# Instagram Design

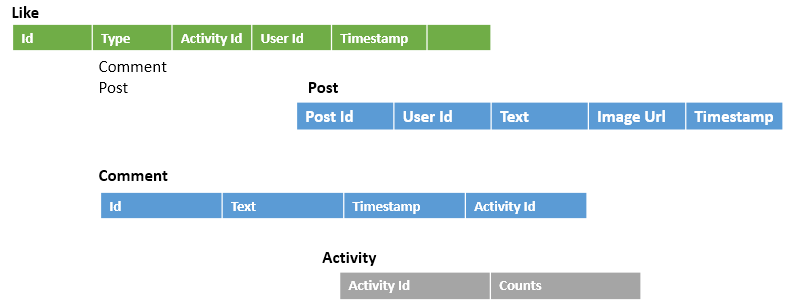
## Getting started

### Features for consideration

* Store/Get images
* Like post
* Comment post (recursive – comment on comment)
* Follow someone
* Publish a news feed

### Database design

We need **Likes Comment Post** tables.



**Activity** table stores the aggregate count of likes for a post or comment. Which will store the aggregate data and will reduce the query time if we try to get the same joining Post/Comments and Like table.

**Follower** table will store the details who follows whom.



### Design Diagram

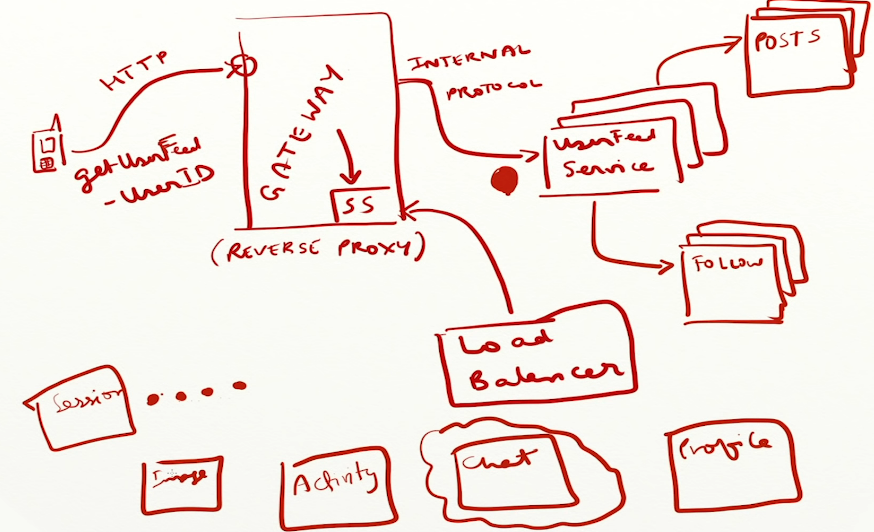
We will be using **Gateway** which will encapsulate many thing like authentication token, protocol conversion can be done here, will work as reverse proxy.

We will be having a service **UserFeedService** which will provide the feed for the corresponding user id. To make it more performing we can run multiple service in different server. We need **Load balancer** to let client route to a service from multiple.

Rather every time **Gateway service** talks with **Load balancer** we can store a **Snapshot of mapping** in Gateway service. Load balancer sends the snapshot to Gateway and will be periodically updated say in every 10 sec. So where the request will be sent will happen from gateway service rather communicating with load balancer every time.

Load will be balanced using **consistent hashing.** Here by hashing the user id we will get where the request to send it.

UserFeedService will communicate with **Post** and **Follow** service.



### Scaling post service

For every user feed request we have re compute all the posts the user will see will be too expensive especially on post service, it will get bombarded. It’s not going to scale.

The best solution here is **pre-compute** the user feed and return when user request.

How we are going to pre compute?

User feed is a set of post I want to see and post are the post for the people to whom I follow.

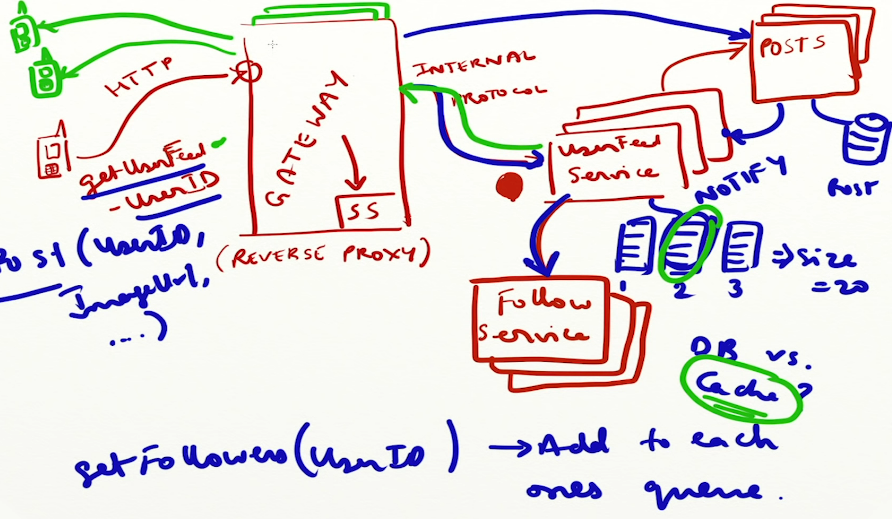
Breaking down it, every time a user to whom I follow post something on Instagram, We need to update my user feed.

So whenever there is a post the **Post service** should **notify** user feed service to update the corresponding user feed who follow this user (add to their queue). So we will have incremental update.

We will store the data in **cache.** If we lose the cache we can re-compute.

When a user is logging in frequently we don’t need to use cache memory for those kind of user. When they login again we can re compute using brute force approach connecting with Post service.

With this approach we can send notification to the user cellphone when someone they follow added a post. Off course they need to poll.



We can push notification from server using **Web Sockets** we can even use **Long pooling** so on and so forth.

### Celebrity post

When a person with millions of followers makes a post if user feed service behaves in a normal way it’s going to send post to a million users and that’s a too heavy task to be done immediately and could crash the system.

2 approaches to solve this problem

* Batch processing
* Don’t send notification to user, wait for them to poll from you

In Batch processing we will send notification to 1000 of peoples in one batch and basically could rate limit the number of requests we are sending outside.

The **Push model** will fan out where millions of request will send out, where in **Pull model** things will go slowly because all the users won’t pull at same time.

Here in Instagram we can use a **Hybrid model.** Where a normal user makes a post we can make use of push model and when a celebrity made a post could use pull model.