EXPERIMENT NO: 3

Student Name: LIKHIL N MAIYA
UID: 23BCS11938
Branch: BE-CSE
Section/Group: KRG-1B
Semester: 6th
Date of Performance: 02/02/2026
Subject Name: System Design
Subject Code: 23CSH-314

Aim

To design a Social Media platform similar to Facebook or Instagram that allows users to register, create posts, follow other users, like and comment on posts, and view a personalized feed by identifying functional and non-functional requirements and defining system APIs.

Objectives

- Understand the architecture of a large-scale Social Media system.
- Identify functional requirements such as posting, following, liking, commenting, and feed generation.
- Identify non-functional requirements like scalability, availability, and latency.
- Analyze feed management strategies for high-scale systems.
- Design RESTful APIs for core Social Media operations.

Procedure

1. Studied real-world social media platforms like Instagram and Facebook.
2. Identified core entities such as User, Post, Followers, Feed, Like, and Comment.
3. Analyzed user flow from registration to feed consumption.
4. Collected functional and non-functional requirements.
5. Designed APIs for user onboarding, post creation, follow actions, feed retrieval, and engagement.
6. Studied feed generation techniques like fanout-on-write and fanout-on-read.
7. Analyzed availability vs consistency trade-offs.

Functional Requirements

1. Users should be able to register and login to the platform.
2. Users should be able to create text, image, or video posts.
3. Users should be able to follow and unfollow other users.
4. Users should be able to like and comment on posts.

5. Users should be able to view a personalized feed from followed users.
6. Users should be able to view user profiles.

Core Entities of the System

- User, Post, Followers, Feed, Like, Comment

API Design

1. **User Registration API**: POST /api/users/register
2. **User Login API**: POST /api/users/login
3. **Create Post API**: POST /api/posts
4. **Follow User API**: POST /api/users/{user_id}/follow
5. **Like Post API**: POST /api/posts/{post_id}/like
6. **Comment API**: POST /api/posts/{post_id}/comment
7. **Get Feed API**: GET /api/feed?limit=20

Non-Functional Requirements

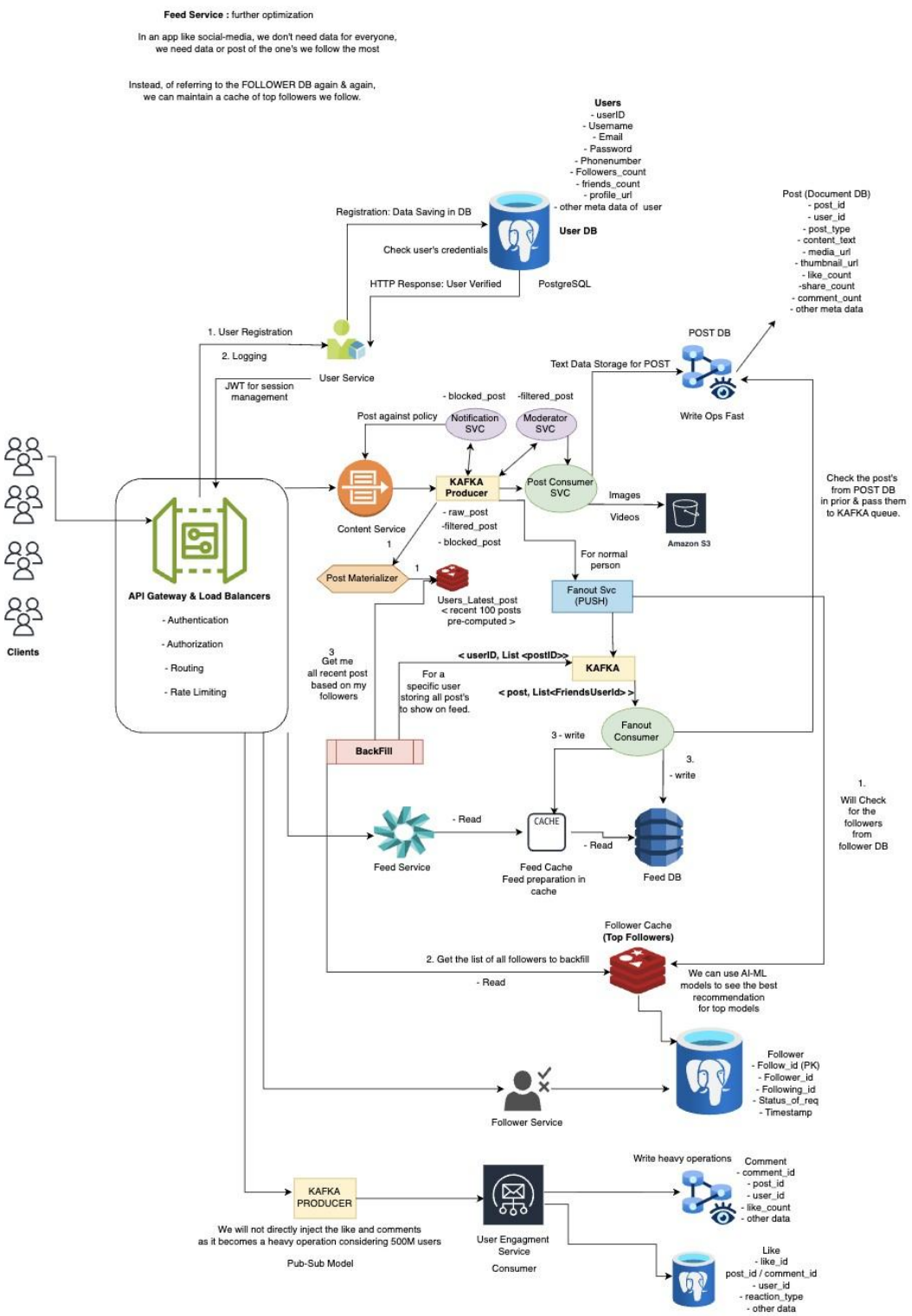
1. Support up to 500 million daily active users.
2. High availability for feed and post viewing.
3. Eventual consistency for likes and comments.
4. Post upload latency under 500 ms. 5. Horizontally scalable architecture.

High Level Design (HLD)

The system consists of Client, API Gateway, Backend Services (User, Content, Feed, Follower, Engagement), Message Queue (Kafka), Cache, and Databases. Media content is stored in object storage.

Outcome

- Designed a scalable Social Media system.
- Identified functional and non-functional requirements.
- Designed RESTful APIs for core operations.
- Understood feed generation strategies.


Feed Service : further optimization
In an app like social-media, we don't need data for everyone, we need data or post of the one's we follow the most
Instead, of referring to the FOLLOWER DB again & again, we can maintain a cache of top followers we follow.
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_ount
- other meta data
1. User Registration
2. Logging
User Service
JWT for session management
POST DB
Text Data Storage for POST
Write Ops Fast
- blocked_post
-filtered_post
Post against policy
Notification SVC
Moderator SVC
Content Service
KAFKA Producer
Post Consumer SVC
Images
Videos
Amazon S3
Check the post's from POST DB in prior & pass them to KAFKA queue.
- raw_post
-filtered_post
- blocked_post
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
< userID, List <postID>>
KAFKA
For a specific user storing all post's to show on feed.
< post, List<FriendsUserId> >
Fanout Consumer
BackFill
3 - write
3.
- write
1.
Will Check for the followers from follower DB
- Read
CACHE
- Read
Feed Service
Feed Cache
Feed preparation in cache
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