Experiment 3

Student Name: Gaurav Singh **UID: 23BCS12992**

Branch: CSE **Section/Group: KRG 1-A**

Semester: 6th **Date of Performance: 28/01/2026**

Subject Name: System Design **Subject Code: 23CSH-314**

1. **Aim**: To design and analyze a **scalable Social Media platform** similar to **Facebook / Instagram** that allows users to register, create posts, follow other users, like and comment on posts, and view personalized feeds while ensuring **high availability, scalability, consistency trade-offs, and low latency**.

2. Objective:

- To identify functional and non-functional requirements of a social media system.
- To design a microservices-based scalable architecture for a large-scale social platform.
- To understand CAP theorem trade-offs in social media systems.
- To design a High-Level Design (HLD) for user management, post creation, feed generation, likes, and comments.
- To understand the role of Kafka, caching, fan-out strategy, and object storage (S3) in realtime systems.
- To analyze how high availability is prioritized over strict consistency in social platforms.

3. Tools Used:

- **Draw.io** – Designing system architecture (HLD).
- **PostgreSQL** – Relational database for user and follower data.
- **Document DB (NoSQL)** – Post, comment, and feed storage.
- **Apache Kafka** – Event streaming and asynchronous processing.
- **Redis** – Caching feeds and follower data.
- **Amazon S3** – Media storage for images and videos.

4. System Requirements:

A. Functional Requirements:

1) User should be able to register and login to the application.

2) User should be able to create posts (text / image / video).

3) User should be able to follow other users or send friend requests.

4) User should be able to like and comment on posts.

5) User should be able to view feed consisting of posts from users they follow.

6) System should support real-time user engagement (likes, comments).

B. Non-Functional Requirements:

1) **Scalability**

- Target Scale: 500 Million Daily Active Users (DAU)
- System must handle millions of concurrent users.

2) **Availability & Consistency**

- High Availability is prioritized over Strong Consistency
- Reason:
- If the platform is unavailable, the system becomes useless.
- Slight delay in post propagation (eventual consistency) is acceptable.

Example:
If Instagram is down for 1 hour → Critical issue
If a post reaches followers after 500 ms → Acceptable

Hence:
Availability >>> Consistency

3) **Latency**

- Post publishing latency ≤ 500 ms
- Feed loading latency should be minimal for smooth user experience.

4) **Reliability**

- System should tolerate failures and recover automatically.

5. High Level Design (HLD):

The system follows a microservices-based architecture:

API Gateway

les routing, authentication, authorization, and rate limiting.

User Service

• Manages user registration, login, JWT-based authentication, and profiles.

Follower Service

• Manages follow / unfollow relationships between users.

Content Service

• Handles post creation, validation, and metadata management.

Media Service

• Uploads and stores images/videos in Amazon S3.

Feed Service

• Generates personalized user feeds using fan-out strategy and caching.

User Engagement Service

• Handles likes and comments using Kafka (Pub-Sub model).

Kafka

• Asynchronous processing of post events, likes, comments, and feed updates.

Cache (Redis)

• Stores frequently accessed feeds and top followers for fast access.

6. Feed Generation Strategy:

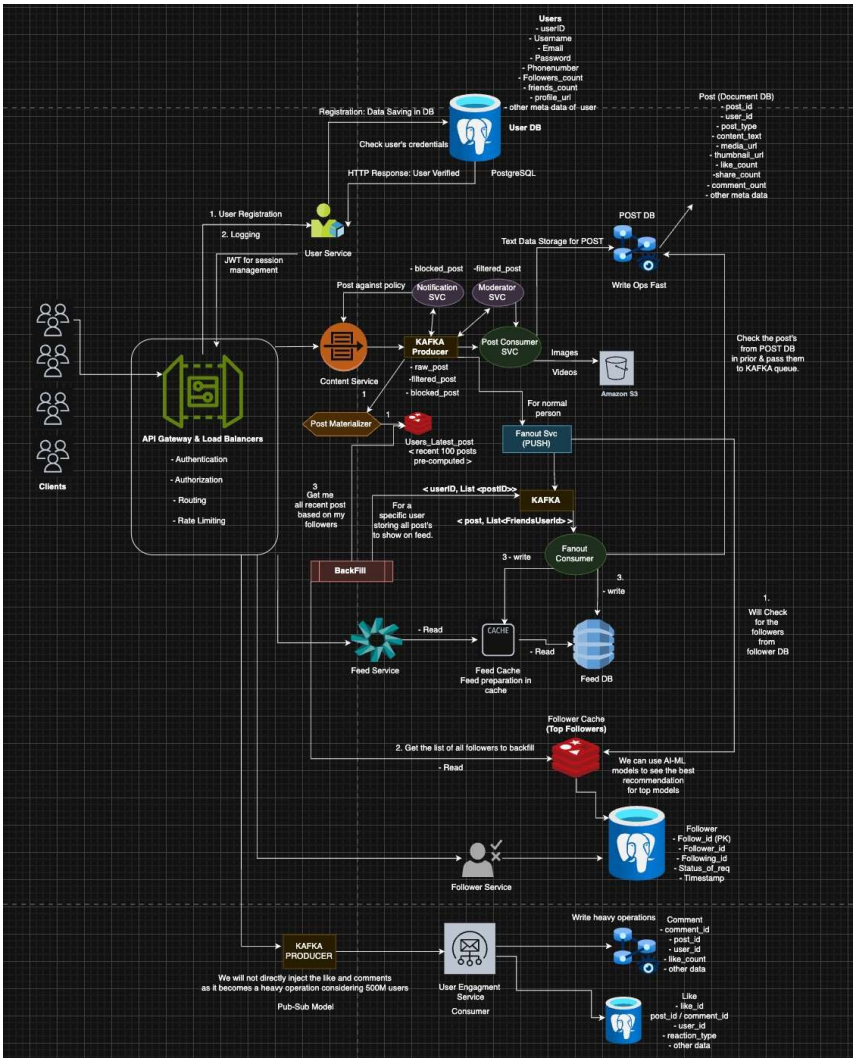
Fan-Out on Write

• When a user posts content, the post is pushed to followers' feeds asynchronously using Kafka.
• Suitable for users with limited followers.

Fan-Out on Read

celebrities with millions of followers, feed is generated at read time to avoid heavy writes. **Hybrid Approach**

- System intelligently chooses between fan-out on write and read.



7. Scalability Solution:

- Horizontal scaling of microservices.
- API Gateway acts as load balancer.
- Kafka decouples services and smoothens traffic spikes.
- Redis caching reduces database load.
- Object storage (S3) handles heavy media traffic efficiently.

8. Learning Outcomes (What I Have Learnt):

- Learned how real-world **social media platforms are designed**.

- Understood **functional vs non-functional requirements** clearly.
- Gained strong understanding of **CAP theorem trade-offs**.
- Learned how **Kafka, caching, and fan-out strategies** are used at scale.
- Understood why **availability is more important than consistency** in social systems.
- Learned how to design **low-latency, highly scalable distributed systems**.