CHANDIGARH UNIVERSITY

DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Experiment - 3

Student Name: Abhishek Sharma **UID:** 23BCS11614

Branch: BE-CSE **Section/Group:** KRG_2B

Semester: 6th **Date of Performance:** 28/1/2026

Subject Name: System Design

Subject Code: 23CSH-314

Aim –

To design a social media platfrom similar to Facebook or Instagram

Objectives –

1. To design a social media platform similar to Facebook or Instagram.
2. To identify functional and non-functional requirements of a large-scale social media system.
3. To define core entities involved in a social media application.
4. To design RESTful API endpoints for user onboarding, posts, and interactions.

Procedure -

1. Study the working of popular social media platforms such as Facebook and Instagram.

2. Identify the main features required for a social media application.

3. List the functional requirements such as user registration, posting content, following users, and interacting with posts.

4. Analyze non-functional requirements including scalability, availability, consistency, and latency.

5. Identify the core entities required for the system such as users, posts, followers, feeds, likes, and comments.

6. Design API endpoints for:

   - User onboarding (registration, login, profile management)
   - Post creation, retrieval, update, deletion, and feed pagination
   - User interactions such as likes, comments, follow and unfollow

7. Document all objectives, requirements, entities, and APIs in a structured format.

Functional Requirements -

1. Client should be able to register and login to the application.
2. Client should be able to create post (text / image / videos)
3. Client should be able to follow each other (or send friend requests)
4. Client should be able to like or comment on the post
5. Client should be able to view the feed of post from users they follow

Non-Functional Requirements –

Scalability: 500M DAU

Consistency & Availability:
In this case -> Social Media

1. We need our application to be HIGHLY available, then consistent
   Reason: If it is not operational / Functional when it was required, then there is no meaning of developing the application.
   Eg: Suppose, Instagram is down for 1 hour -> Big Picture issue
   But, on the other hand, Instagram was up and running, and you posted something which took 500 ms to reach you followers -> is it a issue?? Compared to availability
   Hence, Availability >>>> Consistency

Latency: (Uploading speed of publishing post): 500ms to upload post

Api Endpoints-

A. User On-boarding API's

1. User Registeration: POST API CALL
   : POST / api / users / register_user
2. User Login: POST API CALL:
   POST / api / users / login
3. User Data Display: GET API CALL:
   GET / api / users / {user_id} / profile
4. User Data Update: PUT API CALL:
   PUT / api / users / {user_id} / profile

B. User Post's

1. POST / api / user_id / posts
2. GET / api / posts / {post_id}
3. PUT / api / posts / {post_id}
4. DELETE / api / posts / {post_id}
5. GET / api / posts / feed / limit = {limit} & offset = {offset} : PAGINATION
6. GET / api / users / {user_id} / posts: PAGINATION

C. User Interactions

1. POST / api / posts / {post_id} / like
2. DELETE / api / posts / {post_id} / unlike
3. POST / api / posts / {post_id} / comments
4. GET / api / posts / {post_id} / comments
5. PUT / api / comments / {comment_id}
6. DELETE / api / comments /{post_id}/ {comment_id}
7. POST / api / users / {user_id} / follow
8. DELETE / api / users / {user_id} / unfollow

Core-entities of System

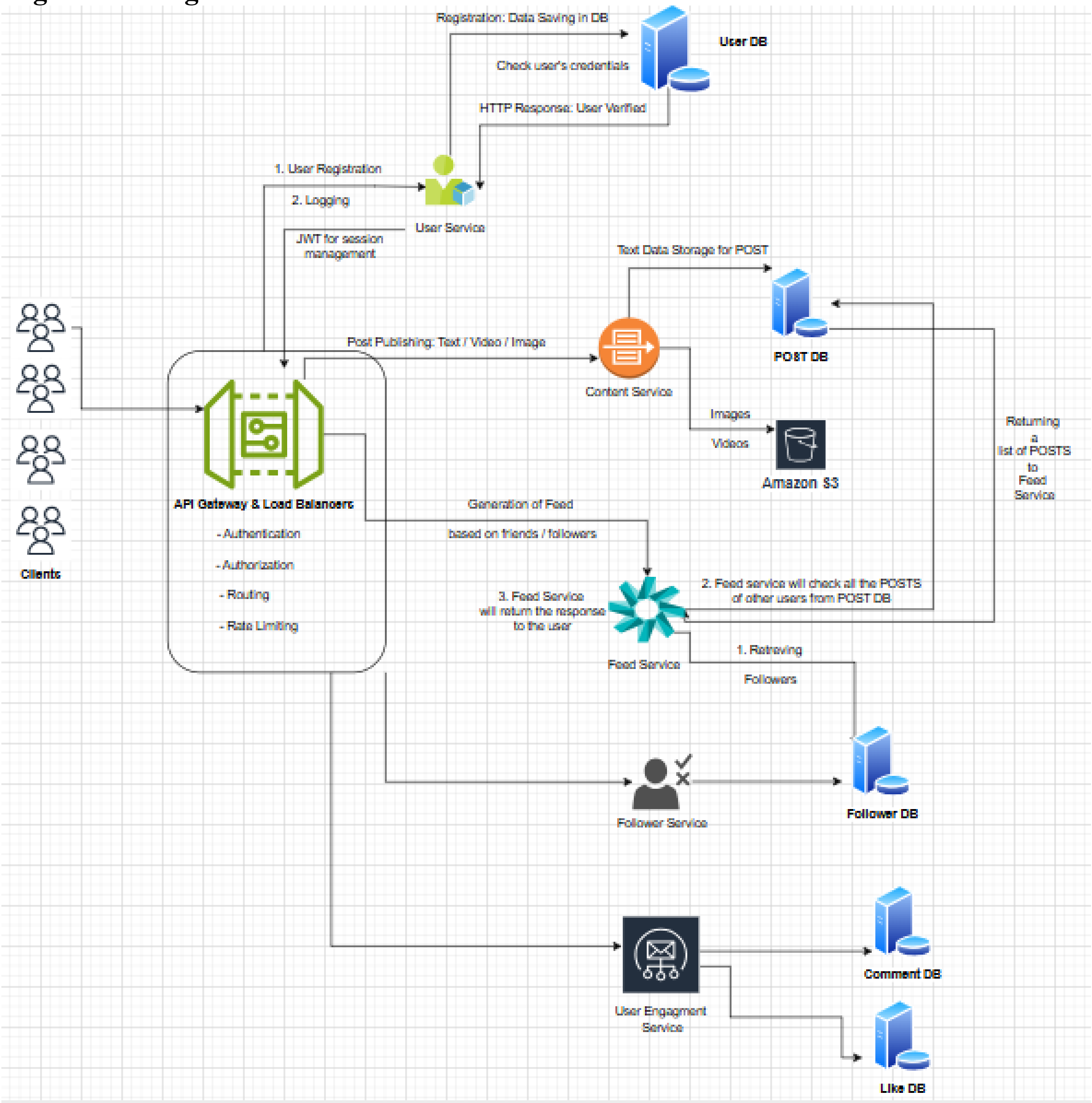
1. Users
2. Followers
3. Post
4. Feeds
5. Like & Comment

Outcome / Result -

1. Successfully designed a high-level architecture for a social media platform.
2. Identified key functional requirements of the system.
3. Analyzed important non-functional requirements focusing on scalability, availability, and latency.
4. Defined core entities required for implementing the application.
5. Designed RESTful API endpoints to support user management, posts, and interactions.
6. Gained understanding of system design considerations for large-scale social media applications.

Required System Design –

High level design:


Registration: Data Saving in DB
User DB
Check user's credentials
HTTP Response: User Verified
1. User Registration
2. Logging
User Service
JWT for session management
Text Data Storage for POST
POST DB
Post Publishing: Text / Video / Image
Content Service
Images
Videos
Amazon S3
Returning a list of POSTS to Feed Service
Clients
API Gateway & Load Balancers
- Authentication
- Authorization
- Routing
- Rate Limiting
Generation of Feed
based on friends / followers
2. Feed service will check all the POSTS of other users from POST DB
3. Feed Service will return the response to the user
Feed Service
1. Retreving
Followers
Follower Service
Follower DB
Comment DB
User Engagment Service
Like DB