Experiment-3

Student Name: Ayush Tiwari
UID: 23BCS11366
Branch: CSE
Section/Group: KRG-3B
Semester: 6th
Date of Performance: 28/01/26
Subject Name: System Design
Subject Code: 23CSH-314

1. **AIM :** To design a Social Media Platform similar to Facebook / Instagram
   Description:A Social Media platform is a platform which allows users to share photos, videos, and text with their friends and followers. To design a scalable and highly available **Social Media System** where users can register, create posts, follow others, and interact with content through likes and comments.

2. Objectives

- To understand the working of a large-scale social media system
- To design functional and non-functional requirements
- To identify core system entities
- To design API endpoints for communication
- To ensure scalability, availability, and low latency

3. Tools Required

- System Design Tools (Draw.io, Lucidchart, etc.)
- Programming Language (Java / Python / JavaScript)
- Database (MySQL / MongoDB)
- API Testing Tool (Postman)
- Web Browser
- IDE (VS Code, IntelliJ, etc.)

4. SYSTEM DESIGN / SYSTEM SPECIFICATION

4.1 Functional Requirements

1. User should be able to register and login.
2. User should be able to create posts (text / image / video).
3. Users should be able to follow each other / send friend requests.
4. Users should be able to like and comment on posts.
5. Users should be able to view feed from people they follow.

4.2 Non-Functional Requirements

1. **Scalability:**
   - System should support 500 million DAU.

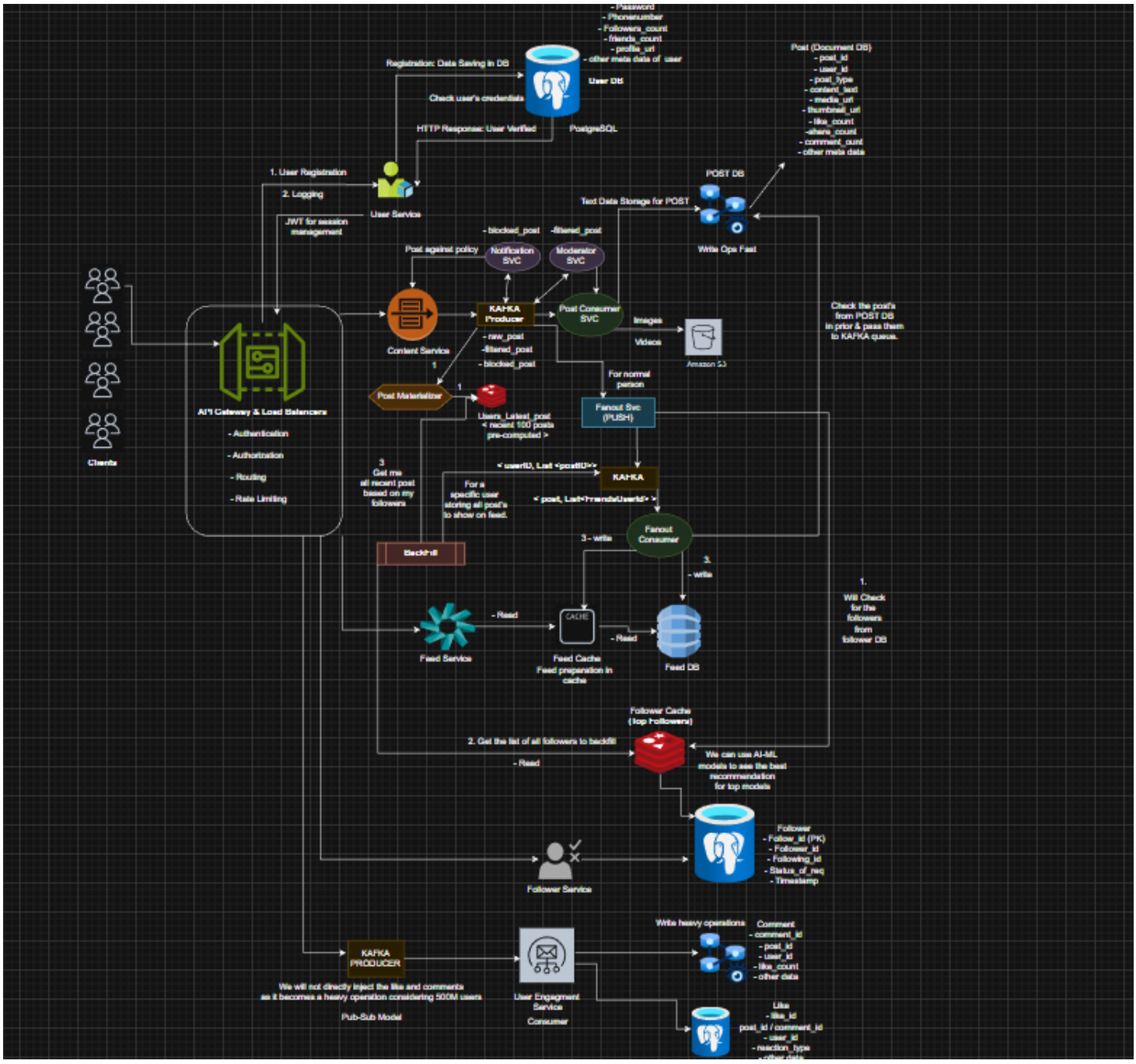
2. **Consistency & Availability (CAP Theorem):**
    - This system needs to be highly available first, then consistent.
    - Reason: If the system is not operational, it becomes useless.
    - Example:
        - If Instagram is down for 1 hour → huge issue.
        - But if a post takes 500ms to reach followers → acceptable.
    - Hence:
      Availability >>> Consistency
3. **Latency:**
    - **Uploading / publishing post should take ≤ 500 ms.**

4.3 Core Entities of the System

- User
- Post
- Comment
- Like
- Follow / Friend Request
- Feed
- Media (Image/Video)

5. **HLD(High Level Design):**

We have to follow a distributed / micro-services approach not the monolithic one.



Overview

This system represents a large scale social media platform similar to Instagram or Facebook. It allows users to register, log in, create posts, follow others, like and comment on posts, and view a personalized feed.

User Access
Users interact with the system through mobile or web applications. All requests are first sent to the API Gateway.

API Gateway
The API Gateway acts as the main entry point to the system. It is responsible for authentication, authorization, routing requests to the correct service, rate limiting to prevent abuse, and load balancing to distribute traffic evenly.

User Service and User Database
User related operations such as registration and login are handled by the User Service. All user information is stored in the User Database. This database contains details such as user id, username, password, profile information, followers count, and following count.

Content Service and Post Creation
When a user creates a post, the request is sent to the Content Service. The post is checked by the Moderation Service to ensure it follows platform rules. At the same time, the Notification Service prepares alerts for followers.

Kafka Messaging System
After moderation, the post is sent to Kafka. Kafka acts as a message broker that allows asynchronous communication between services. This ensures the system can handle a large number of users and posts efficiently.

Post Storage
Post details such as caption, user id, and media links are stored in the Post Database, which is a document based database. Images and videos are stored in cloud storage, and only their links are saved in the database.

Fanout Service
The Fanout Service reads new posts from Kafka and pushes them to the feeds of all followers. It gets the list of followers from the Follower Database and Follower Cache.

Feed Service, Feed Database, and Cache
Each user has a feed that stores posts from people they follow. The Feed Database stores feed data permanently. The Feed Cache stores recent feed data for fast access. When a

user opens the app, the Feed Service first checks the cache. If the data is not found, it reads from the database.

Follower Service
The Follower Service manages follow and unfollow operations. It stores follower and following relationships in the Follower Database and keeps frequently used data in the Follower Cache.

Like and Comment System
The Like Database stores which users liked which posts. The Comment Database stores all comments on posts along with user and time details.

Notification Service
The Notification Service listens to Kafka events such as new posts, likes, and comments. It sends push notifications and in app alerts to users.

System Properties
The system is designed to be highly available and scalable. It prefers availability over strict consistency so that users can always access the platform even if some data is slightly delayed.

7. Learning Outcomes

- Understand the design and architecture of a scalable E-commerce platform.
- Gain hands-on experience with Apache Kafka for real-time data streaming.
- Learn to implement fast and efficient search using Elasticsearch.
- Understand Change Data Capture (CDC) pipelines for real-time data synchronization.
- Develop skills in integrating distributed systems for high availability and scalability.