Feature requirements:

1. New feed of posts

2. Add post: photo, albums

3. Author, caption and like count

4. Like a post

5. Fast infinite scroll

6. Off line mode

7. 10000 posts

**A diagram of a software development

Description automatically generated**

**What ‘s the data model here**

**Object 1**

Post

postId: Int

type : int (photo, album)

author: userId

likeCOunt: int

cpation: string location: string

photoUrl: string

Object 2:

**User**

userId – int

name – string

avatarImageUrL – string

Object 3: here

**Like**

userId – int

postId – int

successfullySent: bool

**API based endpoints ehre**

Rest

Get /users/<user\_id>/feed

postId

limit int (20)

page – prev/next

post /users/<user\_id>/likes

[postId value: boolean (to remove or add like here )

Functinoal requirement

1. How to show the likes when there is no wifi?

Storage

1. We can store this in the Room database,

2. We can add a cache with an expiry of 2 minutes, and if things expire, we can just evict the old data

3.

So we need sth to store thigs in the database layer here.

2. Load images

This is quite important here, when no wifi. What to do? And in this case I would say maybe

And then the code here would make a lot of sense here

1.

This goes in the general mobile system here:

**How are images loaded using glide?**

When you make a network req, glide always checks in the db if the image is there already or not?

If the image isn’t in the in-memory cache, Glide next checks its disk cache. If the image is found on disk, Glide loads it into memory and returns it. If the image isn’t in the disk cache, Glide loads it from the network.

How to imrpove scrolling performance with a lot of images?

1. Using recyclerview

2. Only load content that is visible on the screen

3. Do not use nested layout here, use just 1 constriantllayout to flatten everything here

4. How to lazy load images on Android? (this probably won’t work)

5. Add thumbnail url

The [thumbnail()](https://bumptech.github.io/glide/javadocs/400/com/bumptech/glide/RequestBuilder.html#thumbnail-com.bumptech.glide.RequestBuilder-) will be displayed while the primary request is loading. If the primary request completes before the

6. Use cursor based pagination here

- Cursor indicators the start of the page here, usually it can be based off of sequential id or timestamp (if you have auto-incremented ids or timestamps)s

How would we fetch the next 10 items here GET /users?cursor=10&limit=10

On the server side this looks like:

SELECT \* FROM users  
WHERE id > :cursor  
ORDER BY id ASC  
LIMIT :limit;

Which is actually better than what you had initially here

Fetch feed posts and send post feed here

1. And then here we have the code here

2.

Let's say you have three different view types (

1. News Feed View Type, Image Gallery, Grid Layouts etc) that need to be loaded with images when user loads those views. Assume you have the backend system infrastructure given to you, how would you design Image Loader that efficiently manages loading of different images on each of these views. What are the different design choices and tradeoffs you'd make?

3 different types of image views here

1.