Mastering System Design

Design a Video Sharing Platform (aka YouTube)

What Are We Building?

- Design a system where users can:
  - Upload and watch videos
  - Like, comment, and share
  - Search and browse content
  - Subscribe to channels
  - View recommendations
- Key User Workflows
  - 📤 Upload a Video
    - Chunked upload → Processing → Storage → Playback
  - 🎬 Watch a Video
    - Stream from CDN based on network/bandwidth
  - 🔍 Search/Browse
    - Filter by tags, categories, popularity
  - ❤️ Engage
    - Likes, comments, shares, subscriptions

Functional Requirements

- User registration and authentication
- Upload video content
- Encode video into multiple resolutions
- Stream videos on demand (adaptive quality)
- Video metadata (title, description, tags)
- Likes, comments, views, and subscriptions
- Search by keyword, category, or tags
- Personalized home/feed

Non-Functional Requirements

- ⚡ Low latency video streaming
- 🧩 High availability of videos and metadata
- 🚀 Scalability to billions of videos
- 💾 Efficient storage and cost management
- 🌎 Global delivery using CDN
- 🔐 Security & abuse prevention

Assumptions and Constraints

- 🔧 Assumptions
  - Users will upload mostly short-form content (≤ 15 minutes).
  - Content is static (not live).
  - Multiple video qualities will be supported (240p - 4K).
  - CDN will be used for global content delivery.
  - Metadata is small and queryable (title, tags, timestamps).
- 🚧 Constraints & Challenges
  - High storage volume: TBs per day.
  - Processing pipeline needs to scale.
  - Playback must support adaptive bitrate streaming.
  - Abuse prevention: spam, copyright, explicit content.
  - Consistency of metadata vs video availability.
  - Cost management at scale (storage + CDN egress).

Scale Assumptions

- 👥 100M users
- 📹 10M videos uploaded per day
- 👀 500M daily video views
- 💬 100M comments, likes & shares per day
- ⏱️ Average video length: 10 mins
- 🧠 Video metadata: ~1KB; Engagement events: ~500B per action

Estimating Storage Needs

- Raw Video Storage
  - Avg upload: 10 min @ 5MB/min → 50MB per video
  - 10M uploads/day → 500TB/day
  - With 30-day retention (for processing): ~15PB/month
- Encoded Versions
  - Assume 4 variants: 240p, 480p, 720p, 1080p
  - Storage multiplies ~3x: 1.5PB/day → ~45PB/month

Estimating Bandwidth Needs

- Video Streaming Bandwidth
  - Avg user watches 3 videos/day → 300M hrs/day
  - Assume avg 1Mbps streaming = ~0.45GB/hr
  - Total egress/day = 135PB/day
  - CDN must handle millions of concurrent viewers
- Peak Load Estimate
  - 10M concurrent streams @ 1Mbps → ~10 Tbps egress

Metadata and Engagement Scale

- Video metadata: 10M new rows/day
- Likes/comments: 100M new events/day
- Search index updates in real time
- Hot content = read-heavy patterns

Implications of Scale Assumptions

- 📦 Storage
  - Requires multi-tiered storage (hot, warm, cold)
  - Frequent writes → distributed blob storage (e.g., S3, GCS)
  - Cold storage and deletion policies to control cost
- 🖥️ Processing
  - Encoding pipeline needs autoscaling & GPU support
  - Parallel processing jobs per resolution
- 🌍 Global Distribution
  - CDN integration is non-optional
  - Need region-aware content routing & geo-replication
- 📊 Engagement Data
  - Write-heavy → eventual consistency and event queues
  - Aggregation for views/likes should be async and sharded
- 🔍 Search & Discovery
  - Near real-time indexing at scale
  - Distributed search infrastructure (e.g., Elasticsearch, Meilisearch)

Core Components Overview

- **API Gateway**: Central entry point for clients; routes requests, enforces auth, and manages rate-limiting/logging
- **Upload & Ingestion**: Handles video file uploads, generates video IDs, stores temporarily, and triggers encoding jobs via a queue
- **Encoding & Processing**: Transcodes videos to multiple resolutions, generates thumbnails, prepares HLS/DASH formats, and stores in final storage
- **Video Storage & CDN**: Manages blob storage of video chunks and manifests; integrates with CDN for fast, geo-distributed playback
- **Metadata Service**: Stores video info (title, tags, uploader, etc.), enables search/filtering, and provides mapping between video ID and location
- **User Service**: Manages user accounts, auth (OAuth/JWT), channel subscriptions, and user preferences
- **Engagement Service**: Tracks views, likes, comments; supports async event logging and anti-abuse moderation logic
- **Search & Discovery**: Powers real-time search for videos/channels using tags, titles, and trends; supports indexing of new content
- **Recommendation Engine**: Personalizes video feed using behavioral data, embeddings, and collaborative filtering models

Communication Between Services and API Design

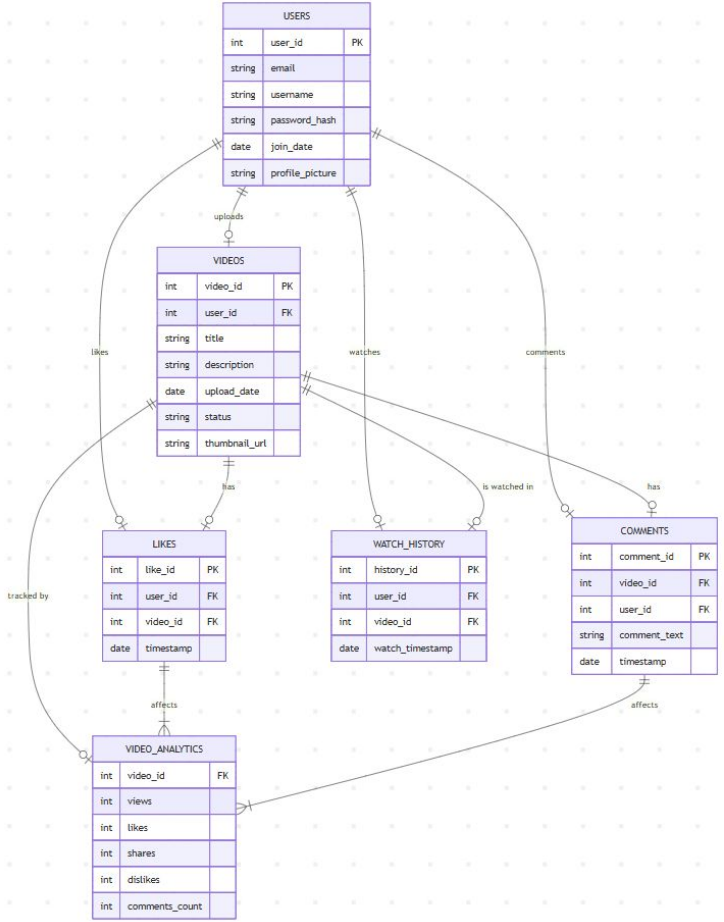
- 🔄 Types of Communication
  - Synchronous (HTTP/gRPC) for metadata fetch, user info, search
  - Asynchronous (Message Queue/Event Bus) for uploads, encoding, engagement events
- API Design
  - Client → API Gateway (REST)
    - Security: Authentication via OAuth2/JWT tokens
    - Endpoints: POST /upload, GET /videos/{id}, POST /like
  - API Gateway → Internal Services (REST)
    - Security: API Gateway handles authentication, rate-limiting, and routing
    - Services: Metadata, Encoding, Engagement, User
  - Video Upload → Encoding Service
    - Async Processing: Triggered via event bus (Kafka/SQS)
    - Security: Secure access to file storage via signed URLs

Storage and Caching Decisions

- Storage for Video Data
  - Video Files:
    - Object Storage (S3, GCS, Azure Blob) for scalable, durable storage.
    - Stores video chunks, manifest files, and thumbnails.
  - CDN:
    - Content Delivery Network (e.g., Cloudflare, Akamai) for fast, global video streaming.
    - Reduces latency by caching videos at edge locations.
- Storage for Metadata
  - Relational Database:
    - MySQL/PostgreSQL for structured metadata (video titles, tags, user data, etc.).
    - Provides fast querying and indexing for search.
  - NoSQL Database:
    - MongoDB for flexible data (e.g., user preferences, video recommendations).
- Caching & Performance
  - In-Memory Cache:
    - Redis/Memcached for frequently accessed data (video metadata, user sessions).
- Backup & Durability
  - Regular backups of databases and video files stored in geographically distributed regions.

Cursory DB Schema

- Users Table
  - user_id (PK), email, username, password_hash, join_date, profile_picture
- Videos Table
  - video_id (PK), user_id (FK), title, description, upload_date, status, thumbnail_url
- Likes Table
  - like_id (PK), user_id (FK), video_id (FK), timestamp
- Comments Table
  - comment_id (PK), video_id (FK), user_id (FK), comment_text, timestamp
- Watch History Table
  - history_id (PK), user_id (FK), video_id (FK), watch_timestamp
- Video Analytics Table
  - video_id (FK), views, likes, shares, dislikes, comments_count



Strategic Tech & Infra Decisions

- Frontend Framework
  - React.js / Vue.js for responsive, component-based UI.
- Backend Framework
  - Node.js with Express for scalable, asynchronous API handling.
- Database
  - PostgreSQL/MySQL for structured metadata storage.
  - MongoDB for flexible, scalable data (e.g., user preferences).
- Video Storage & CDN
  - AWS S3 / GCS for scalable, durable video storage.
  - Cloudflare / AWS CloudFront for low-latency video delivery.
- Authentication
  - OAuth2 / JWT for secure, token-based authentication.
- Event Processing
  - Kafka / SQS for asynchronous task processing (e.g., encoding, engagement).
- Infrastructure
  - AWS / GCP for scalable cloud infrastructure.
  - Kubernetes for efficient microservice orchestration.

The Final Design - Video Sharing Platform