Design Twitter

1. Here how would you design twitter?

And then here you have

What are the endpoints aavialbe?

**Part 1 start with the API deisgn here**

**API Design part 1:**

**Post a tweet**

This API will allow the user to post a tweet on the platform.

**Parameters**

User ID (): ID of the user.

Content (): Contents of the tweet.

Media URL (): URL of the attached media *(optional)*.

**Returns**

The status of the response

**Part 2:**

**Follow or unfollow a user**

**Parameters**

Follower ID (): ID of the current user.

Followee ID (): ID of the user we want to follow or unfollow.

Media URL (): URL of the attached media *(optional)*.

Returns whether successful

API 3

Get newfeed : param: user id of the user

Retursn: tweets

**Part 3:**

High level of architecture design microsservice

Media Service

Search

NewsFeed

Tweet

Analytics

Notification

**Feed generation for a partiular user here**

Let’s assume we want to generate the feed for user A, we will perform the following steps:

1. Retrieve the IDs of all the users and entities (hashtags, topics, etc.) user A follows.
2. Fetch the relevant tweets for each of the retrieved IDs.
3. Use a ranking algorithm to rank the tweets based on parameters such as relevance, time, engagement, etc.
4. Return the ranked tweets data to the client in a paginated manner.

Step 2:

Publishign:

**When you want to publish to a specific user here**

**Approach 1:**

**Fan out on read here (pull model)**

When a user creates a tweet, and a follower reloads their newsfeed, the feed is created and stored in memory. The most recent feed is only loaded when the user requests it. This approach reduces the number of write operations on our database.

The downside of this approach is that the users will not be able to view recent feeds unless they “pull” the data from the server, which will increase the number of read operations on the server.

And then here

Fan out on write here: fPush model here

In this model, once a user creates a tweet, it is “pushed” to all the follower’s feeds immediately. This prevents the system from having to go through a user’s entire followers list to check for updates.

However, the downside of this approach is that it would increase the number of write operations on the database.

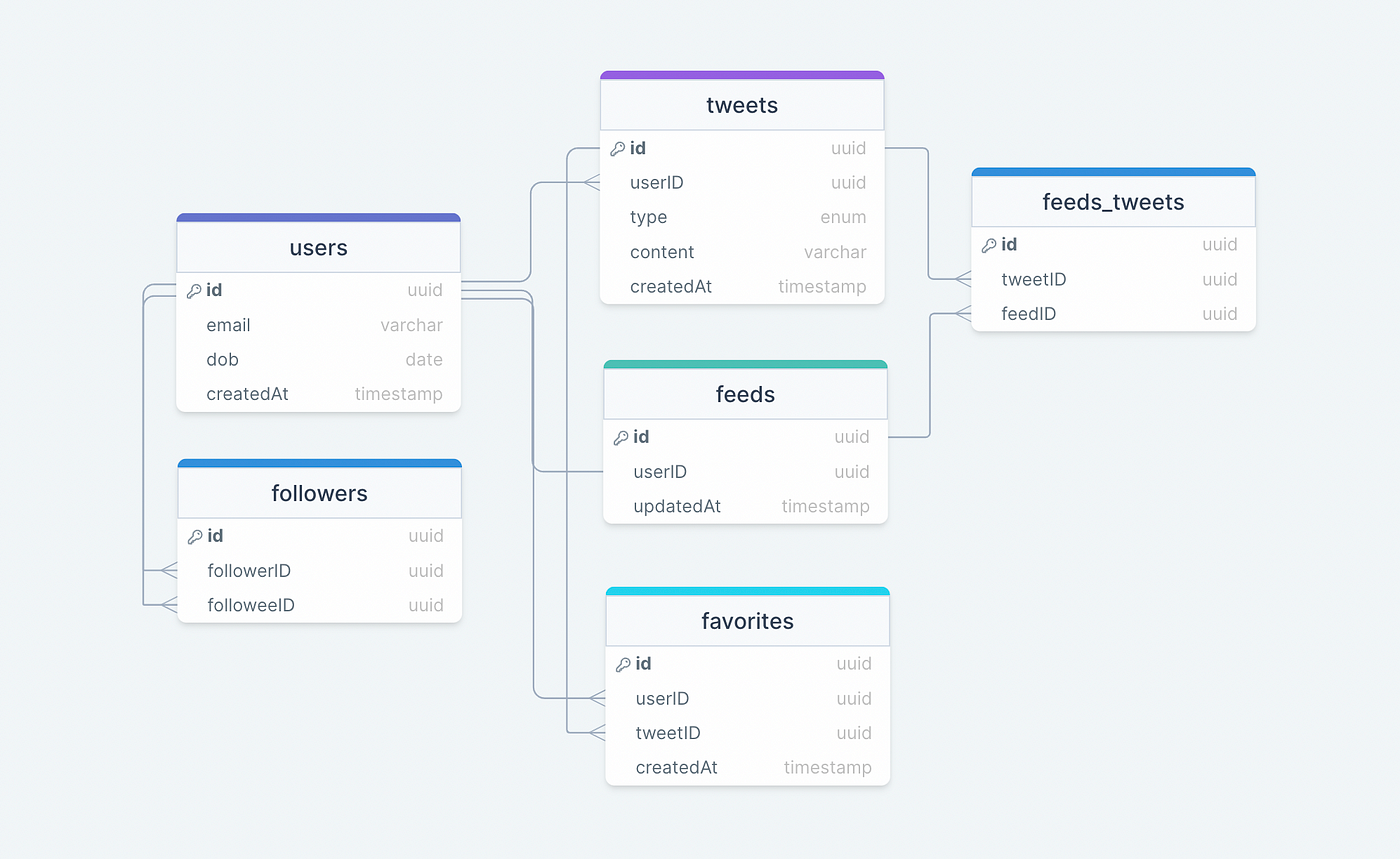
**What are some criteria userd for the ranking algorithm here?**

1. Closeness : How often a user likes, comments or msges someone,

If a user frequently likes, comments, or messages the edge creator, then the value of affinity will be higher, resulting in a higher rank for the post.

1. the value assigned according to each edge. A comment can have a higher weightage than likes, and thus a post with more comments is more likely to get a higher rank.

System design part 2 here:



What’s the diff between feeds and

Tweets can have many types here (text, image, video and so forth here )

**Tweet feed**

Your Twitter feed (also called a timeline) is your list of updates from users you are following.

**Tweet:**

The most basic definition you know, a tweet a message posted via tweeter here.

**favorites**

This table maps tweets with users for the favorite tweets functionality in our application.

Feed\_tweets

Part 4: How would we design the database table here

A column called originalId that refers to the original tweet in the twettable here.

**Data paritioning here**

To scale out our databases we will need to partition our data. Horizontal partitioning (aka [Sharding](https://karanpratapsingh.com/courses/system-design/sharding)) can be a good first step. We can use partitions schemes such as:

* Hash-Based Partitioning
* List-Based Partitioning
* Range Based Partitioning
* Composite Partitioning

The above approaches can still cause uneven data and load distribution, we can solve this using [Consistent hashing](https://karanpratapsingh.com/courses/system-design/consistent-hashing).

*For more details, refer to*[*Sharding*](https://karanpratapsingh.com/courses/system-design/sharding)*and*[*Consistent Hashing*](https://karanpratapsingh.com/courses/system-design/consistent-hashing)*.*

Mutual friends

1. Build a social graph, a directional edge from follower to followee here

**Caching:**

1. Which cache evictino would you use here?
2. Which cache eviction did you user here?