



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

## Experiment - 3

**Student Name:** Arpit Anand

**Branch:** BE-CSE

**Semester:** 6<sup>th</sup>

**Subject Name:** System Design

**UID:** 23BCS12710

**Section/Group:** KRG\_1B

**Date of Performance:** 29/01/26

**Subject Code:** 23CSH-314

### **Aim:**

To design a **Social Media Platform** that allows users to register, login, create posts, follow other users, and interact with posts through likes and comments, ensuring high availability, scalability, and low latency.

### **Objectives:**

1. To understand the working of a Social Media system
2. To identify **functional requirements** of the system
3. To identify **non-functional requirements** such as performance and scalability
4. To design a high-level system flow using **draw.io**
5. To understand core entities involved in the platform

### **Procedure-**

1. Identify functional requirements of a social media platform.
2. Define non-functional requirements such as scalability, latency, and availability.
3. Analyze CAP theorem trade-offs for social media systems.
4. Identify core entities required for system implementation.
5. Design the system architecture using Draw.io.
6. Validate the design against real-world social media behavior.

### **Functional Requirements -**

1. Users should be able to **register and login** to the application.
2. Users should be able to **create posts** (text / image / video).
3. Users should be able to **follow other users** or send friend requests.
4. Users should be able to **like and comment** on posts.
5. Users should be able to **view a feed** consisting of posts from users they follow.



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

## Non-functional Requirements

### A. Scalability

- System should support up to 500 million daily active users (DAU).

### B. Consistency and Availability

- The system prioritizes high availability over strong consistency.
- Temporary delays in post propagation are acceptable.

#### Justification:

If the application is unavailable during peak time, it leads to a major business loss.

Example:

If Instagram is down for 1 hour → **critical issue**

If a post reaches followers in 500 ms instead of instantly → **acceptable** Hence,

**Availability >>> Consistency**

### C. Latency

- Post upload and publish latency should be around **500 ms**.

## Outcome / Result -

A complete high-level design of a social media platform was successfully created, identifying its functional requirements, non-functional constraints, core entities, and feed management strategy.

The diagram illustrates a complex system architecture for a social media application, featuring a grid background. The components and their interactions are as follows:

- Clients:** Represented by icons of people, they interact with the **API Gateway & Load Balancers**.
- API Gateway & Load Balancers:** Handles requests and distributes them to various services. It includes:
  - Authentication
  - Authorization
  - Routing
  - Rate Limiting
- User Service:**
  - Handles **1. User Registration** and **2. Logging**.
  - Manages **JWT for session management**.
  - Interacts with **PostgresSQL** for **Registration: Data Saving in DB** and **Check user's credentials**.
  - Returns **HTTP Response: User Verified**.
- Content Service:**
  - Handles **Post against policy** and **Post Materializer**.
  - Interacts with **Notification Svc** and **Moderator Svc**.
  - Produces **raw\_post**, **filtered\_post**, and **blocked\_post** to **KAFKA Producer**.
- KAFKA Producer:**
  - Produces **raw\_post**, **filtered\_post**, and **blocked\_post**.
  - Interacts with **Post Consumer Svc** and **Fanout Svc (PUSH)**.
- Post Consumer Svc:**
  - Consumes **raw\_post**, **filtered\_post**, and **blocked\_post**.
  - Interacts with **Images** and **Videos** (stored in **Amazon S3**).
- Fanout Svc (PUSH):**
  - Handles **For normal person** and **Users Latest post < recent 100 posts pre-computed >**.
  - Interacts with **KAFKA** and **Fanout Consumer**.
- KAFKA:**
  - Handles **< userID, List < postID > >** and **< post, List < Friends UserID > >**.
  - Interacts with **Fanout Consumer** and **Feed DB**.
- Fanout Consumer:**
  - Handles **3- write** and **3- write** operations.
  - Interacts with **Feed DB** and **Feed Service**.
- Feed Service:**
  - Handles **Read** operations.
  - Interacts with **Feed Cache** and **Feed DB**.
- Feed Cache:**
  - Handles **Feed Cache Feed preparation in cache**.
  - Interacts with **Feed DB** and **Feed Service**.
- Feed DB:**
  - Handles **Read** operations.
  - Interacts with **Feed Service** and **Feed Cache**.
- Post (Document DB):**
  - Stores **post\_id**, **user\_id**, **post\_type**, **content\_text**, **media\_url**, **thumbnail\_url**, **like\_count**, **share\_count**, **comment\_count**, and **other meta data**.
  - Interacts with **Write Ops Fast** and **Check the post's in prior & pass them to KAFKA queue.**
- Backfill:**
  - Handles **3 Get me all recent post based on my followers** and **For a specific user storing all post's to show on feed.**
  - Interacts with **Feed Service** and **Feed DB**.
- Other Services:**
  - PostgresSQL:** Used for **Registration: Data Saving in DB** and **Check user's credentials**.
  - Amazon S3:** Used for **Images** and **Videos**.
  - Write Ops Fast:** Used for **Post (Document DB)**.
  - Check the post's in prior & pass them to KAFKA queue.**
  - Will Check for the followers from follower DB**

