Team: Design Instagram

TEAM LEAD: SEHAR ALI

RESEARCH ANALYST: MAAKHAM JAVED

DOCUMENTATION LEAD: MUHAMMAD ALI

PRESENATION LEAD: TEHRIM IQBAL

Problem Statement

Overview:

- Instagram is a social media platform used by millions to share and consume visual content. It delivers fast and smooth visual content while supporting real-time engagement at a massive scale.

Problem:

- Instagram must be scalable, secure and highly responsive to protect user data, maintain privacy, and handle massive traffic. The system must also ensure efficient database design and high performance under heavy traffic.

Users

User Personas

- Creators- Post content to express themselves and build an audience.
- Casual Users- Browse content and interact with others.
- Viewers/ Consumers- Discover trending posts, reels and hashtags.
- Advertisers/Brands- Reach target audiences and measure engagement.
- Admins/Moderators- Manage platform activity, review reports and ensure safety.

Goals

Main User Goals

- Share- Upload photos, videos, stories and reels.
- Discover- Explore new content, trends, and recommend posts.
- Engage- Like, comment, follow, message, and interact with creators.
- Grow- Build followers, increase visibility, improve engagement.
- Monetize- Use ads, promotions, and creator tools to earn revenue.

Requirement Analysis- Functional

FUNCTIONAL DETAIL	DESCRIPTION	PRIORITY
USER REGISTRATION & AUTHENTICATION	Users sign up with a username, email, or phone and authenticate through secure password hashing, JWT sessions, and optional MFA.	MUST-HAVE
PROFILE MANAGEMENT	Profiles let users update their info, privacy settings, and account type, with optional analytics and contact links.	MUST-HAVE
PHOTO/VIDEO UPLOAD	The system stores common media formats, transcodes them into multiple versions, adds metadata, and supports efficient uploads via pre-signed URLs.	MUST-HAVE
FOLLOW/UNFOLLOW	Core features include follow/unfollow, likes, comments, messages, and notifications for important user activity.	MUST-HAVE
PERSONALIZED FEED	The system must provide an infinite, ranked feed that mixes followed content with recommendations and supports cursor-based pagination.	MUST-HAVE
LIKES, COMMENTS, & SHARES	The system must record engagement actions and forward them to analytics and ranking, allowing some counts (like likes) to update eventually under heavy load.	MUST-HAVE
STORIES (24-HOUR)	Users can post ephemeral Stories that expire in 24 hours, are edge-cached, and optionally saved as highlights.	SHOULD-HAVE
SEARCH & DISCOVERY	Search must cover users, hashtags, captions, and surface recommendations using indexing and ML signals.	SHOULD-HAVE
NOTIFICATIONS	The system must send timely push and in-app notifications for likes, comments, follows, and messages using APNs and FCM.	MUST-HAVE

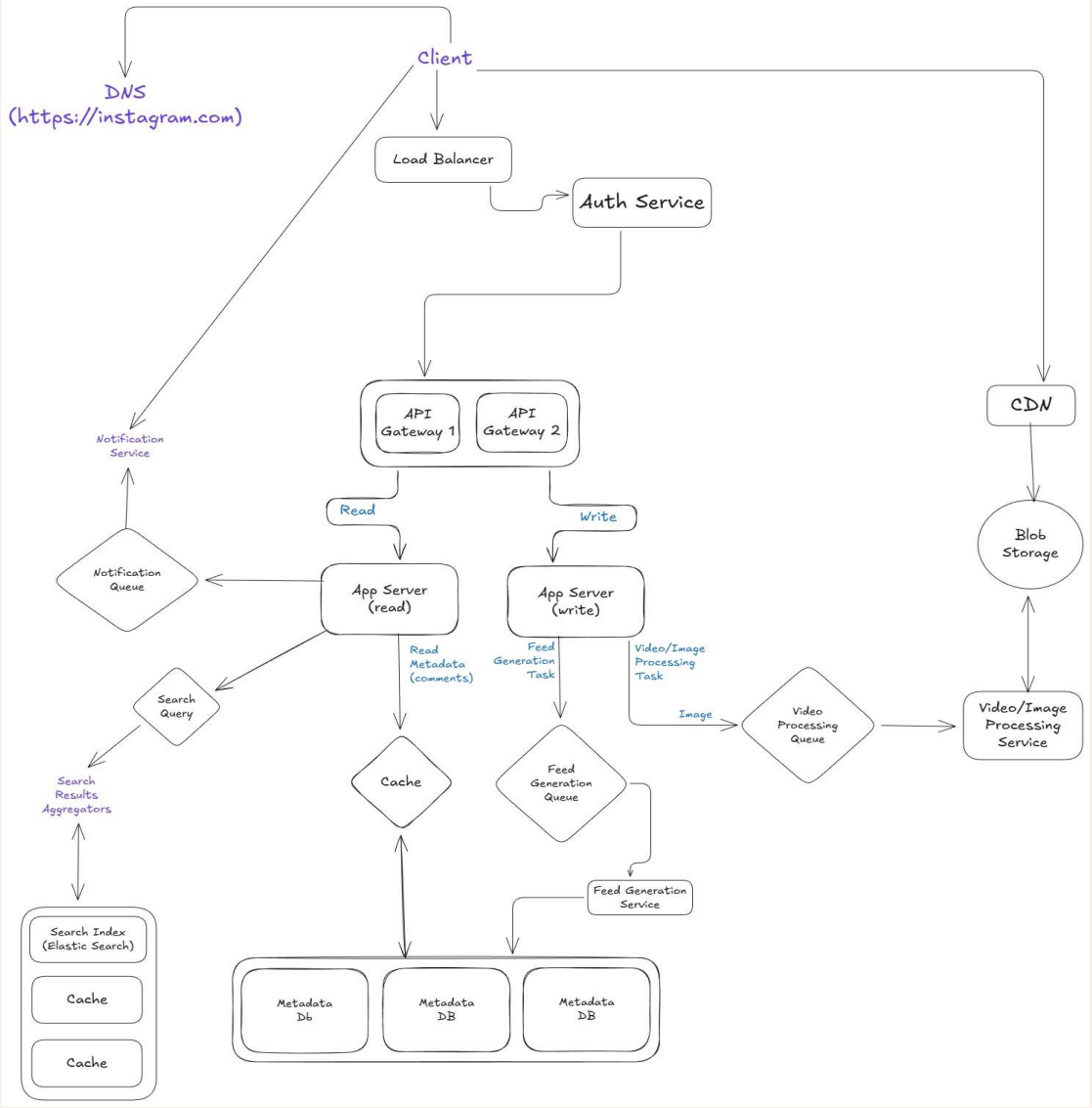
Requirement Analysis - Non- Functional

FUNCTIONAL DETAIL	DESCRIPTION	PRIORITY
SCALABILITY	The system must scale horizontally with sharding, autoscaling, and multi-region deployment to handle massive traffic and media storage.	MUST-HAVE
LATENCY	Feed and UI actions must stay under 200ms p95 latency, and video 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	Eventual consistency is acceptable for social features and feeds, but authentication and billing must remain strongly consistent.	SHOULD-HAVE
SECURITY & PRIVACY	Encrypt data in transit and at rest, use least-privilege access, rotate secrets, and support DSAR requests.	MUST-HAVE
OBSERVABILITY	Collect traces, metrics, and logs, with SLOs and alerting in place.	SHOULD-HAVE

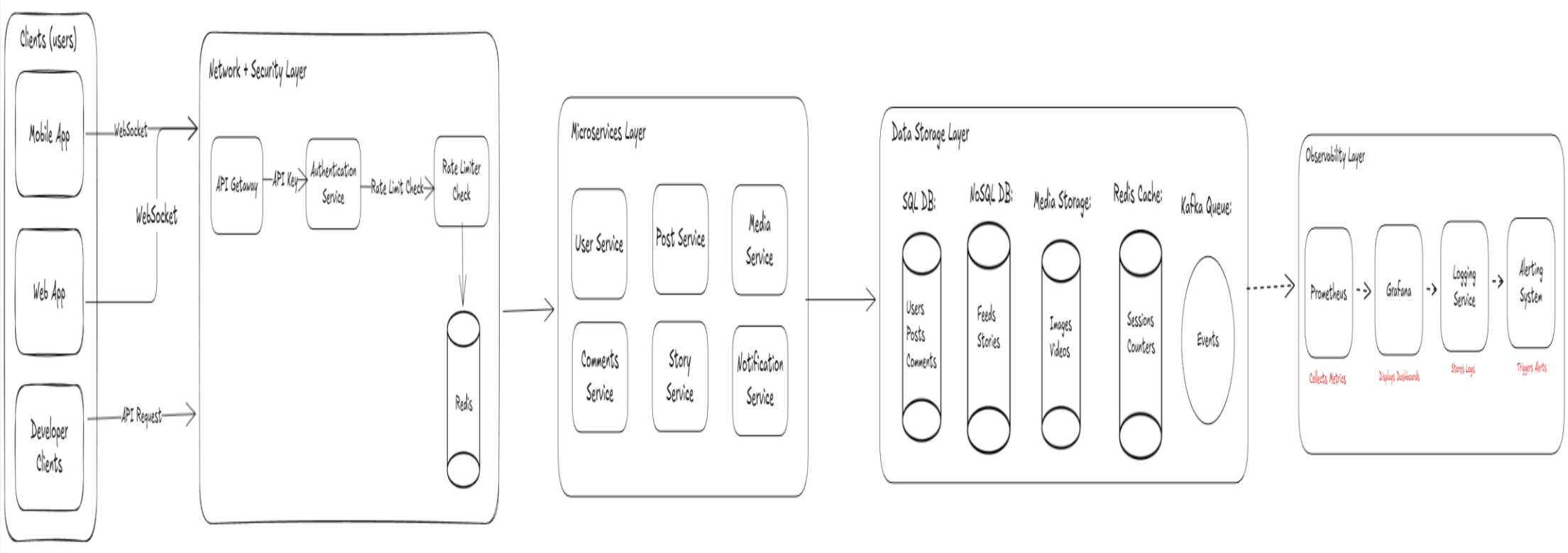
Component

COMPONENT	RESPONSIBILITY
CLIENTS	UI, render feed, upload media, open Web-Socket 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

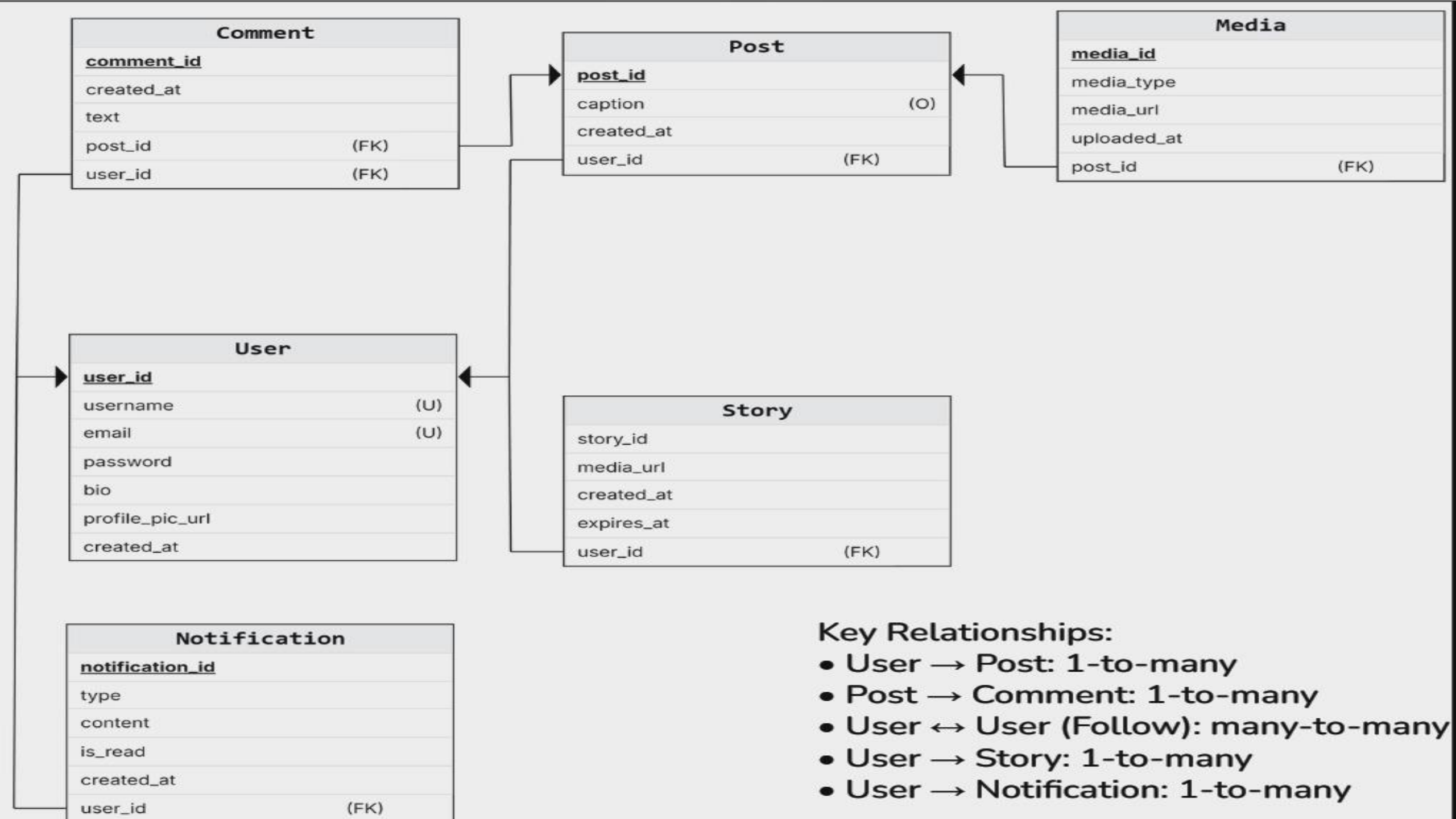
High-Level Function Diagram



System Architecture Design



Excalidraw Link: https://excalidraw.com/#json=CMxD54yPrurTcGPLyBan,hB44DyEU0CW_iKN4MorEWw


Comment
comment_id
created_at
text
post_id (FK)
user_id (FK)
Post
post_id
caption (O)
created_at
user_id (FK)
Media
media_id
media_type
media_url
uploaded_at
post_id (FK)
User
user_id
username (U)
email (U)
password
bio
profile_pic_url
created_at
Story
story_id
media_url
created_at
expires_at
user_id (FK)
Notification
notification_id
type
content
is_read
created_at
user_id (FK)
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

Technology and Communication Decisions

COMMUNICATION:

REST APIS: Main method for login, posting, feed retrieval.

Push Notifications: Sent to users when events happen (likes, comments, messages) via Kafka.

Data Format: JSON for all requests/responses.

PERFORMANCE & SCALING:

Load Balancing: Spreads traffic across gateways & servers, supports. horizontal scaling.

Caching: Redis for feeds/sessions, CDN for global image/video delivery.

SECURITY:

Transport: all traffic secured with HTTPS/TLS.

Authentication: JWT tokens for stateless sessions.

Credentials: passwords stored with Argon2 + salting.

Defense: rate limiting & behavior checks against bots/brute force.

DESIGN EVALUATION & SYSTEM RELAIABILITY

Design Evaluation

- Monolithic: Simple but not scalable; REJECTED
-
- Serverless: Auto-scaling but limited for heavy tasks, REJECTED
-
- Microservices: + Events: Modular, scalable, resilient, SELECTED

System Reliability

Scalability: Services scale independently, databases sharded, caching/CDN reduce load

Fault Tolerance: Queues, replication, failover keep system running during failures

Security: Authentication, rate limiting, role checks protect user data

Observability: Metrics, logs, alerts ensure performance and quick issue detection

Design Evaluation & System Reliability

DESIGN EVALUATION

- MONOLITHIC: Simple but not scalable; REJECTED
- SERVERLESS: Auto-scaling but limited for heavy tasks, REJECTED
- MICROSERVICES+ EVENTS: Modular, scalable, resilient, SELECTED

SYSTEM RELIABILITY

- SCALIBILITY: Services scale independently, databases sharded, caching/CDN reduce load
- FAULT TOLERANCE: Queues, replication, failover keep system running during failures
- SECURITY: Authentication, rate limiting, role checks protect user data
- OBSERVALITY: Metrics, logs, alerts ensure performance and quick issue detection