Designing Instagram

CISC 3140 – Design & Implementation of Large-Scale Applications

Fall 2025

Instructor: Professor Samanta

Authors:
Muhammad Ali
Maakham Javed
Tehrim Iqbal
Sehar Ali

December 14, 2025

Understanding & Framing the Problem

System Overview

Instagram is a widely used social media platform that enables users to upload, share, and interact with photos, videos, ephemeral stories, and short-form video content (Reels). The platform provides social interactions (likes, comments, shares), direct messaging, and a highly personalized content feed driven by recommendation algorithms. The design objective is to support massive scale (hundreds of millions to a billion users), ensure low latency user experiences, and guarantee high availability and data durability for media content.

Primary Users and Personas

- **Content Creators:** Actively upload photos, videos and reels; require analytics and monetization features.
- **Casual Users:** Post occasionally, consume content frequently; require seamless feed expe- rience.
- **Passive Viewers:** Rarely post, primarily browse; require low-latency content delivery.
- **Influencers and Businesses:** Require advanced analytics, advertising tools, and audience- building features.
- **Advertisers:** Need campaign management and performance reporting.
- **Administrators and Moderators:** Maintain safety via reporting tools, automated content filtering, and human review.

Main User Goals

Users engage with the platform to accomplish several core objectives that guide the design of both frontend user experiences and backend systems.

- **Share Visual Content:** Users upload and publish photos, videos, and stories to express creativity or share life moments.
- **Discover New Content and People:** Through personalized recommendations and search features, users explore creators, trends, and topics.
- **Engage With Others:** Users interact socially through likes, comments, shares, direct messages, saves, and collaborative participation.
- **Grow an Audience:** Creators, influencers, and businesses seek to build a follower base and increase reach.
- **Monetize Activity:** Brands, influencers, and businesses use the platform for

advertising, sponsored content, and promotional activities.

Expected Scale

The system must be designed to operate at a global scale while providing consistent, real-time performance and reliability. The platform is expected to support:

- **Millions of Daily Active Users:** The system must handle extremely high engagement across all features.
- **Worldwide Availability:** Users will access the platform across diverse regions, languages, devices, and network conditions.
- **High Concurrency:** The platform must simultaneously manage user activities such as posting content, streaming videos, browsing feeds, and messaging.
- **Massive Data Volumes:** The system must accommodate extensive media libraries, interaction logs, behavioral data, stories, and real-time analytics.

Similar Systems For Inspiration

Several existing platforms provide architectural insights and serve as references for the sys- tem's features and requirements:

- **Instagram:** The baseline reference for core functionalities such as feeds, profiles, stories, media sharing, and interaction patterns.
- **TikTok:** A leading example of short-form video discovery and advanced, real-time recommendation algorithms.
- **Snapchat:** Inspiration for ephemeral content design, lightweight messaging, and user privacy features.
- **Facebook:** A model for large-scale social networking, identity systems, and community-oriented features.
- **Twitter/X:** Demonstrates real-time engagement, rapid content dissemination, and trending topic mechanisms.

Requirement Analysis

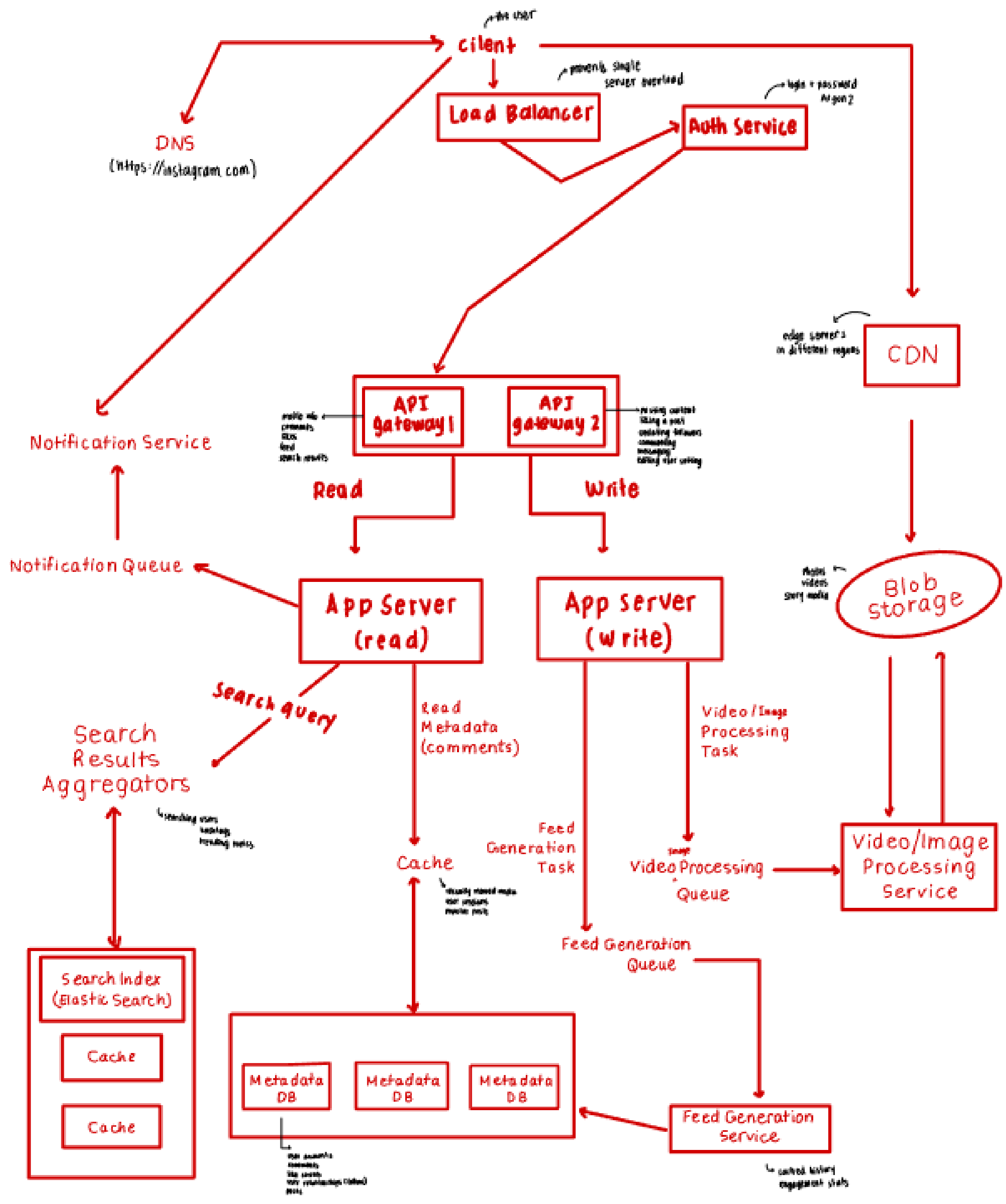
Functional Requirements:

Functional Detail	Description	Priority
User Registration & Authentication	Users register with a username, email, or phone and authenticate securely using password hashing, JWT for sessions, and optional MFA for high-risk flows	Must-have
Profile Management	Profiles must allow users to manage display name, bio, photo, and privacy settings, specifying an account type (personal, creator, business) while linking to analytics and contact information	Must-have
Photo/Video Upload	The system must accept and store common media formats (JPEG, PNG, MP4), transcode them into multiple renditions, attach metadata (captions, tags, geolocation), and support direct uploads to object storage via pre-signed URLs for efficiency	Must-have
Follow/Unfollow	Core features must include Follow/Unfollow functionality, Likes, comments, and messages, and a must-have notification system to alert users of relevant activity	Must-have
Personalized Feed	The core requirement is an infinite-scrolling feed that is ranked by relevance, freshness, and predicted engagement, combining content from followed accounts and recommendations while supporting pagination and cursors	Must-have
Likes, Comments, & Shares	The system must reliably record interactions that modify engagement metrics and propagate them to analytics and ranking features, though certain updates (like likes count) can be eventually consistent under high load	Must-have
Stories (24-hour ephemeral)	Users must be able to post lightweight, ephemeral content (Stories) that expires in 24 hours, is aggressively cached at the edge, and can be optionally saved as highlights.	Should-have
Search & Discovery	Search must cover usernames, hashtags, and captions, and surface recommended accounts and trending content using a specialized index (Elasticsearch) and ML-driven signals.	Should-have
Notifications	The system must deliver timely push and in-app notifications for likes, comments, follows, and messages, requiring integration with APNs and FCM.	Must-have

Non-Functional Requirements

Functional Detail	Description	Priority
Scalability	The system design must scale horizontally using sharding/partitioning to sustain billions of requests and petabytes of media, requiring autoscaling and multi-region deployment	Must-have
Latency	Feed retrieval and primary UI interactions must meet a p95 latency of $200ms$, while video media start time must be minimized using CDN and adaptive streaming	Must-have
Availability	Critical user-facing services must target 99.99% uptime with failover across AZs/regions, requiring the formal definition of Recovery Time Objectives (RTO) and backup policies	Must-have
Durability	Media objects require high durability (119s on object store) and cross-region replication to support disaster recovery	Must-have
Consistency	System may use eventual consistency for social interactions and feeds but must ensure strong consistency for authentication and billing data	Should-have
Security & Privacy	Encrypt data in transit (TLS) and at rest; apply least-privilege access control, rotate secrets, and provide DSAR capabilities	Must-have
Observability	Collect traces, metrics, and structured logs (Prometheus, Jaeger, ELK). SLOs and alerting must be in place	Should-have

Conceptual Design and Component Breakdown

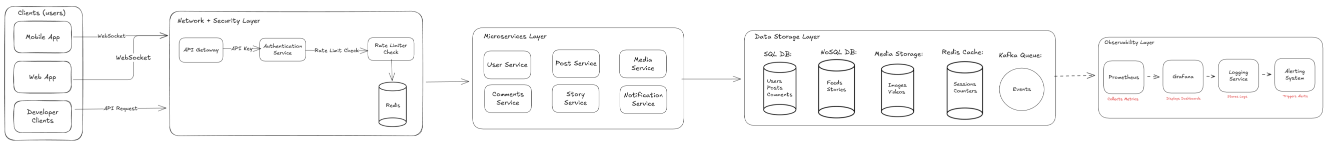


Component Responsibilities

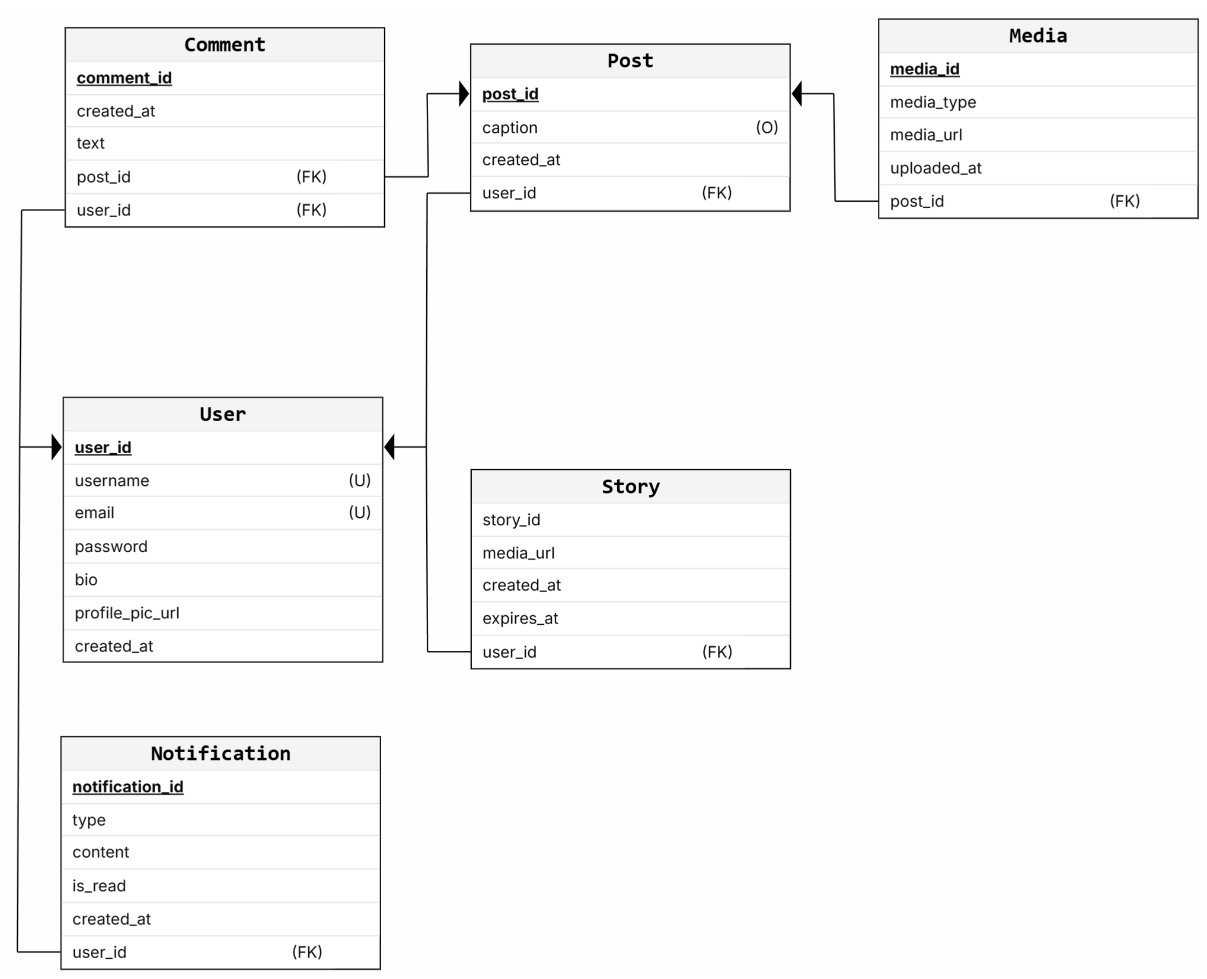
Component	Responsibility
Clients	UI, render feed, upload media, open WebSocket for messages/notifications
Load Balancer	Route traffic, TLS termination, health checking, georouting
API Gateway	Token validation, rate limiting, routing, request shaping, API versioning
Auth Service	Login, token issuance, refresh, MFA, session management
Read Services	Serve low-latency read APIs (feed fetch, profile, search) using cache and timeline DB
Write Services	Handle posts, likes, comments, follow/unfollow; publish events to message bus
Message Bus (Kafka)	Durable event stream used by consumers: feed generator, notification service, analytics
Timeline DB (Cassandra)	Store per-user feed entries, optimized for appends and range scans
Cache (Redis)	Hot cache for feed fragments, sessions, top posts
Media Processor	Transcode media into renditions, generate thumbnails, extract metadata
Blob Storage	Durable object storage for original and transcoded media
CDN	Edge caching and fast delivery of media content globally
Monitoring	Collect metrics, traces, logs, alerting and SLO dashboards

Deep System Architecture

Option	Pros	Cons	Decision
Microservices & Event-drive	Scales each service independently; fault isolation	Operational Complexity	Selected
Monolith	Simple initial build	Poor scalability at Instagram scale	Rejected
Serverless-first	Reduced ops overhead	Issues with long-running tasks (transcading)	Rejected



Data Design



Core Entities & Relationships

The primary entities are User, `Post`, `Comment`, `Media`, `Story`, and `Notification`.

Key Relationships:

- `User` → `Post`: 1-to-many
- `Post` → `Comment`: 1-to-many
- `User` ↔ `User` (Follow): many-to-many
- `User` → `Story`: 1-to-many
- `User` → `Notification`: 1-to-many

Data Storage Strategy

A hybrid database model (SQL + NoSQL) is used:

- **SQL (Relational):** Best for structured data and interactions like `User`, `Post`, and `Comment` to maintain structure.
- **NoSQL:** Handles high-volume, changing data like `Stories` and `Recommendation Feeds`.

Consistency and Scaling

- **Strong Consistency:** Required for core user interactions (likes and comments) to ensure credibility and show feedback.
- **Eventual Consistency:** Tolerated for non-critical features like `feeds` and `recommendations`. This prioritizes availability for non-critical operations while maintaining reliability for core interactions.

Optimization & Reliability

- **Indexing:** Applied to frequently queried fields: ‘`user_id`‘, ‘`post_id`‘, and ‘`created_at`‘.
- **Partitioning/Sharding:** Data is distributed by `user_id` to balance load. Sharding by `ID ranges` for `posts` and `comments` balances storage.
- **Backup Strategy:** Includes regular snapshots, replication across data centers, and disaster recovery protocols to ensure reliability.

Technology & Communication Decisions

Restful HTTP APIs

- Primary communication method for core user interactions: login, posting, profile management, following/unfollowing, feed retrieval
- Stateless and scalable design works well behind load balancers
- Easy integration with mobile and web clients using standard HTTPS

Push Notifications

- Delivered using APNs (iOS) and FCM (Android)
- Triggered by backend events (e.g., new follower, like, comment) through Kafka consumers
- No persistent connections required, since messaging is out of scope

Data Format

- All API responses and requests use JSON
- Lightweight, human-readable, and broadly supported

Load Balancing

- The load balancer distributes traffic across multiple API gateway and app server instances
- Health checks ensure failed instances are automatically removed from rotation
- Supports horizontal scaling as traffic grows

Caching

- Redis caches frequently accessed feed data, engagement counters, and user sessions
- CDN caches images and videos globally to minimize latency and origin load
- Reduces pressure on SQL + NoSQL databases by serving repeat data quickly

Security

- All communication over HTTPS/TLS
- JWT tokens for stateless authentication with short expiration times
- Passwords stored with Argon2 or bcrypt hashing and secure salting
- IAM policies enforce least-privilege access to blob storage
- Rate limiting + behavior checks defend against bots and brute force attacks

Design Evaluation

Monolithic Architecture

- **Pros:** Straightforward deployment.
- **Cons:** Poor scalability and difficult to maintain at Instagram's global scale.
- **Decision:** Rejected due to lack of flexibility and inability to handle billions of requests efficiently.

Serverless-first Architecture

- **Pros:** Automatic scaling.
- **Cons:** Not suitable for long-running tasks such as media transcoding or feed generation; limited control over performance tuning.
- **Decision:** Rejected because critical workloads require persistent, high-performance services.

Microservices + Event-Driven Architecture

- **Pros:** Each service scales independently, fault isolation, clear separation of concerns, asynchronous event handling via Kafka queues.
- **Cons:** Higher operational complexity, requires strong observability and coordination.
- **Decision:** Selected because it provides the scalability, resilience, and modularity needed for Instagram's diverse features (posts, stories, notifications).

Key System Characteristics

- **Scalability:** Services scale horizontally, databases are sharded, and caching/CDN reduce load.
- **Reliability:** Failover mechanisms, replication, and queues ensure continuity during failures.
- **Security:** Authentication, rate limiting, and role-based access control protect user data.
- **Observability:** Metrics, logs, and alerts provide visibility into performance and reliability.

System Reliability, Security & Scalability

Reliability & Failover

- Load balancer distributes traffic across multiple service instances so failures don't interrupt users
- SQL + NoSQL data replicated across availability zones for continuous read/write access during node issues
- Queues prevent data loss by retrying feed/media jobs when services fail
- Cached feeds enable graceful degradation if backend latency increases

Backups & Disaster Recovery

- Daily backups and database snapshots allow restoration after failures
- Blob storage keeps media durable via automatic replication
- Periodic DR tests verify backup integrity and recovery speed

Scalability

- Stateless services scale horizontally by adding more app/API instances
- Sharding partitions user and feed data to reduce hot spots and support growth
- Asynchronous workers handle heavy writes (fan-out + processing) to keep APIs fast
- Redis + CDN reduce read load on databases and speed up global access

Security & Authentication

- All communication over HTTPS/TLS
- JWT tokens for secure, stateless session handling
- Passwords hashed with Argon2/bcrypt and protected with salting
- IAM roles and signed URLs ensure least-privilege access to stored media
- Rate-limiting and bot detection protect from abuse

Monitoring & Alerting

- Metrics tracked for latency, errors, cache hits, and queue depth
- Structured logging and distributed tracing help diagnose issues quickly
- Alerts trigger when performance or reliability targets (SLOs) begin to degrade