Experiment No: 3

Student Name: Preeti shukla
UID: 23BCS12815
Branch: BE CSE
Section/Group: 23BCS_KRG-2_B
Semester: 6th
Date of Performance: 28/01/2026
Subject Name: System Design
Subject Code: 23CSH-314

1- Aim - To design a social media platform similar to Facebook or Instagram.

2- Objective:

- To understand social media application workflow.
- To design functional and non-functional requirements.
- To create the system architecture (High-Level Design – HLD).
- To design modules and classes (Low-Level Design – LLD).
- To implement core APIs for user authentication, posts, feed, likes, and comments.

3- Requirements:

A- **Functional Requirement**

- Client should be able to register and login to the application
- Client should be able to create post (text / image / videos)
- Client should be able to follow each other (or send friend requests)
- Client should be able to like or comment on the post
- Client should be able to view the feed of post from users they follow

B- **Non-Functional Requirement**

- **Scalability:** 500M DAU
- **Consistency & Availability**

Since this is a **social media application**, the system must be **highly available first and then consistent**.

Reason:

If the application is not operational or functional when required, there is no meaning in developing the application.

Example:

If Instagram is down for **1 hour**, it becomes a major issue. However, if the application is running and a post takes around **500 ms** to reach followers, this delay is acceptable.

Hence:

Availability >>> Consistency

- **Latency**

Uploading or publishing a post should take approximately **500 ms**.

4- API endpoint creation

A. User On-boarding API's

- **User Registration:** POST API CALL
  POST /api/users/register_user
- **User Login:** POST API CALL
  POST /api/users/login
- **User Data Display:** GET API CALL
  GET /api/users/{user_id}/profile
- **User Data Update:** PUT API CALL
  PUT /api/users/{user_id}/profile

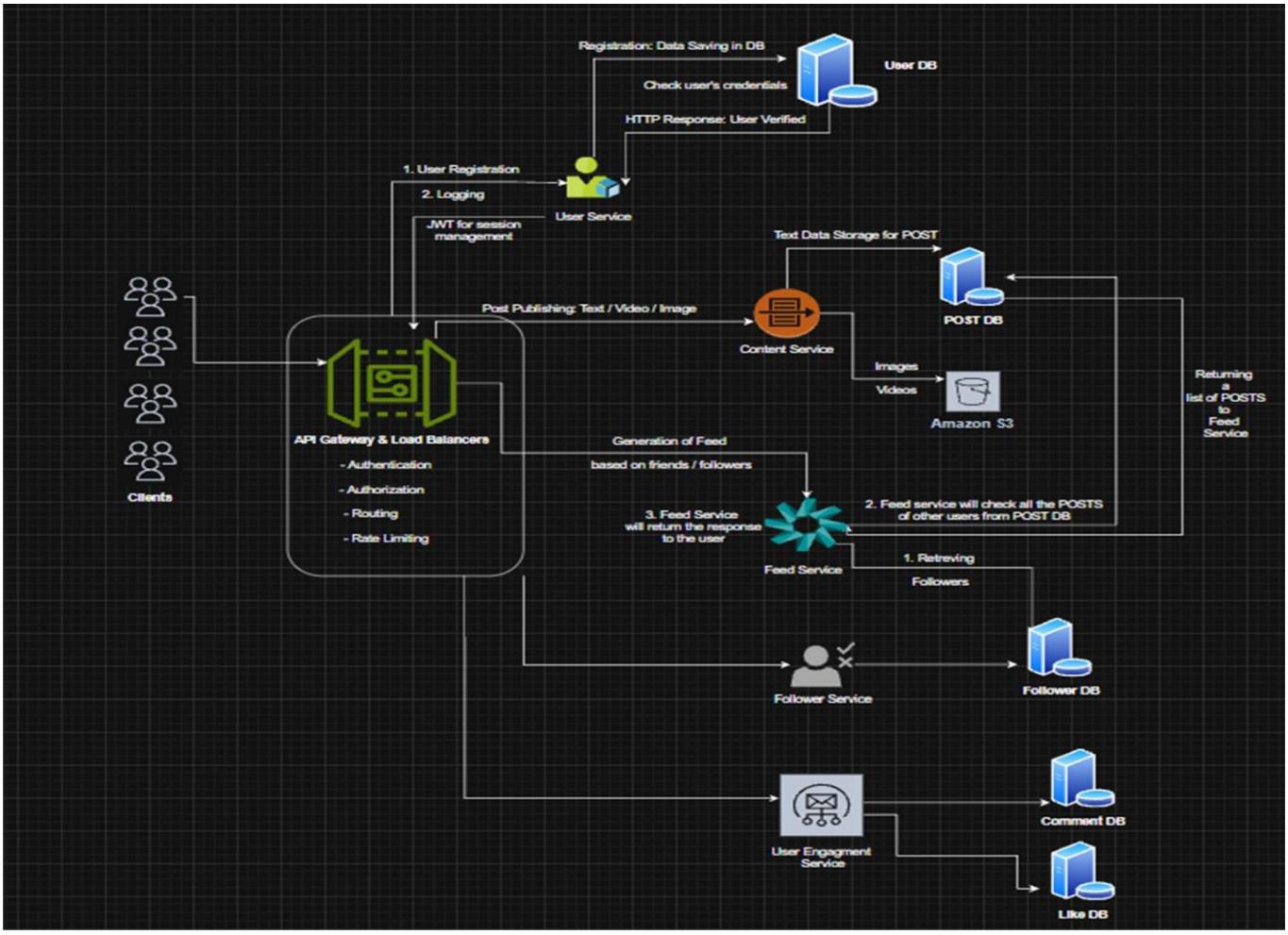
B. User Post's

1. POST /api/{user_id}/posts
2. GET /api/posts/{post_id}
3. PUT /api/posts/{post_id}
4. DELETE /api/posts/{post_id}
5. GET /api/posts/feed?limit={limit}&offset={offset} : **Pagination**
6. GET /api/users/{user_id}/posts : **Pagination**

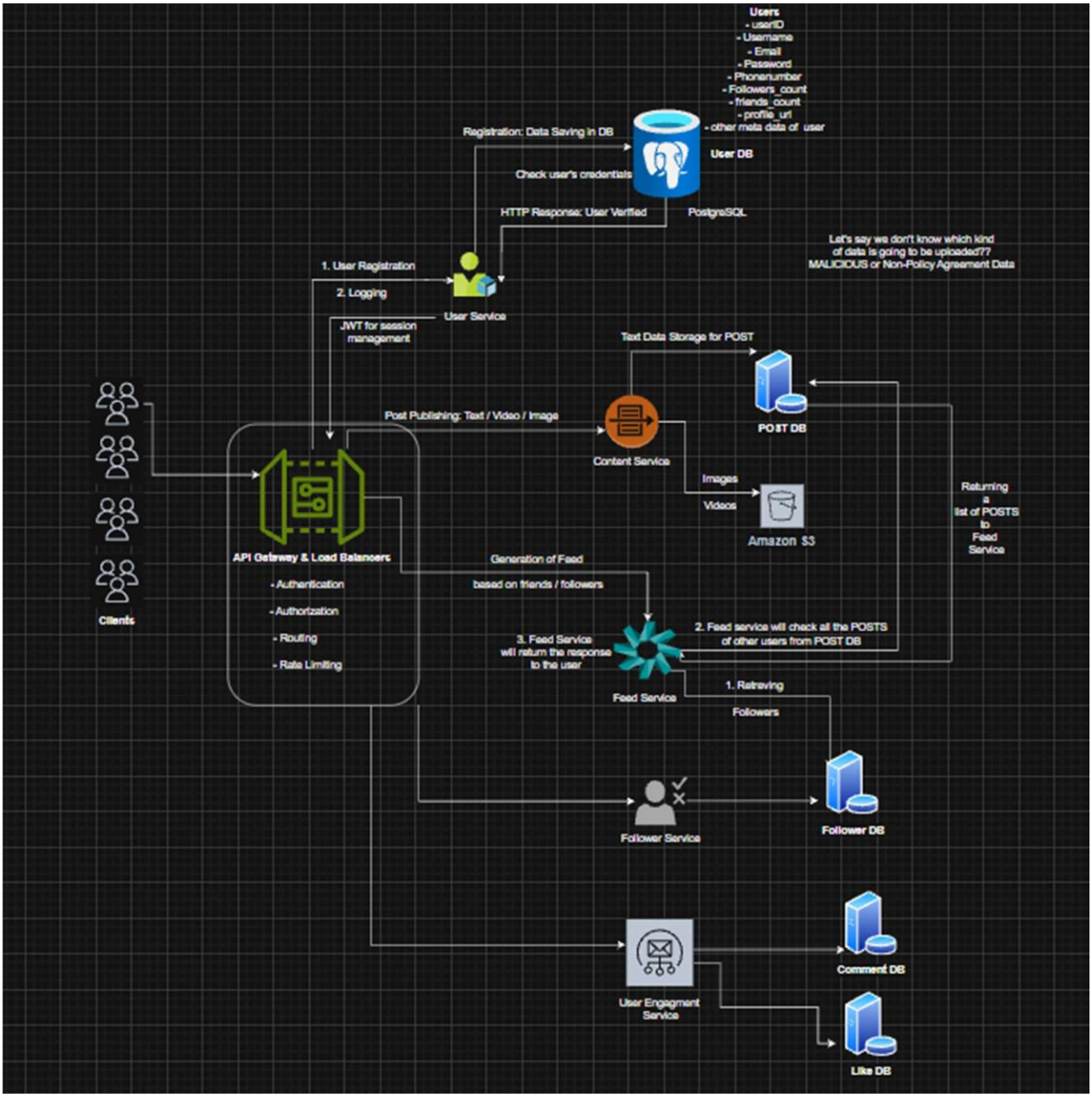
C. User Interactions

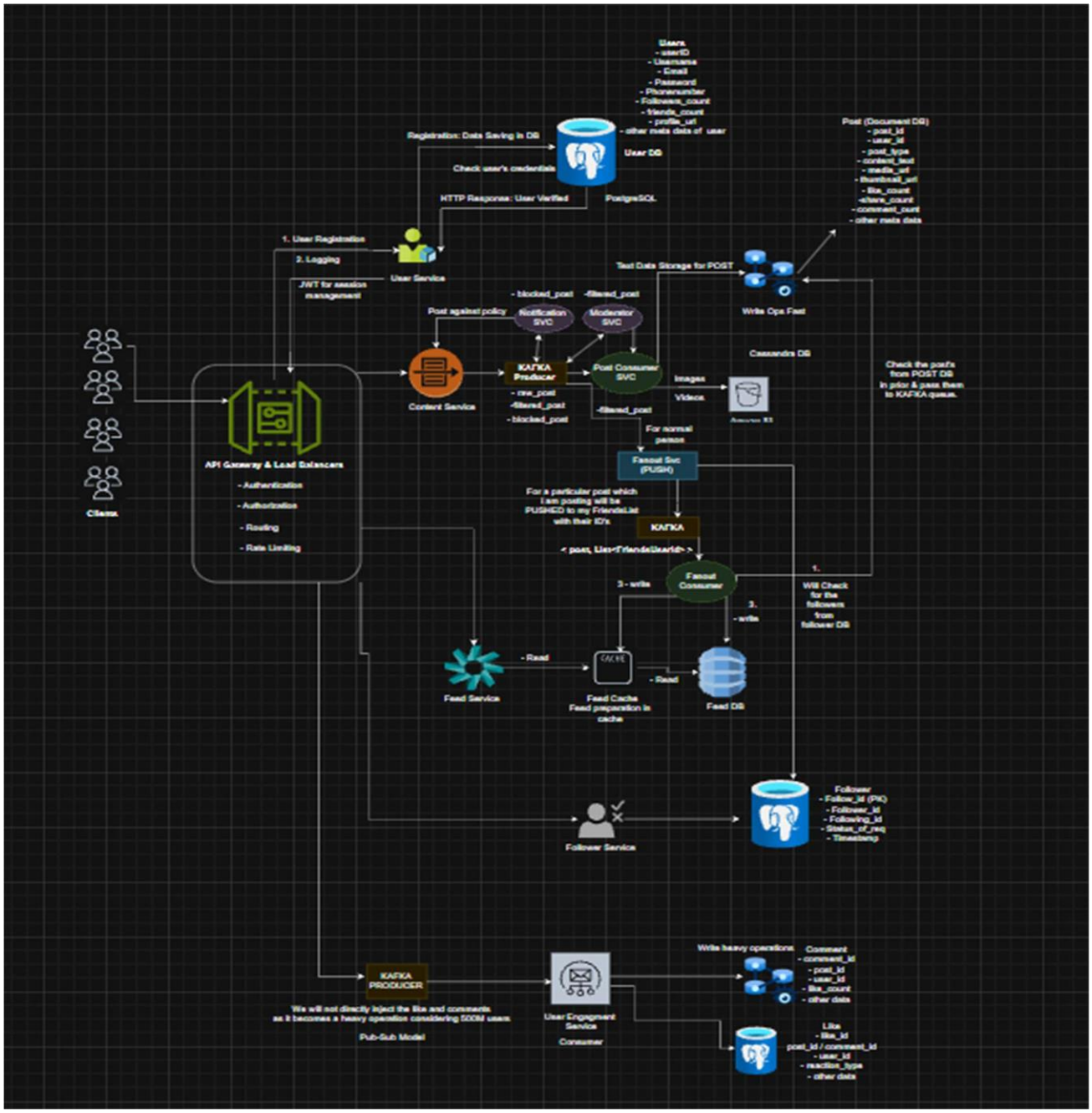
1. POST /api/posts/{post_id}/like
2. DELETE /api/posts/{post_id}/unlike
3. POST /api/posts/{post_id}/comments
4. GET /api/posts/{post_id}/comments
5. PUT /api/comments/{comment_id}
6. DELETE /api/comments/{post_id}/{comment_id}
7. POST /api/users/{user_id}/follow
8. DELETE /api/users/{user_id}/unfollow

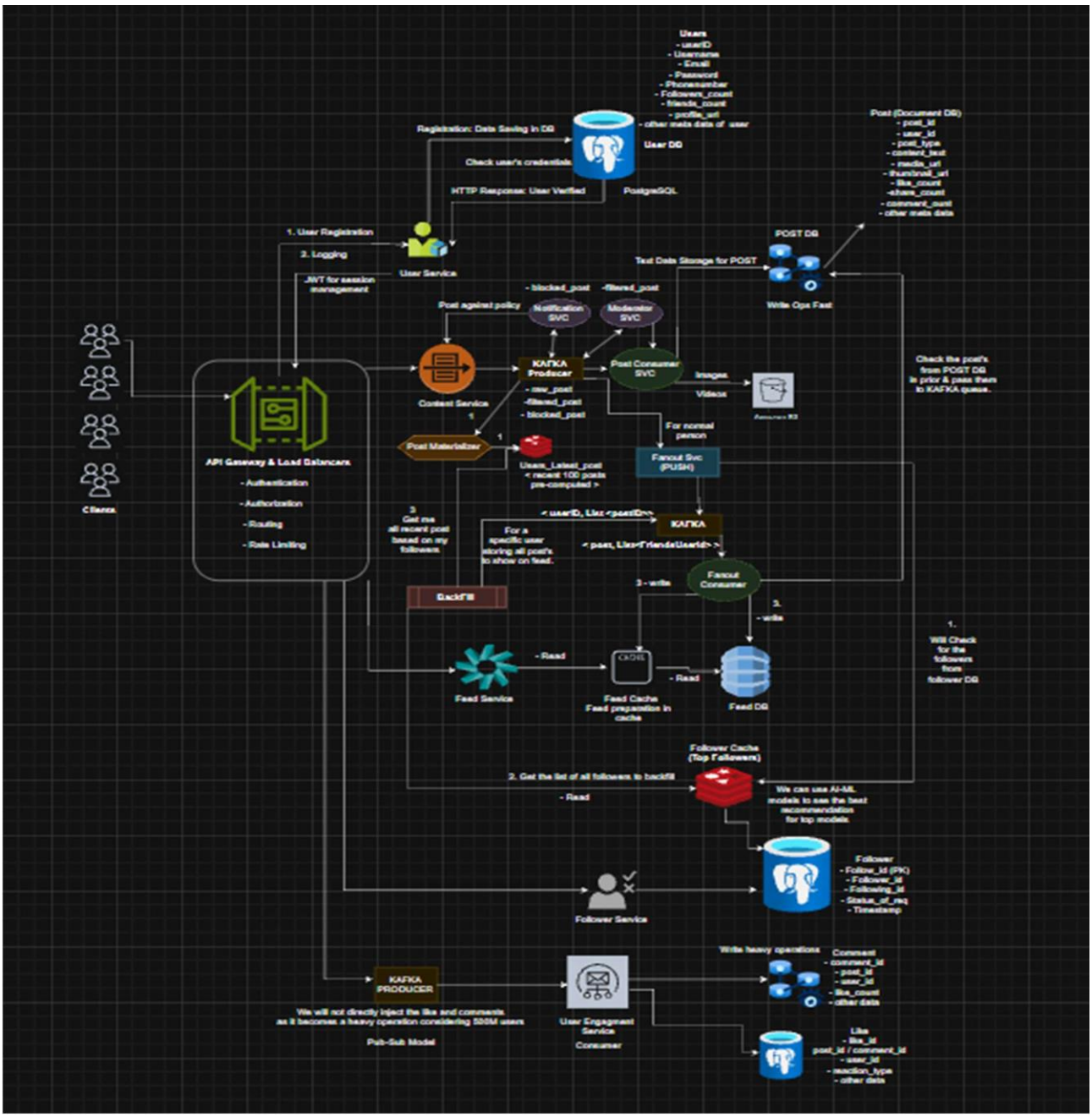
5- High-Level Design


Registration: Data Saving in DB
User DB
Check user's credentials
HTTP Response: User Verified
1. User Registration
2. Logging
User Service
JWT for session management
Text Data Storage for POST
POST DB
Post Publishing: Text / Video / Image
Content Service
Images
Videos
Amazon S3
Returning a list of POSTS to Feed Service
Clients
API Gateway & Load Balancers
- Authentication
- Authorization
- Routing
- Rate Limiting
Generation of Feed
based on friends / followers
3. Feed Service will return the response to the user
2. Feed service will check all the POSTS of other users from POST DB
Feed Service
1. Retreving
Followers
Follower Service
Follower DB
User Engagment Service
Comment DB
Like DB

6- Low- Level Design




Users
- userID
- Username
- Email
- Password
- Phonenumber
- Followers_count
- friends_count
- profile_url
- other meta data of user
User DB
PostgreSQL
Registration: Data Saving in DB
Check user's credentials
HTTP Response: User Verified
Post (Document DB)
- post_id
- user_id
- post_type
- content_text
- media_url
- thumbnail_url
- like_count
- share_count
- comment_count
- other meta data
1. User Registration
2. Logging
User Service
JWT for session management
Test Data Storage for POST
Write Ops Fast
- blocked_post
-filtered_post
Post against policy
Notification SVC
Moderator SVC
Cassandra DB
Content Service
KAFKA Producer
- new_post
-filtered_post
- blocked_post
Post Consumer SVC
Images
Videos
-filtered_post
Check the post's from POST DB in prior & pass them to KAFKA queue.
For normal person
Fanout Svc (PUSH)
Clients
API Gateway & Load Balancers
- Authentication
- Authorization
- Routing
- Rate Limiting
For a particular post which I am posting will be PUSHED to my FriendsList with their ID's
KAFKA
< post, List<FriendsUserId> >
Fanout Consumer
3 - write
1.
Will Check for the followers from follower DB
3.
- write
- Read
- Read
Feed Service
CACHE
Feed Cache
Feed preparation in cache
Feed DB
Follower
- Follow_id (PK)
- Follower_id
- Following_id
- Status_of_req
- Timestamp
Follower Service
KAFKA PRODUCER
We will not directly inject the like and comments as it becomes a heavy operation considering 500M users
Pub-Sub Model
User Engagement Service
Consumer
Write heavy operations
Comment
- comment_id
- post_id
- user_id
- like_count
- other data
Like
- like_id
post_id / comment_id
- user_id
- reaction_type
- other data


Users
- userID
- Username
- Email
- Password
- Phonenumber
- Followers_count
- friends_count
- profile_url
- other meta data of user
User DB
PostgreSQL
Registration: Data Saving in DB
Check user's credentials
HTTP Response: User Verified
Post (Document DB)
- post_id
- user_id
- post_type
- content_text
- media_url
- thumbnail_url
- like_count
-share_count
- comment_count
- other meta data
POST DB
Write Ops Fast
Text Data Storage for POST
1. User Registration
2. Logging
User Service
JWT for session management
- blocked_post
-filtered_post
Post against policy
Notification SVC
Moderator SVC
KAFKA Prodecar
Post Consumer SVC
Images
Videos
Content Service
- new_post
-filtered_post
- blocked_post
Check the post's from POST DB in prior & pass them to KAFKA queue.
For normal person
Post Materializer
Users_Latest_post
< recent 100 posts pre-computed >
Fanout Svc (PUSH)
API Gateway & Load Balancers
- Authentication
- Authorization
- Routing
- Rate Limiting
Clients
3
Get me all recent post based on my followers
For a specific user storing all post's to show on feed.
< userID, List<postID>>
KAFKA
< post, List<FriendsUserId> >
Fanout Consumer
3 - write
BackFill
3.
- write
1.
Will Check for the followers from follower DB
- Read
Feed Service
CACHE
Feed Cache
Feed pagenation in cache
- Read
Feed DB
Follower Cache
(Top Followers)
2. Get the list of all followers to backfill
- Read
We can use AI-ML models to see the best recommendation for top models
Follower
- Follow_id (PK)
- Follower_id
- Following_id
- Status_of_req
- Timestamp
Follower Service
Write heavy operations
Comment
- comment_id
- post_id
- user_id
- like_count
- other data
KAFKA PRODUCER
We will not directly inject the like and comments as it becomes a heavy operation considering 500M users
Pub-Sub Model
User Engagment Service
Consumer
Like
- like_id
post_id / comment_id
- user_id
- reaction_type
- other data

7. Outcome / Result

- Understood the workflow of a social media application system
- Learned to design functional and non-functional requirements.
- Gained knowledge of high-level and low-level system architecture.
- Learned how scalability and availability are achieved in large-scale systems.
- Developed understanding of core APIs for user management, posts, and feed handling.