News Feed System Design

We need to build a system for News feed (maybe LinkedIn, Facebook, Instagram).

General points:
- We have news feeds and infinite scrolling.
- Most of the post on Insta are related to our content only but we don't follow them.
- Earlier the Algorithm was bit different from today, earlier we only used to see the contents of the people we are following. Now Algorithm has changed, if you say "Pizza" in your phone 5 times, there is a high chance that you will see an ad or content related to pizza.
- Earlier, also when you liked a post say with #mountain, there is a high chance we get posts with #mountain in our feed because the system thought that person might be interested in posts of #mountain.

If you get this problem in the interview, don't think about these above-mentioned features atleast in the start. Why? Because they are high-level features and lot of machine learning is involved.
Say if you spend some time on a job post on LinkedIn, you will keep seeing similar job posts on LinkedIn, that's how the algorithm is currently trained. If we use Instagram in the morning, we get some motivational quotes, while in the evening, it gives like traveling videos, in the night it shows poetry etc. thats how it is trained.
We could design all these functionalities but the only constraint we have in interview is time. In real-world scenario, a feature design could take up to multiple weeks to get approval while in System design interview, interviewer expects you to deliver a workable solution with some depth, not very high-level. They want to know whether you understand the Entity part, API part, multiple services, different databases, need for cache, how to implement cache, etc.

When you start an interview for News Feed, ask interviewer what features do you need, thats our Functional requirement

Functional requirement:
- Any user should be able to create post (Create post)

- If someone has created the post, user should be able to view posts (View posts)
- We should have an option to follow user also (Follow users)
- Page scrolling / pagination capability (Page scrolling)
- User can like posts (Like post)
- User can comment on posts (Comment on post)
- Share the post
- Save the post
- Repost a post

It’s a Many-to-many relationship, one user is following multiple users and the same user is being followed by multiple users.

For example, User1 will only see the contents of the *users they are following*
that means, if User1 follows 2, 3, 4 only and if they spend time on User2 travel posts, User1 will not see any travel-related posts from other users that they are not following.

Due to time constraints, we pick only *Create post, View post, Follow user, Page scroll*

Non-functional requirement:
Whenever we start talking about Non-functional requirement, we got to start thinking about CAP theorem
- High-Availability is a must (Our system must be running and we should see some post in our feed all the time)
- Consistency (is it necessary for all the following users to see that post in real-time? No). Eventual consistency is ok
- Low-Latency (Response time should be low: 500ms approximately)
- How many users we have? Users: 2 Billion (YouTube also 2B).

What are the core entities?
- User table
- Post table
- Follow table

How will you design your APIs?
- *Create post*
==> POST { content: image/video/text }
compressed video size: 1MB
when post is created, we get a 201 response { post_id: “123” } with post_id. LongInt 8 Bytes
201 response
{

post_id: “123”
}

- *View post*
==> GET /feed?pageSize={size}&cursor={cursor}
200 response with
{
posts: [posts],
nextCursor: 2pm
}

What’s cursor?
In Cursor, I will share a timestamp to this Feed endpoint say 1pm, which is the previous timestamp. In the meantime, it processes the GET request, it is going to get the next cursor value (say 2pm). So we provide data between 1pm and 2pm. It is upto 2pm. Whenever I refresh the page or scroll again, I am re-sending the GET Feed request, say at 2:30pm, I will send the {cursor} (timestamp) as 2:00pm and in the nextCursor as 2:30pm (current time). You have fetched all the posts upto 2:30pm

{
posts: [posts],
nextCursor: 2:30pm
}

In DB, we have multiple rows of data
Post table

ID	Content	Creator_ID	CreatedAt

Normally, in Instagram, it cannot hold data for all 30min interval. It may hold contents in 2min intervals. If scrolling speed is too high, it cannot get all the data in one go, it will get the next 2min data first then next 2min and so on. Whenever we refresh application again, it invokes the GET Feed API with the latest 2min content only.

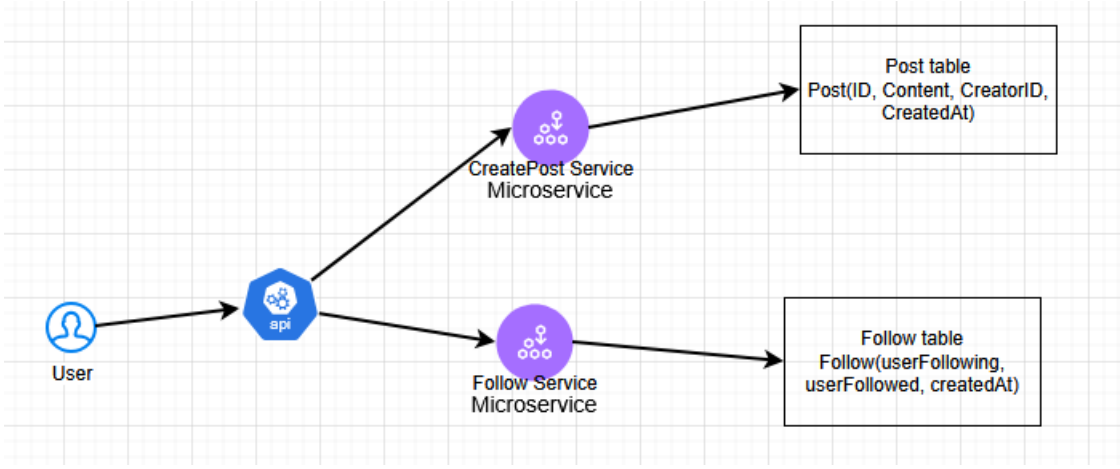
- *Follow user*

==> PUT user/follow/{userID}
{userID} of the person we are following. When User1 clicks the follow button, this API will be invoked. userID can come from user login session

```
{
        // No body required
}
```

Data storage:
SQL, NoSQL or GraphDB are fine.

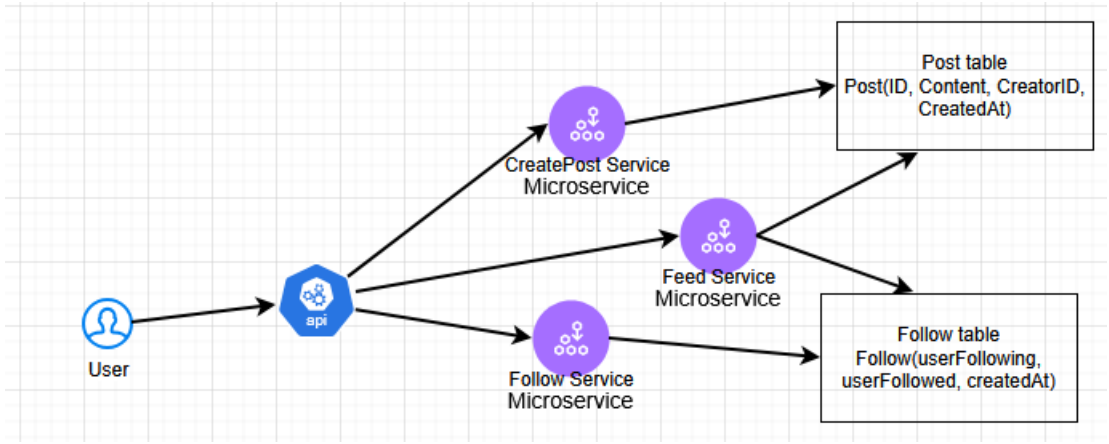
High-level design:
You don't have to be pitch perfect about it.
We have 3 services here: POST (create our post), Note: User table is not created because it is straight-forward. This is only a high-level design



Functionality:
1. For a particular user, we may have to find their followers
2. For a particular user, we have to find who they follow
3. When we are creating Feed service, getFollowingAccounts() and accordingly we will getPosts(userID, cursor, newCursor). If I am following 10 accounts that means, getPosts() will be called 10 times for each userID (userFollowing). I think we will get userIDs from getFollowingAccounts then pass into getPosts(). Then it will sort records based on Timestamp in reverse order.

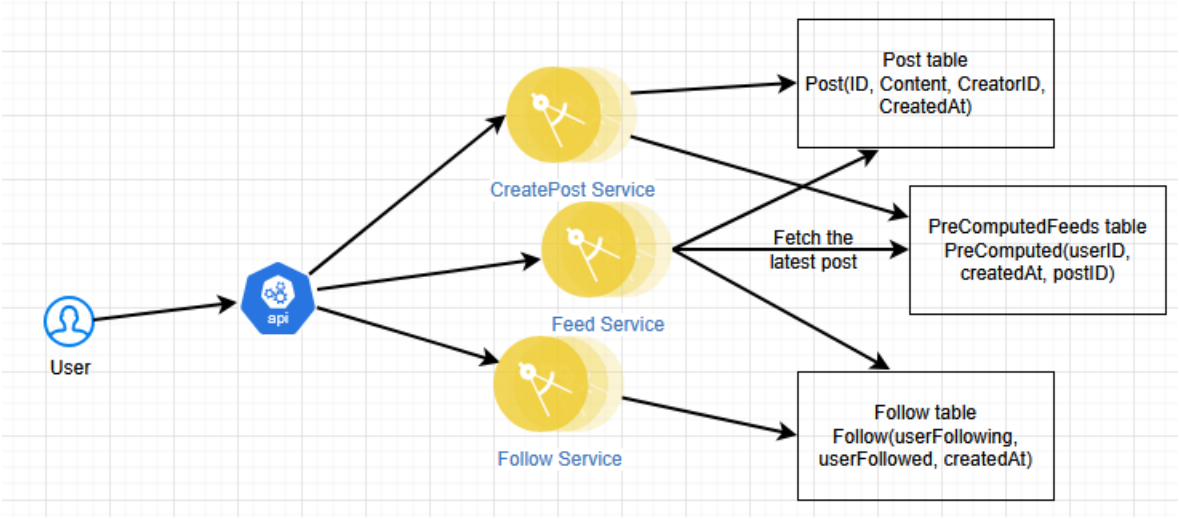
These 3 steps look tedious, too much work, if it is too much work, it will add latency

4. We can decrease Latency using *indexes*. We can create indexes on some keys. Git, Facebook use GraphDB, much efficient. In our example, we could use either SQL or NoSQL. In GraphDB, we will create a directed edge for the userFollowing and userFollowed nodes. Coming back to index, we can see userFollowed and userFollowing used a lot so we can put indexes on both the attributes. Also we can create index on creatorID and createdAt. If our system becomes very heavy, we need a lookup table.



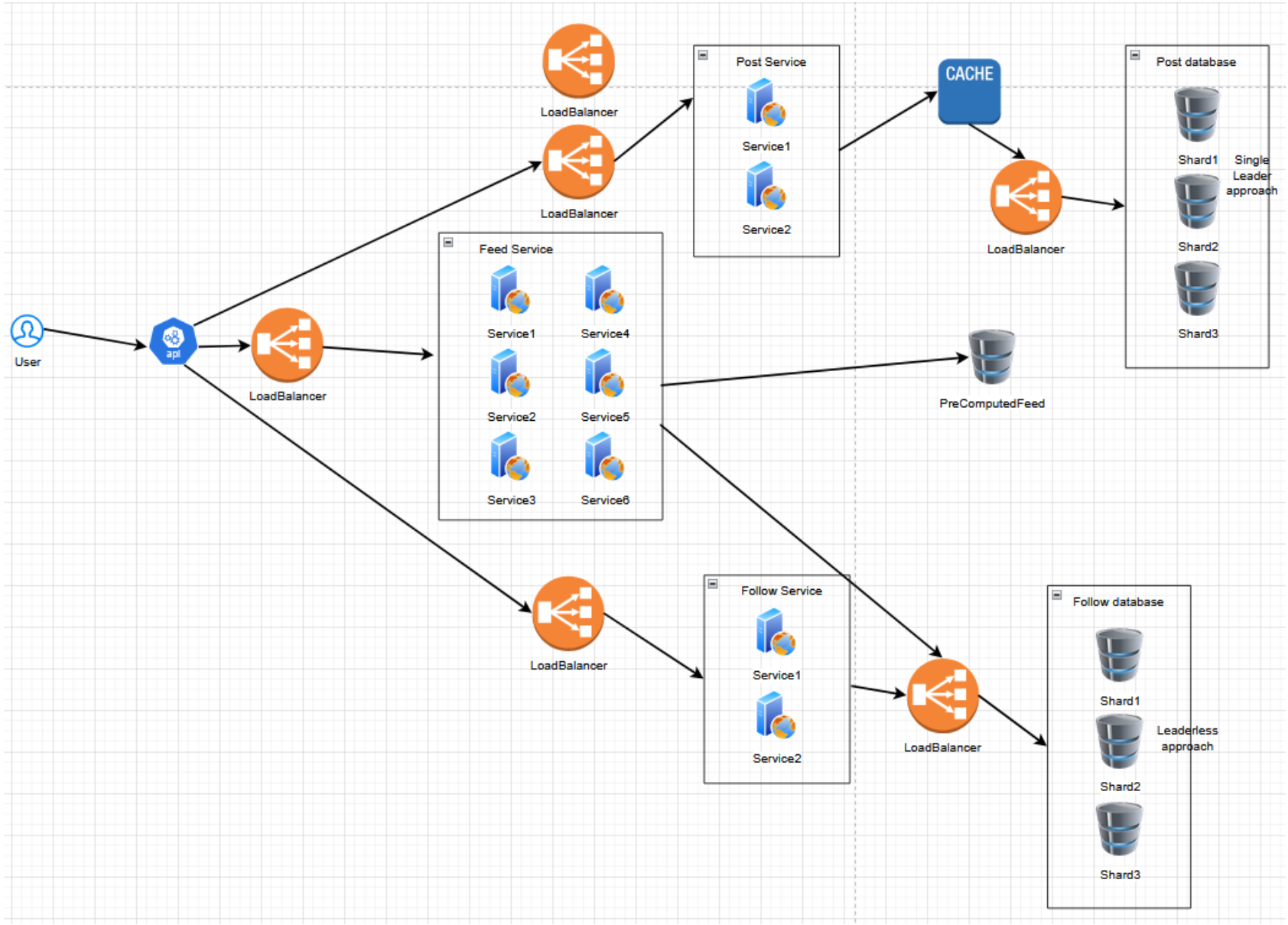
Feed service interacts with both Post and Follow tables to get the data. The job of Feed service is to fetch userFollowing feeds for a particular user. First it fetches userIDs of userFollowing for a particular user_id then goes into Post table to fetch all the records corresponding to those userFollowing userIDs. Those fetched records/posts will become feeds for the logged-in user.

Lookup table to decrease latency
We can create another lookup table called as “PreComputedFeeds”, whenever we create a post, we can make an entry in this table. It will have userID, createdAt, post_id. Feed service later uses it to fetch the latest post. We wont keep all the records in this table, only like latest 5min or 1hour records. Say after 1hour, data stored will be deleted. By keeping a shorter table, it is better for quick lookups for the Feed service.



PreComputedFeeds, will go into Background jobs for cleanup

<u>Deep Dive:</u>
Number of active users: 2 Billion
How long a post remains in our DB? Infinite time
500 users are creating posts in a second.
Request per second (RPS) = 500 Million requests / second for Post service
In a second, 200 Million users are following other users
RPS = 200 Million / second for Follow service
2 Billion users are checking their Feed every second
RPS = 2 Billion / second for Feed service


User
api
LoadBalancer
LoadBalancer
LoadBalancer
Post Service
Service1
Service2
CACHE
LoadBalancer
Post database
Shard1
Shard2
Shard3
Single Leader approach
Feed Service
Service1
Service4
Service2
Service5
Service3
Service6
PreComputedFeed
LoadBalancer
Follow Service
Service1
Service2
LoadBalancer
Follow database
Shard1
Shard2
Shard3
Leaderless approach

If someone asks what if your LoadBalancer fails, then the answer is we can have multiple loadbalancers. We can add Cache for Post database to decrease latency. What's the eviction policy for cache? When will the cache updated? Check previous notes for answers.

YouTube channel: ByteByteGo good for SystemDesign
https://www.youtube.com/ByteByteGo

https://arpitbhayani.me/masterclass/

https://www.youtube.com/@AsliEngineering