it save browser networks, reduce load on DB

↳ How it implement

How many connection Established?

min = 3
max = 100

→ if start 3 connection 3 request

→ if 4 request it established 4 connection

→ but less then 100

→ terminate connection that are not used for 10 min

Fundation Topic In System Design

Database	Scaling	Concurrency
Caching	Delegation	Communication

Concurrency, Communication } → most of System not all

Database: → one user multiple blogs
→ Multiple users

user
- id
- name
- bio

blogs
_id
- author_id
- title
- is_delete
- published_at
- body

→ Schema

Importance of is deleted [soft delete] → every time hard delete re-balancin
many time

when user involkes delete blog, insteated of DELETE we UPDATE
↓
hard
Delete

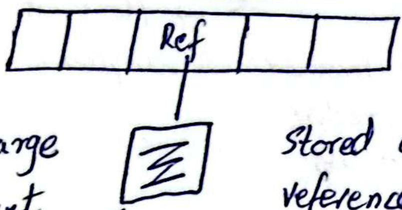
key reasons: Recoverability , Archival , Audit
↓
Google Drive

Audit → Not hard Deleted
Further use

* Eazy on the database engine [No tree re-balancing]
↓
delete is batching like gogle drive delete

Coulumn Type: body vs bio

long text



large text
Stored as reference

LONGTEXT

Stored disk location and store reference

Expensive Read

Short Text

Short Text
Stored along with other columns

VARCHAR

Stored on location

Storing datetime in DB

datetine as datetime
↑
Convenlient
Sub-optimal
heavy on size and Index

02-04-2025 T 9:01:362
↑
Serialized in some format

datetime as <u>epoch integer</u> — seconds since 1st Jan 1978

efficient
Optimal, light weight

4 byte ← <u>1798623547162</u>
↓
Stored as Integer
→ <u>Not</u> is able to read.
↳ take time

datetime as <u>custom</u> format
(int)

YYYYMMDD – <u>20250402</u>
↑
<u>Readability</u>

here we get local device Time if we need

[be creative about Solution]

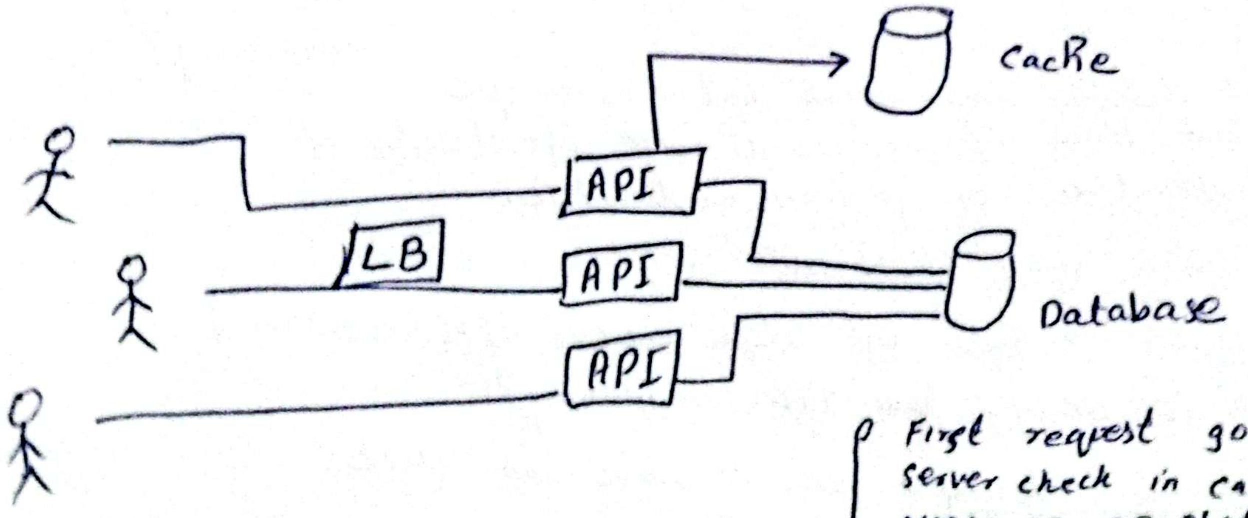
Every is have trade-off
→ <u>How to store SQL Datetime</u>
↓
<u>Doc</u>

Caching

- Reduces response time by saving, having <u>Any</u> computation → <u>Dynamic Programming</u>, <u>Memorization</u>, caching
    - Cache are not only RAM based

Typical use: reduce <u>disk I/O</u> on <u>network I/O</u> or compute

Caches are just <u>glorified</u> <u>Hash Tables</u> → <u>key-values</u>
↓
with some advanced data structures



[First request go server, server check in cache then query go on database
<u>Much faster</u>
reduce Expensis DB I/O

Fun exercise:

Find possible place that You Can as cache with an example

eg <u>Central</u> <u>cache(RAM)</u> for Application-level Cache

↓

Save DB computations

① why we can't add <u>cache</u> in DB as Component? <u>why not this,</u>
(RAM)

② In client Side when we used global storage in come cache?

③ when we do softdelete it increase size of database?
↳ but we do periodic delete | move at for specific time e.g <u>google Drive</u>

④ when we do soft_delete to hard_delete?
↳ eg → <u>facebook account Delete in 30 Days</u>

⑤ Do we comparision on caching like reduce data?
de-compression
<u>depend time & space</u>

<u>what is GDPR?</u>

⑥ When we <u>delete</u> by <u>CORN Job</u> it would be possible down_time?
soft
↳ Make <u>Time & Delete</u>

what is strong consistency?

⑦ When we delete like email that is unique index that time if we want with soft_delete it occur dupplication so go with hard-delete.

⑧ Is that DB is doing CRON Job?

⑨ There are function in Redis that delete entities after certain of time, how is doing, how TTL happening?

⑩ <u>How database Indexing</u> → Does it store on disk or RAM
↓
How indexing Make faster Database search?

⑪ Is soft.delete or Archival same thing?
↳ Move on
other place
our main DB
we do hard delete

⑫ when we doing comparsion Redis vs DynamoDB? why

⑬ When we do scaling DB what we do in connection pooling?
→ its depend on implementation ↳ Implement

⑭ How we do horizental scaling database?

⑮ Is there any props and cons when we using connection pooling?
→ worry about how properly doing connection pooling.
→ just blindly connection pooling is not good Idea.

⑯ when we choose we take write decision choose that particular tech stack?

Life is build of problem Just Find Optimal Solution ☺