



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Experiment - 3

Student Name: Jetavya Singh Naruka

UID: 23BCS11171

Branch: BE-CSE

Section/Group: KRG_3B

Semester: 6th

Date of Performance: 04/02/26

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.

The diagram illustrates a complex system architecture for a social media feed service, involving multiple services, databases, and data flows.

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

Services and Components:

- Client**: Represented by a group of people icon.
- API Gateway & Load Balancers**:
 - Authentication
 - Authorization
 - Routing
 - Rate Limiting
- User Service**:
 - 1. User Registration
 - 2. Logging
 - JWT for session management
- Content Service**:
 - Post against policy
 - Notification Svc
 - Moderator Svc
- Post Materializer**:
 - 1. Get me all recent post based on my followers
- Kafka Producer**:
 - raw_post
 - filtered_post
 - blocked_post
- Post Consumer Svc**:
 - Images
 - Videos
 - For normal person
- Post (Document) DB**:
 - Check the post's from POST DB in prior & pass them to KAFKA queue.
- Text Data Storage for POST**:
 - Write Ops Fast
- Amazon S3**:
 - Images
 - Videos
- Fanout Svc (PUSH)**:
 - For normal person
- Fanout Consumer**:
 - 3. write
 - 3. write
- Feed Cache**:
 - Feed Cache Feed preparation in cache
- Feed Service**:
 - Read
- Follower Cache (Top Followers)**:
 - 1. Will Check for the followers from follower DB

Data Flows and Messages:

- Registration: Data Saving in DB**: User Service to User DB (PostgreSQL).
- Check user's credentials**: User Service to PostgreSQL.
- HTTP Response: User Verified**: PostgreSQL to User Service.
- Post against policy**: Content Service to Notification Svc and Moderator Svc.
- Post Materializer** sends **1** to **Kafka Producer**.
- Kafka Producer** sends **raw_post**, **filtered_post**, and **blocked_post** to **Post Consumer Svc**.
- Post Consumer Svc** sends **Images** and **Videos** to **Amazon S3**.
- Post Consumer Svc** sends **For normal person** to **Fanout Svc (PUSH)**.
- Fanout Svc (PUSH)** sends **3. write** to **Fanout Consumer**.
- Fanout Consumer** sends **3. write** to **Feed Cache**.
- Feed Cache** sends **- Read** to **Feed Service**.
- Feed Service** sends **- Read** to **Follower Cache (Top Followers)**.
- Follower Cache (Top Followers)** sends **1. Will Check for the followers from follower DB** to **Post (Document) DB**.
- Post (Document) DB** sends **Check the post's from POST DB in prior & pass them to KAFKA queue.** to **Kafka Producer**.

