Functional

1. Users - able to post, delete, view the post
2. Users can follow the users
3. System can generate news feed for those who are following a particular account.

Non Functional

1. High Availability
2. Low latency for news feed(200ms)
3. Eventual Consistency

Design Ideas

Read heavy

We can Store images in S3 and ids in Db and also we can have duplicate of S3, so that retrieval is fast i.e. we can use CDN.

Estimation

500 mn total users = 500* 10^6 ↑

DAU - 1 Mn, and each Active user upload 2 images so total 2Mn photos

1 image size = 200kb, so 2*10^6 * 200* 10^3 = 400 * 10^9 = 400 GB

For 10 yrs = 400 * 365 * 10^9 * 10yrs = 1460 TB

HLD for Download and upload

Photo -> id(Long), userId(long), place(String), path(String), Creation Date

User -> id(Long), userName(String), followers(Long), followee(Long), totalPosts(Long), email, phone, dob, creationDate, lastLogin

UserFollow -> followerId(Long), followeeId(Long) [Both together will be a primary key i.e. compositeKey]

Since we need joins for photo and user, it will be better to **SQL**, but disadvantage is as DB increases it will be difficult to sharding as SQL dBs don't provide inbuilt Sharding capbailities. NoSQL will be difficult to join.

1. Total bytes can be around 100bytes, Total users were 500 M so total 50 Gb user Data.
2. Image size= 2M images per day = 2*10^6*150 = 0.3Gb/Day * 10yrs= 1.1 TB
3. For user follow = 30bytes , now one person follow on avg 500 users. so total 500 Mn * 500*30bytes = 5 TB

Component Design

UploadTime > DownloadTime, assume 1 server has 500 connections. so user uploading a photo will hold a connection longer.

To balance this, we can split reads and writes. This design pattern is called CQRS
RateLimiter can be used.

We can use RoundRobin load balancing here as every req can be treates as new req. and there is no req like every req should go to one db so consistent hasing algo can be used.

Partitioning on UserId advantage - images will be on same shard. Range based partitioning can be done. Disadvantage is if influencer is there many images are there.

Partitioning on ImageId advantage - equal Servers.
Disadvantage - slower lookup as multiple shards are there to lookup and retrieve.