Sub :

Day

Time : Date : / /

Push Notification

~~can~~ ~~be~~ App open না থাকলেও Notification আসবে.

Notification API → Generic API
↳ Client কে Noti Show করে

Push API → allows a service worker to handle msg from server
Service Worker → Scripts
uses low amount of rsc
Always listen করে Notific আসছে কিনা.
Press, এই process

Web push (process) → Server → Client এ msg
↓ এ যে যে component involve হয় in the process of pushing msgs from the server to the client

Push service → সব Browser এ নিজস্ব Push Service diff থাকে. ~~because~~ chrome এ যে Notific যাই ও brave এ যাই না, এ.এটা একটা server.

Web Push Protocol: It describes how an app/server interacts with a push service

Usage →

• Alert user to an imp event (যেমন কো some Notific যাবে. Only send an info. if clicked doesn't do anything

• Display an icon → click করলে কোন link এ নিয়ে যাবে

• Can integrate button, sound, vibration etc. → Browser এ করা tough.

Sub:

Restrictions → basic notification
Only ensure the supports করে না
যে browser sound/vibration

<u>Notification API</u>
Invocation API → Notific device এ show করতে.
Interaction API → যদি Notific এ click করলে
link এ নিতে, left swipe করলে Notific close
হবে → এগুলো handle করতে

Permission check Service Order → Notification দেখাতে
Request Permission Regular check করতে হবে.

Primary key → Notification log করতে
- Suggestion improve করতে চাইলে
- কোন Notific কতজন response করছে
- কোন type এর User কে কোন Noti
পাঠাবো — এগুলো track রাখতে use
হয়.

Service $\xrightarrow{\text{কাজ}}$ Display notific
handle interaction
sleep

Display notific
↳ Service worker → listens for $\boxed{\text{events}}$ (close, click
then based on the event, service worker
কাজ করে.

Push API

User যখন Permission grant করে, তখন ঐ
web site কে Browser এর push service grant
করবে. তখন ঐ website push service এ
subscribe করতে পারবে. Then push service Server
register (web App) কে object পাঠাবে. It has

endpoint : Server যেখানে req পাঠাবে, endpoint has
a unique identifier.
যেটা User কে Indicate করে

Key: App এর public key

Notification flow

Browser এর   এর server
Server Push msg Push Server এ পাঠায়. ~~Push~~
~~Service Browser এর service worker কে পাঠাবে~~.
Browser Then ঐ msg intended client কে পাঠায়.
(user এর Browser)

Client এর Service Worker Push msg এর জন্য
listen করে. যদি আসে Then Service Worker
handle করে → display msg
→ handle Interaction
→ Sleep

Sub: ______ Day ______ Time: ______ Date: / /

Notification API

Invocation → How Noti Appear
Client Side (Noti Appear → Server এর কোন Relation নাই)

Noti দেখানোর আগে Request Permission → User May Revoke it anytime, So Permission = granted check নাকি user করছে. Again Multiple times permission side annoying. So can add a matrix to restrict # of times show permission box

Display Notification: 1st Check permission = granted. Then call service worker. Then service worker shows it to the browser.

May Contain ~~bod~~ options: body, icon, vibrate

vibrate [100, 50, 100] → even i[0] vibrate then odd [1] silent হবে
i[0], i[1], i[2]
↓
Min 1 এর value হবে

primary key: Us Log User behavior for further research.
কোন Notification এ কতক্ষণ User interact করছে
based on that we can refine future noti suggestion

Sub: options ~~don't~~ may contain Action (key : value)

action (open/close) ↙ : for name base the action

· Particular user কে [illegible] Notifi (মেসেজ দেব →

Needs primary key.

· Service Worker কে Noti দেখাবে. (Noti Display হয় নাই still)

Client side ~~[illegible]~~ sends request to SW to show Noti along with options. SW Displays the Noti. After that, SW listens for event (noti close, click on it). based on that SW handles the event.

endpoint হচ্ছে server এর কাছে ওর কোন User specific হয় না. ওটা cookie info মতন না.

Server side

Push API : কাজ ⇒

Every browser has ~~each~~ own Push service. from the app User when grants permission for noti, it means, that app has permission to access that browser's Push Service. Then that app can subscribe to ~~that~~ own ~~p~~ browser's push service. Not the User's browser. ~~Then after registration~~ The push service will send an object to the app server : end point : User কে msg পাঠাতে হলে use হয়, এখানে টার unique identifier আছে. এখানে particular User এর unique id থাকে.

Key : App এর public key. App যখন push Service এ Noti পাঠাবে, তখন এই public key দিয়ে encrypted ~~[illegible]~~ করে ঐ end point এ Noti পাঠাবে. Push Noti কাজ করে Only in ~~[illegible]~~ HTTPS

$\downarrow$

Secure

Sub :

System Design, Performance ~~Loading~~ Scaling
& Load Balancing

Server এর কাছে
User req আসে :
Load

at first

Tinder App ⟶ Static
Business
Permanent ↘ 3টা tier 1টা

Initially App build এর time 3টা tier
server এ রাখতে চায় (Monolithic) ↗ 3টা tier
server এ রাখলেও Monolithic. if business
tier is divided into multiple server then
it's MicroService. Again, 10টা server এ same
Biz logic রাখলে it's Monolithic.

- User বাড়লে Server এ Load বাড়বে. Then

1st step : Performance Tuning
- Code refactoring
- Convert into asynchronous fn
- logic Improvement
- Caching (Common/Repeated info memory তে store করব. When needed can send that data without any calculation/or db access.)

if Performace tuning isn't possible anymore
on becomes expensive. Then switch to
vertical scaling. If it's not possible anymore
switch to Horizontal Scaling.

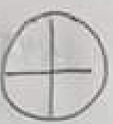
Suppose
Bottleneck → Biz logic (Match Making). Then
Biz logic Tier এ Load ~~[illegible]~~ বাড়বে. Then
to improve it →

Load balancing → 4টা server (Same Biz logic
4টা Server এই থাকবে)

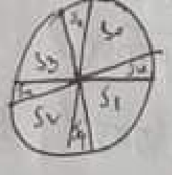
~~Consistent Hashing~~
4টা Server : (0-3) Random number দিব. Then
কোন Req আসলে ঐ Server এ যাবে → There's
a problem Here $\boxed{\text{Caching}}$. Fb Server :
এখন fb ~~ser~~ সব User এর info সব server
এ রাখা Not good. Particular user info
particular server এ থাকবে. ঐ User Req
আসলে Always ঐ Particular server এ যাবে.
It's done by Consistent Hashing.

User এর Req / Part of Req প্রত্যেকবার ~~দিয়ে~~ Hash করে
Still give me the same hash value. [illegible]
Mod the Hash value with # of Server.
Then যে Result আসে ঐ Server এ Req যাবে,

$$\begin{tikzcd}\text{User Req} \arrow[r] & \text{Load Balancer} \arrow[r] & \text{Particular Server}\end{tikzcd}$$


৪টা Server এ Load distribute হবে
হবে

Now add a new Server. Each Server এর
Amount of change বেশি হলে Caching becomes
Useless. Bcz if we send req to the prev Server,
it ~~is~~ is replaced by new Server. The target
is to minimize the amount of change.



$S_4 \rightarrow$ এ S_0-S_3 থেকে 5% S_4 এ যাবে.
ফলে কম User Caching Prob face
করবে.

After Load Balancing Switch to Microservice

Microservice

Biz tier কে Decouple করব

১টা service ১টা DB/table এর সাথে Related
if 2 services are closely connected then
we should not devide them.

Store Image: Profile এর সাথে রাখলে
Image service → ^AI implement করলে ঝামেলা.
So it's better to decouple them.

Matcher Service: 2টা user Matched হলে
তাহলে future recomm এ ওদের আর দেখাবে না.

- Client Server → Client $\underset{\xleftarrow{\text{Response}}}{\xrightarrow{\text{Req}}}$ Server
  (HTTP) → chat impossible

Peer to Peer → সবাই equal 2জনই Server/Client
(XMPP)

Sub :

Day

Time : Date : / /

Db Sharding : Horizontal Partitioning of the DB
Store user data in multiple Db based on location

Now Sort the user (for BD)

AUS
BD
USA
IND

Border Area: Problem has no solution

Sub :

System Design → Instagram.

Features

1 Stone/ Get Image

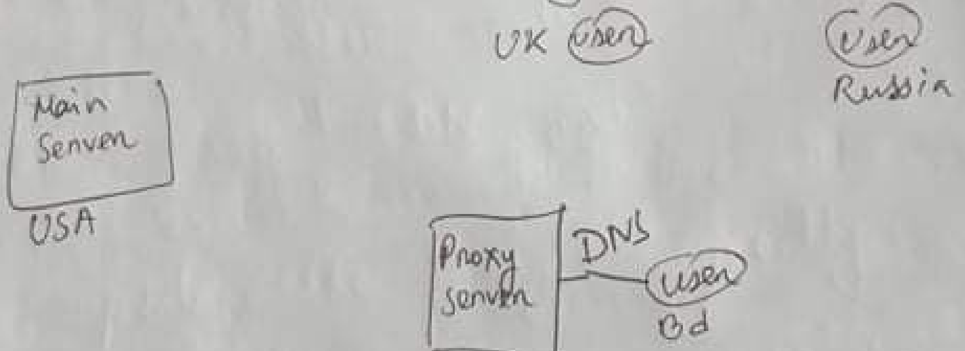
[Create a CDN & based on that users can get the imgs]

→ File

Prob: Senven USA, User BD. বারবার Req পাঠালে Load বেশি. Then Roundtrip অনেক বেশি time. (Req. Respon time বেশি হবে.

We need to implement CDN on the Image

CDN
Content Delivery Network.



Total usen base কে Multiple Regions এ Divide করব. Then set up a Proxy Senven (It delivers static Content (img, vid) btn usen & Main Senven. Usen will send req to Insta. DNS will pass the IP address (Proxy Senven not the Main Senven)

static Files
├ Img, vid, pdf
└ TTL (Time to leave)

Sub: Time: Date: / /

So user ~~[illegible]~~ basically sends req to Proxy server. Proxy Server will send req to Main Server.

CDN Advantage

1. Performance: Good, Round trip time কমে যায়.
2. Load on the Main Server lessens. Since only the proxy server is sending req instead of Users.
3. Security: CDN servers has firewall that provides extra layer of security

Qs How proxy servers gets the data on when do they get it

static files
├ Img, vid, pdf
└ TTL (time to leave) → why?: 7days পরে (7 days)

Proxy Server Main Server কে Req পাঠাবে to figure out the current situation of that file.

Day
Date : / /
Time :
Sub :

Push CDN: Main Server যখনই কোন Content change হয় তখনই Proxy Server কে, পাঠাবে. content
Example: Delete Photo হলে info টা Proxy কে পাঠাবে doesn't matter if anyone req for it or not. Similarly if someone update a photo.
যখনই কোন change হয় Main Server পাঠালে Proxy কে Update করে. Then user req পাঠালে Proxy থেকে পায়.
We can config from server when content expires/updated.

Q: When Push CDN good?
- Sites with small ~~traffic~~ traffic. Then server can update the proxy frequently.
- Static files changes/ frequently updated.

Sub :

Day

Time : Date : / /

<u>Pull CDN</u>

- fetch New Content from Server when first user req for it.
- TTL determines for how long content is cached (Expire হয়ে গেলে data fetch: when user req for it or when data is expired)

- Suitable for sites with High traffic

(Prob: TTL → 7 day. Now if a user deletes something after 5 days ও then someone asks for that img, TTL will provide him expired content.

CDN কে provide করে? Amazon, CloudeFlare

CDN Disadvantage:

1. Increase Cost: for all the proxy server
2. Waste of space (celebrity এর Profile তো proxy তে store করা হয়, since সবাই দেখে)
3. 1টা img এর জন্য Multiple URL. (ঐ same img BD server ও UK server diff URL দিবে.

Sub :

Day
Time :
Date : / /

4. Expired Content (Pull CDN).

Insta Features

(2) Like / Comment on Imgs (Post)

- Recursive cmnt: C1 └Reply C2 └Reply C3 X
  (1 Layer)
  Comment এর Reply তে Comment ~~কর~~ করা যাবে, But Reply তে ওদের comment ~~ক~~ করা যাবে.

- Like comment / Post

(3) Follow Someone.

(4) Publish News Feed.

Sub : Day Time : Date : / /

DB Design

Post

ID	User ID	text	ImgURL	Timestamp	Likes increment	Comment

Likes

ID	Parent ID	User ID	Timestamp	Active	Type
	Likes may be in a Post/ Comment. Need to store that			কোনো Post এ Like দিলে True. Like Remove করলে False	Post/Comment

Prob: Need to do a string OP to check whether it's Post/common (for large DB, expensive) to overcome that Type

How to count like/comment:
to query on the whole Db not feasible.
Instead we can add another column Likes.
Again Aggregated value should not be in the Post table. Make another table Activity

Sub :

Day

Time : Date : / /

Activity

ID	Parent ID	Likes	Comment

Follower

ID	Follower ID	Followee ID	Timestamp

Comment

ID	Parent ID	Text	Timestamp

user

Sub:

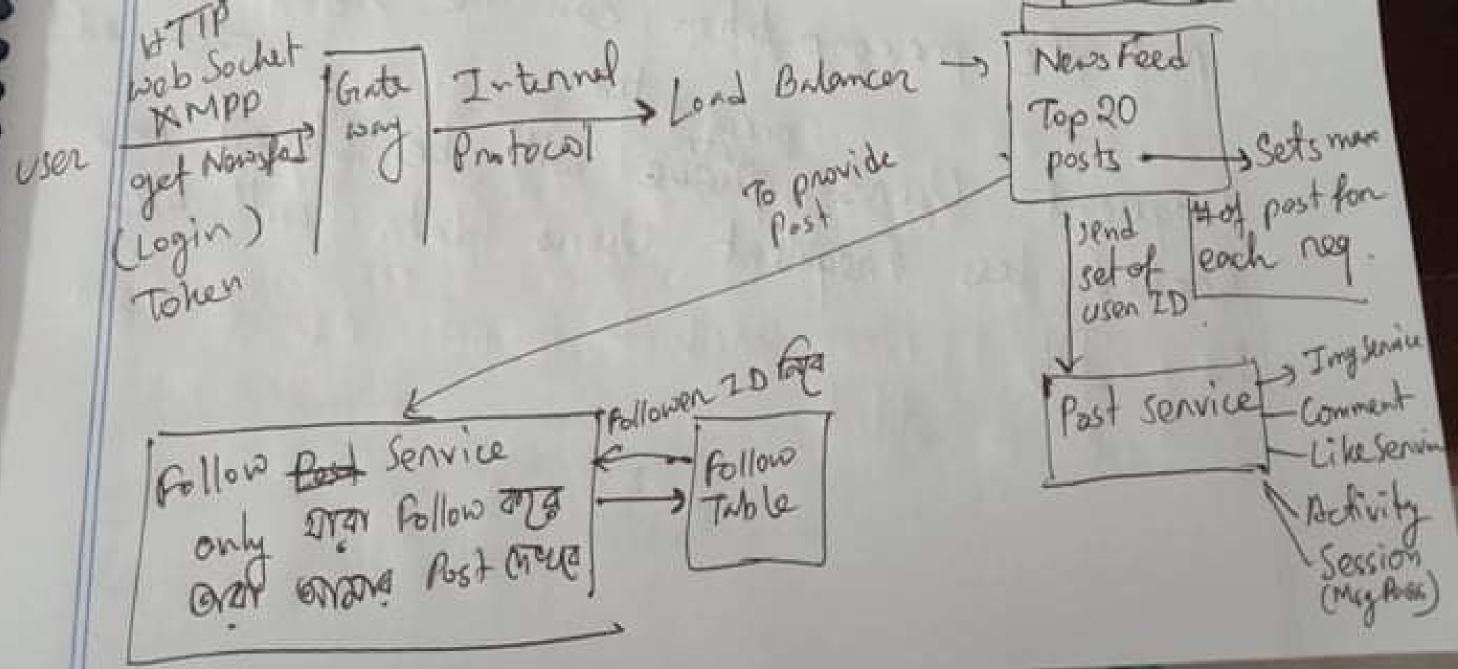
Cone Insta design

Show News Feed

User sends req to ~~Gateway server~~ Insta, Insta then sends req to Gateway Server to authenticate

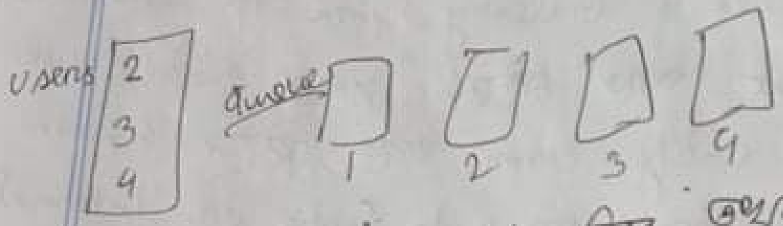
User 1, 2 chat করতে চাইলে XMPP Protocol use করবে. Outside entity comm করতে চাইলে known Protocol use করতে হবে But Insta এর internal (Http, XMPD)

Server can comm through internel protocol that only Insta knows. Pros: Security. Third Party doesn't understand what's going on.



Optimization: Can limit Post number (Reduce Load on the server)

: Can Precompute



Usen 2 যখন Post দিবে তখন Usen 2 এর
এর follower এর Queue তে Usen 2 এর
post(postID) update হবে. Followers News Feed Refresh
না করলেও Queue তে store হবে. Then
Usen Refresh করলে ঐ PostID এর Post show
করবে. → Precomputed.

Now the precomputation can be stored → Db
→ Cached.

Frequent Usens precomputed ~~queue~~ data will be stored in Cached. Less Frequent Usens data will be in Db.

Sub :

Day
Time : Date : / /

Precomputation Pros : If not, Senven এ প্রচুর
neq যায় each sec. In case of precomputed
data, the queues are updated, so no need to
send neq to the senven to get a post.
It neduces load on the Senven.

Gnp msg এ ব্যবহার

Notification

যাদের follow করি, সে Post করলে Notified হতে চাই.
User post করলে তার follower দের Push Notifi
পাঠাতে পারি. In case of celebrities (a lot
of followens) we can send [batch notific.]
Every 10 sec Insta will send ~~too~~ Notific to
its 10000 follower. Another option. Pulling
Notific. Device will check/poll every 1 min
to see if it has any notific. Insta will
send notifi only when a device ~~sends~~ poll
A lot of device may be offline, then Insta
won't send any notific.

Deployment

push changes or updates from one env to another
└→ Local → Dev → staging → Live

Initially change করত local env (1 coders pc)
live env (client) এ change করত as the whole new feature might break down the whole app.

Dev env এ যায়.
Local env এ কাজ করলে Dev server same.
Most of the case Local, Dev server same.
Dev env এ only new feature থাকে. Not whole microservices.
Staging server → exact Replica of live env.
(All the microservice + Newly Added service).

Deployment Process Flow
Start: New process এর code start
end: Successfully code change live/client কে দিতে পারব
Deployment: New product launch or a new feature ● added.

Sub :

① S/W Deployment Plan: 1টর env থেকে আরেকটার env এ deploy (Dev → Production) : URL change. credentials change হতে পারে. 1st step : .env change, static Repo 2টর diff server এ থাকে.

② Actual development

③ Testing the change → Diff Developer might work on diff module, they will do the Unit Testing

④ Deploy change to the production/live env

⑤ Monitor changes → speed ঠিক আছে কিনা.
→ কোন sub module break করছে কিনা
→ check user feedback.

Deployment types
- ~~Meta~~ Dep of Meta data: whole control থাকে developer এর কাছে. : codes, CSS.
- Dep of content: photo/vid. Content creators can add/modify their contents

Deployment Best Practices

1. Use Git

2. Work in branch: Create a branch. Change the code. Test the Code. Then Review code. if no major conflict, push it to main branch (live with my change)

3. Review the difference

4. Deployment Schedule: While deploying u have enough developers with you. Deployment Schedule করব যখন there's least amount of active users. So data loss minimum.

5. সব developer কে deployment এর permission দিব না. Only those who have adiquete amount of knowledge about the s/w should have access to the deployment.

Sub :

6. Stay Calm → Error দিলে Roll back. Prod env এ
Production env এ error আসলে Roll
error ঠিক করব না. For small error Roll
back not necessary.

DevOPs → Methodology, set of best/good practices
→ Shorten delivery times
→ fasten development
উপর depend করে

depends on করে

Continuous Integration (CI): New code is
integrated with the shared repo

cont Deployment (CD) → কোন change আসলে
এটা whole set of test দিয়ে যাবে. Then
live হবে.

S/W Release: Existing Code → made some changes
$\boxed{V.2} \xrightarrow{\text{gave new name}} \boxed{V.3}$

Deployment: ~~Ghon~~ After giving new version name
live এ গেলে Then এটা Deployment