



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

## Experiment - 3

**Student Name:** Anshika Singh

**UID:** 23BCS11326

**Branch:** BE-CSE

**Section/Group:** KRG\_2A

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

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

**Subject Name:** System Design

**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.

**Users**

- userID
- Username
- Email
- Password
- Phonenumber
- Followers\_count
- Friends\_count
- profile\_url
- other meta data of user

**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**

JWT for session management

**API Gateway & Load Balancers**

- Authentication
- Authorization
- Routing
- Rate Limiting

**Clienta**

**User Service**

Registration: Data Saving in DB

Check user's credentials

HTTP Response: User Verified

**PostgresSQL**

**Content Service**

Post against policy

**Notification SVC**

**Moderator SVC**

**KAFKA Producer**

- blocked\_post
- filtered\_post
- raw\_post
- filtered\_post
- blocked\_post

**Post Materializer**

1

**Users Latest post**

< recent 100 posts pre-computed >

**Fanout Svc (PUSH)**

For normal person

**Images**

**Videos**

**Amazon S3**

**KAFKA**

< userID, List <postID>

< post, List<Friends/UserId>

**Fanout Consumer**

3. write

3. write

**BackFill**

**Feed Service**

- Read

**CACHE**

Feed Cache Feed preparation in cache

**Feed DB**

- Read

**Followers Cache (Top Followers)**

1. Will Check for the followers from follower DB

3. Get me all recent post based on my followers

For a specific user storing all post's to show on feed.

Check the post's from POST DB in prior & pass them to KAFKA queue.

**POST DB**

**Write Ops Fast**

