Workshop 2

Juan Sebastian Colorado Caro, Walter Alejandro Suarez Fonseca
School of Engineering
Computer engineering
Universidad Distrital Francisco José de Caldas
Bogotá, Colombia
Email: jscoloradoc@udistrital.edu.co, wasuarezf@udistrital.edu.co
Click here to Git Repository

Abstract

The explosive growth of short-form video platforms like TikTok has highlighted the importance of robust, scalable, and efficient data architectures to support high-throughput video ingestion, real-time user interaction, and personalized content recommendation. This project proposes the database system design for a scalable short-form video platform inspired by TikTok, focusing on the integration of both relational and NoSQL technologies to balance transactional integrity with high-performance data retrieval. The system architecture includes a hybrid storage model, ETL pipelines for analytics, and query strategies for user engagement tracking and content recommendations. Initial queries demonstrate the system's capability to support user stories and business needs, such as retrieving user activity logs, trending content, and personalized video feeds.

Index Terms

data, analytics, databases, big data, SQL, NoSQL

I. Introduction

Inspired by TikTok's success, this project focuses on the design of a scalable data system architecture tailored for such a platform. The proposed system incorporates a hybrid data storage approach using both relational databases—for managing structured entities like users and video metadata—and NoSQL solutions—for handling unstructured or semi-structured data such as comments, likes, and video interaction metrics. The architecture supports real-time access, data-driven recommendations, and long-term analytics. This document outlines the architectural components, the type of information the system must handle, and key queries that illustrate the retrieval and processing of essential data to meet both business requirements and user expectations. Additionally, improvements from previous workshop iterations are integrated to refine the data model and system capabilities.

II. Data System Architecture

A. High-level architecture diagram

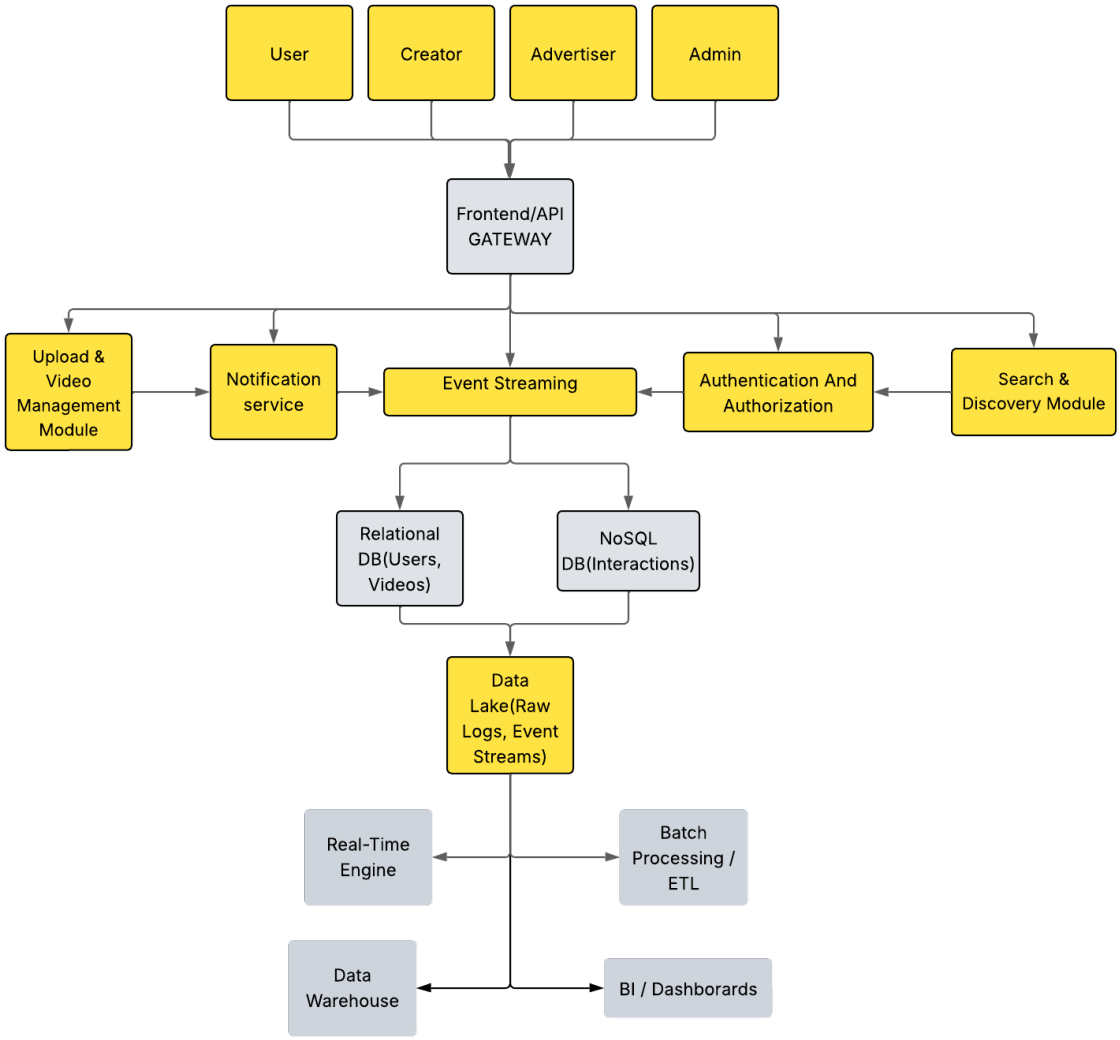

Fig. 1. High-level architecture diagram

III. Technology Stack

This section summarizes the core technologies used in the platform architecture, focusing on the role of each component, the selected technology, and the rationale behind the choice.

Technologies and Rationale

Frontend

Role: User interface for viewers, creators, advertisers, and admins.
Technology: React (Web), Flutter (Mobile)
Why: Easy to learn, widely used, supports responsive design, and has strong community support.

API Backend
Role: Handles business logic and communication between frontend and databases.
Technology: Node.js with Express
Why: JavaScript-based, lightweight, and easy to integrate with both SQL and NoSQL.

Relational Database
Role: Stores structured data like users, videos, and transactions.
Technology: PostgreSQL
Why: Open-source, reliable, supports complex queries, and has good documentation.

NoSQL Database
Role: Stores high-volume interactions like likes, comments, and views.
Technology: Firebase Realtime Database
Why: Simple setup, real-time sync, and low learning curve for handling unstructured data.

Object Storage
Role: Stores video files and media assets.
Technology: Firebase Storage or Amazon S3
Why: Easy to integrate with backend and mobile apps, scalable and affordable.

Event Streaming
Role: Streams real-time events (likes, views, uploads) to storage and processing modules.
Technology: Redis Streams
Why: Lightweight alternative to Kafka, easier to configure and deploy for small to medium systems.

Data Lake
Role: Stores raw logs and event data for future analytics.
Technology: Cloud Storage Buckets (e.g., Google Cloud Storage)
Why: Simple object storage with high capacity and minimal configuration.

Data Warehouse
Role: Stores processed data for analytics and dashboards.
Technology: Google BigQuery
Why: Easy to use, integrates with cloud storage, and requires minimal setup for analytical queries.

Dashboards & BI
Role: Displays metrics for admins, creators, and advertisers.
Technology: Google Data Studio or Metabase
Why: No-code/low-code, user-friendly interfaces, and quick integration with SQL/BigQuery.

Component Roles and Data Flow

- **Users (User, Creator, Advertiser, Admin)**: Interact with the platform through the frontend to upload videos, search content, receive notifications, and perform various actions.
- **Frontend/API Gateway**: Manages all incoming user requests. It routes them to the appropriate internal modules and services securely and efficiently.

- **Upload & Video Management Module**: Handles video file uploads, metadata entry, and draft saving. Sends video metadata to the relational database and files to object storage.
- **Notification Service**: Sends real-time notifications to users (e.g., new followers, comments) by consuming event data from the streaming module.
- **Authentication and Authorization**: Verifies user identity and applies access control policies based on user roles (e.g., admin, creator).
- **Search & Discovery Module**: Enables users to search and discover videos using filters like categories, hashtags, and duration. Retrieves data from relational and NoSQL databases.
- **Event Streaming**: Captures real-time events (likes, views, uploads) and distributes them to storage and processing components.
- **Relational Database (Users, Videos)**: Stores structured data such as user accounts, video metadata, and transactions.
- **NoSQL Database (Interactions)**: Stores high-volume, semi-structured data such as likes, views, and comments. Optimized for fast writes and reads.
- **Data Lake (Raw Logs, Event Streams)**: Stores raw, unprocessed data and event logs for further transformation and analysis.
- **Real-Time Engine**: Processes incoming event streams to compute dynamic metrics such as trending videos or live user engagement.
- **Batch Processing / ETL**: Periodically processes and transforms stored data from the data lake and loads it into the data warehouse.
- **Data Warehouse**: Stores structured and historical data optimized for analytics and reporting.
- **BI / Dashboards**: Provides visual reports and business insights for administrators, creators, and advertisers based on data warehouse content.

IV. Information Requirements

The system must retrieve and manage various types of information to support the functionalities outlined in the business model and user stories. Below is a list of the main types of information, their descriptions, and how they support the platform's logic.

1. User Profiles and Authentication Data

Description: Information about users, including credentials (for authentication), profiles, roles (viewer, creator, advertiser, admin), and preferences.
Link to Business Model: Essential for all customer segments (users, creators, advertisers, admins) to interact with the platform.
Link to User Stories:

- User registration and authentication (Functional Requirement #1)
- Follow creators and receive notifications (User Story: *Follow a Creator*)

2. Video Metadata

Description: Details about uploaded videos, such as title, description, tags, duration, uploader ID, geolocation, and engagement metrics (likes, comments, shares).

Link to Business Model: Core to the value proposition for creators and users, enabling content discovery and monetization.
Link to User Stories:

- Video upload and management (Functional Requirement #2)
- Search for videos with filters (User Story: *Search for Videos*)
- Discover trending videos (User Story: *Discover Trending Videos*)

3. User Interactions

Description: Data on user interactions with videos, including likes, comments, shares, watch time, and follow actions.
Link to Business Model: Drives engagement metrics for advertisers and creators, supporting monetization.
Link to User Stories:

- Like a video (User Story: *Like a Video*)
- Comment on a video (User Story: *Comment on a Video*)
- Real-time analytics for advertisers (User Story: *View Real-Time Analytics*)

4. Advertisements and Campaigns

Description: Information about ad campaigns, including targeting parameters (demographics, interests), budgets, impressions, clicks, and conversions.
Link to Business Model: Key revenue stream from advertisers; supports targeted ad placements.
Link to User Stories:

- Create ad campaigns (User Story: *Create Ad Campaign*)
- Edit campaign targeting (User Story: *Edit Campaign Targeting*)
- Export campaign reports (User Story: *Export Campaign Report*)

5. Monetization and Transactions

Description: Records of financial transactions, such as virtual gifts, ad revenue payouts, and sponsored content payments.
Link to Business Model: Critical for creator monetization and platform revenue streams.
Link to User Stories:

- Process payments for creators (Functional Requirement #6)
- Pause/resume ad campaigns (User Story: *Pause/Resume Campaign*)

6. Content Moderation Reports

Description: Data on flagged videos, including report reasons, severity levels, and moderation status (approved/rejected).
Link to Business Model: Ensures platform health and compliance, supporting community safety.
Link to User Stories:

- Report inappropriate content (Functional Requirement #5)
- Review flagged videos (User Story: *Review Content Reports*)

7. Analytics and Business Intelligence (BI)

Description: Aggregated metrics for dashboards, including user retention, engagement trends, revenue reports, and content performance.
Link to Business Model: Provides actionable insights for admins, creators, and advertisers.
Link to User Stories:

- Access BI dashboards (Functional Requirement #7)
- Monitor system health (User Story: *Monitor System Health*)

V. Query Proposals

This section presents a set of representative queries designed to demonstrate how the proposed data system retrieves key information to support business needs and user interactions. In accordance with the hybrid architecture of the platform, the queries are divided between relational (SQL) and non-relational (NoSQL) approaches, each serving different access patterns and performance requirements.

For each of the main information requirements identified in the system—ranging from user profiles to business intelligence metrics—this section provides at least one example query, along with a brief explanation of its purpose and the insight it delivers. When applicable, the queries demonstrate how SQL and NoSQL complement each other by handling different aspects of the same functional domain (e.g., video metadata vs. engagement analytics).

A. Sample Queries

1) User Profiles and Auth Data:

- SQL Query

```
SELECT u.user_id, u.username, u.email, u.role,
COUNT(f.follower_id) AS follower_count
FROM User u
LEFT JOIN Follow f ON u.user_id = f.followed_id
WHERE u.role = 'CREATOR'
GROUP BY u.user_id, u.username, u.email, u.role
ORDER BY follower_count DESC
LIMIT 10;
```

Purpose: Retrieve the top 10 creators with the most followers. This helps identify high-impact users for promotional or monetization purposes.

2) Video Metadata:

- SQL Query

```
SELECT video_id, title, description, upload_datetime
FROM Video
WHERE visibility = 'PUBLIC'
ORDER BY upload_datetime DESC
LIMIT 10;
```

- noSQL

```
db.Reactions.find({ video_id: 123 }, { likes_count: 1 });
```

Purpose: The SQL query retrieves recently uploaded public videos. The NoSQL query gets real-time engagement data (like counts).

3) Video Metadata:

- NoSQL Query

```
db.Views.aggregate([
  { $match: { video_id: 123 } },
  { $unwind: "$view_events" },
  { $group: { _id: null, avg_duration: { $avg: "$view_events.duration" }
]);
```

Purpose: Calculate average watch time of a specific video to measure engagement.

Insight: This data is essential for performance scoring, recommendation engines, and creator insights. NoSQL enables high-throughput ingestion and aggregation of watch data.

4) Advertisements and Campaigns:

- SQL Query

```
SELECT campaign_id, name, budget, start_date, end_date
FROM Campaign
WHERE advertiser_id = 21 AND end_date >= CURRENT_DATE;
```

- NoSQL Query

```
db.AdImpressions.countDocuments({ campaign_id: 21, type: "click" });
```

Purpose: SQL fetches active campaign metadata. NoSQL counts user clicks for real-time campaign performance.

Insight: SQL supports campaign management, while NoSQL enables rapid, scalable analytics of ad engagement.

5) Monetization and Transactions:

- SQL Query

```
SELECT u.username, t.amount, t.type, t.transaction_datetime
FROM Transaction t
JOIN User u ON t.user_id = u.user_id
WHERE t.type = 'SUBSCRIPTION'
ORDER BY t.transaction_datetime DESC
LIMIT 10;
```

Purpose: Display recent subscription transactions, including user info, for administrative and accounting purposes.

Insight: Critical for verifying revenue flows and creator payments.

6) Content Moderation Reports:

- SQL Query

```
SELECT r.video_id, r.reason, r.status, u.username AS reported_by
FROM ContentReport r
JOIN User u ON r.reporter_id = u.user_id
WHERE r.status = 'PENDING';
```

Purpose: List all pending reports with context for moderators to review inappropriate content.

Insight: Enables content moderation workflow, ensuring safety and compliance.

7) Analytics and Business Intelligence (BI):

- SQL Query

```
SELECT DATE(transaction_datetime) AS day,
       SUM(CASE WHEN type = 'SUBSCRIPTION' THEN amount ELSE 0 END)
       AS subs_revenue,
       SUM(CASE WHEN type = 'AD_PAYMENT' THEN amount ELSE 0 END)
       AS ads_revenue
FROM Transaction
GROUP BY day
ORDER BY day DESC
LIMIT 7;
```

Purpose: Compare platform revenue by source over the past week.

- NoSQL Query

```
db.Views.aggregate([
  { $unwind: "$view_events" },
  { $group: { _id: "$video_id", views: { $sum: 1 } } },
  { $sort: { views: -1 } },
  { $limit: 5 }
]);
```

Purpose: Find top-performing videos by views. NoSQL computes it from raw event data.

8) System Performance Metrics:

- NoSQL Query

```
db.SystemMetrics.aggregate([
  { $match: { metric: "latency",
    timestamp: { $gte: ISODate("2025-05-28T00:00:00Z") } } },
  { $group: { _id: "$endpoint", avg_latency: { $avg: "$value" } } }
]);
```

Purpose: Compute average latency per API endpoint for a given day.

Insight: Critical for monitoring and scaling backend services; identifies performance bottlenecks in the platform.

B. Needs Leverage

The hybrid architecture of the platform is intentionally designed to take advantage of the distinct strengths of relational and non-relational data systems. Relational (SQL) databases are used for structured, transactional data that requires consistency, referential integrity, and auditability—such as user accounts, subscriptions, payments, and moderation workflows. These queries support administrative tasks, financial tracking, and long-term analytics.

In contrast, NoSQL (document-based) storage is optimized for high-velocity, schema-flexible, and interaction-heavy data, such as video views, user reactions, comments, session logs, and ad impressions. These queries support real-time features like video feed generation, engagement metrics, live analytics dashboards, and monitoring.

In practice, both technologies work in tandem: SQL provides a reliable backbone for core business operations, while NoSQL enables scalable, low-latency access to dynamic content and behavior-driven insights. For example, while SQL might compute total subscription revenue per day for business intelligence dashboards, NoSQL allows instant retrieval of a user's watch history or a video's live engagement statistics.

VI. Improvements to Workshop 1

A. Business Model Scope Adjustment: Before vs. After

After receiving feedback, the initial version of the Business Model Canvas was refined to better reflect a realistic scope. The original model aimed to replicate features and operations of large-scale platforms such as TikTok. The revised version focuses on what is achievable in an early-stage or academic project context. Below is a direct comparison:

Business Model Component	Before (Overambitious)	After (Realistic)
Key Activities	- Implementing and optimizing targeted advertising strategies - Conducting market research and revenue optimization - Managing brand sponsorship campaigns	- Managing the user and creator community - Moderating user-generated content - Supporting basic creator promotions and feedback channels
Key Resources	- Big Data and real-time analytics infrastructure - AI-based recommendation engines	- MVP-level platform with essential video-sharing features - Early user base and basic engagement metrics
Value Proposition	- Advanced monetization options for creators - Real-time ad analytics for brands - Personalized recommendations via machine learning	- Simple content creation and sharing tools - In-app tipping or support for creators - Organic discovery of trending content
Revenue Streams	- Targeted advertising and branded content - Premium analytics features - Sale of anonymized user data	- In-app tipping system - Freemium model (e.g., ad removal) - Simple ad placements for local brands
Customer Segments	- Global advertisers and data analysts - Established content creators with commercial goals	- Young users seeking entertainment - Aspiring creators wanting exposure - Small businesses looking to promote their content

TABLE I
BUSINESS MODEL CANVAS: BEFORE VS. AFTER SCOPE REFINEMENT

These revisions help maintain a feasible business model, aligned with the resources, scale, and goals of the project. More advanced features, such as real-time analytics or advanced monetization, can be explored in later development stages.

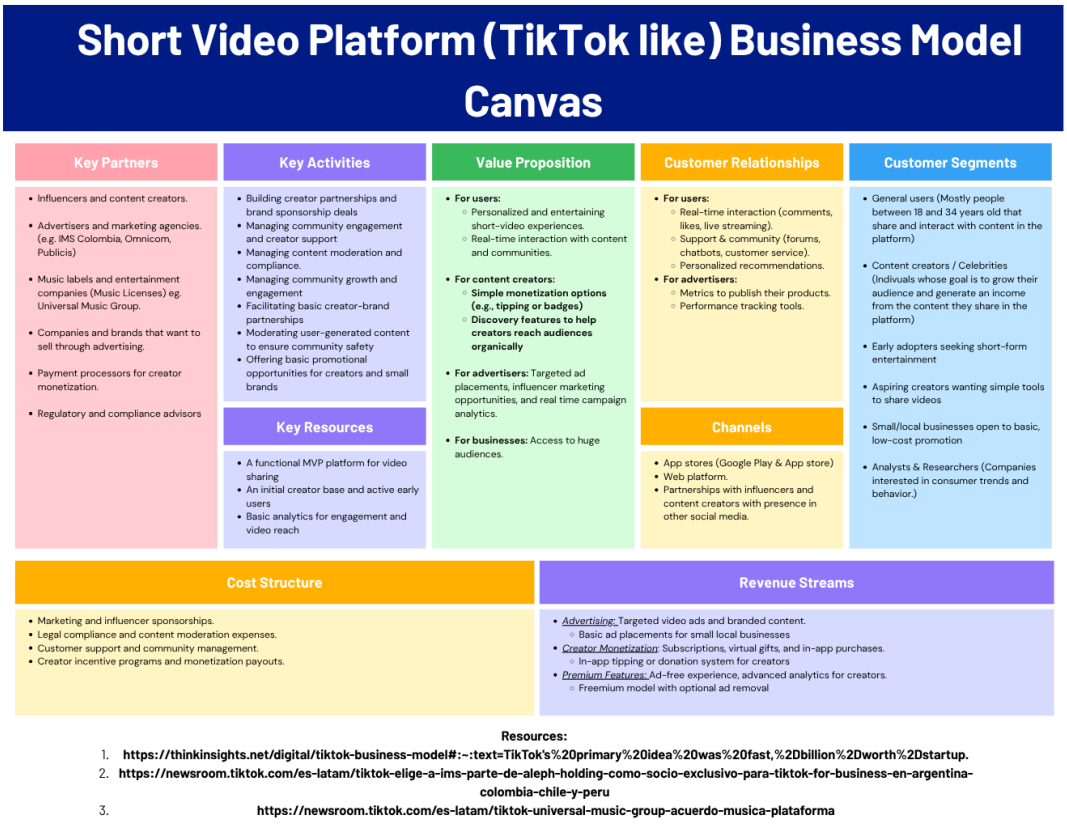

Fig. 2. Business Model Canvas Refined

B. Functional Requirements Mapping

Functional Requirements

1) **Functional requirement #1: User Registration and Authentication**
   - The user must be able to register with a username, email, and password.
   - The user must be able to log in securely using authentication (e.g., OAuth2).

2) **Functional requirement #2: Video Upload and Management**
   - The creator must be able to upload videos with a title, description, tags, and location.
   - The creator must be able to save video drafts and publish them later.

3) **Functional requirement #3: Content Interaction**
   - The user must be able to like, comment on, and share videos.
   - The user must be able to follow creators and receive notifications for new uploads.

4) **Functional requirement #4: Search and Discovery**
   - The user must be able to search for videos using filters like categories, hashtags, and duration.
   - The user must be able to view a "Trending" feed updated based on recent engagement.

5) **Functional requirement #5: Reporting and Moderation**
   - The user must be able to report inappropriate content.
   - The admin must be able to review and manage content reports.
6) **Functional requirement #6: Monetization and Payments**
   - The advertiser must be able to create ad campaigns with targeting options.
   - The advertiser must be able to view real-time analytics for campaigns.
   - The creator must be able to receive payments for ad revenue or virtual gifts.
7) **Functional requirement #7: Analytics and BI**
   - The creator must be able to access statistics about their videos (views, retention, engagement).
   - The admin must be able to generate periodic reports on user growth and content trends.

C. User Stories with Priority, Effort Estimate, and Acceptance Criteria

This section presents the refined user stories for the main user roles in the platform: User, Content Creator, Advertiser, and Administrator. Each story now includes a defined priority, estimated effort, and expanded acceptance criteria.

1) User Stories – General Users:

User Story 1 – Register and Authenticate
As a user
I want to register and log in to the platform
So that I can access its features securely
Priority: High
Estimated Effort: 1.5 days
Acceptance Criteria:
- Registration form includes username, email, password.
- User receives confirmation upon successful registration.
- Login system validates credentials securely.
- Users with incorrect credentials see appropriate error messages.

User Story 2 – Discover Trending Videos
As a user
I want to discover trending videos quickly
So that I can stay updated and engage with popular content
Priority: High
Estimated Effort: 1.5 days
Acceptance Criteria:
- Given I am logged in, when I open the Trending tab, then I see the top 20 videos sorted by engagement in the past 24 hours.
- Each video shows title, thumbnail, and like count.
- Feed updates at least once per hour.

User Story 3 – Like a Video
As a user
I want to like videos
So that I can interact and support content I enjoy
Priority: High
Estimated Effort: 1 day
Acceptance Criteria:

- When I click the Like button, the like is saved and acknowledged instantly.
- The button changes color or state to reflect the like.
- The total like count updates in real time.

User Story 4 – Follow a Creator
As a user
I want to follow creators
So that I can see more of their content in my feed
Priority: Medium
Estimated Effort: 1 day
Acceptance Criteria:

- When I click Follow on a creator's profile, I start following them.
- The feed is updated with content from followed creators.
- I receive notifications when followed creators post new videos.

User Story 5 – Report Inappropriate Content
As a user
I want to report inappropriate videos
So that the platform can take action to maintain safety
Priority: High
Estimated Effort: 1 day
Acceptance Criteria:

- I can click a report button on any video.
- A modal lets me choose a reason for reporting.
- A success message confirms that the report was submitted.

2) User Stories – Content Creators:

User Story 6 – Upload a Video
As a content creator
I want to upload videos to the platform
So that I can share my content with followers
Priority: High
Estimated Effort: 1.5 days
Acceptance Criteria:

- Upload form allows title, description, tags, and video file.
- Video is available on the creator's profile and in feeds.
- Upload must succeed with confirmation and validation.

User Story 7 – View Performance Analytics
As a content creator
I want to see analytics of my videos
So that I can understand my audience and improve content
Priority: Medium
Estimated Effort: 1 day
Acceptance Criteria:

- Dashboard shows views, likes, average watch time.
- Data can be filtered by date range.
- Analytics update at least once daily.

User Story 8 – Receive Virtual Gifts
As a content creator
I want to receive virtual gifts from viewers
So that I can earn revenue for my content
Priority: Medium
Estimated Effort: 1 day
Acceptance Criteria:

- Viewers can send gifts while watching content.
- I receive notifications for received gifts.
- Gift history is viewable in the analytics dashboard.

User Story 9 – Promote My Profile
As a content creator
I want to promote my profile via the platform
So that I can reach new audiences and grow followers
Priority: Low
Estimated Effort: 1.5 days
Acceptance Criteria:

- A form allows me to set budget, audience, and duration for profile promotion.
- My promoted profile appears in recommended sections.
- I can see basic metrics of campaign performance.

D. User Stories – Advertisers

User Story 10 – Create Ad Campaign
As an advertiser
I want to create ad campaigns with specific targeting
So that I can reach my desired audience effectively
Priority: High
Estimated Effort: 2 days
Acceptance Criteria:

- Campaign form includes budget, targeting (age, location, interests), start and end dates.

- Ads start serving after approval.
- Campaign status and performance are visible to the advertiser.

User Story 11 – Edit Campaign Targeting
As an advertiser
I want to edit the targeting parameters of a campaign
So that I can optimize ad performance
Priority: Medium
Estimated Effort: 1 day
Acceptance Criteria:

- I can update location, age range, and interest parameters.
- The changes take effect within one hour.
- Editing is only allowed on active campaigns.

User Story 12 – Pause or Resume Campaign
As an advertiser
I want to pause or resume a campaign
So that I have control over my ad spend
Priority: Medium
Estimated Effort: 0.5 days
Acceptance Criteria:

- Campaign can be paused/resumed from dashboard.
- System logs pause/resume time.
- Ads stop serving while paused.

User Story 13 – Export Campaign Report
As an advertiser
I want to export reports on my campaign
So that I can analyze performance offline
Priority: Medium
Estimated Effort: 1 day
Acceptance Criteria:

- Report includes impressions, clicks, engagement metrics.
- Exportable as PDF or CSV.
- Accessible from the advertiser dashboard.

E. User Stories – Admin

User Story 14 – Review Reported Content
As an admin
I want to view and manage content reports
So that I can ensure platform safety
Priority: High

Estimated Effort: 1.5 days
Acceptance Criteria:

- Admin dashboard lists all pending content reports.
- Admin can mark content as reviewed, rejected, or removed.
- Actions are logged with timestamp and admin ID.

User Story 15 – Monitor System Health
As an admin
I want to monitor system performance
So that I can detect and address issues proactively
Priority: Medium
Estimated Effort: 1 day
Acceptance Criteria:

- Dashboard shows server uptime, error rates, latency by endpoint.
- System alerts are triggered when thresholds are exceeded.
- Logs are accessible for inspection and diagnosis.

User Story 16 – Manage Users
As an admin
I want to manage user accounts
So that I can enforce policies and resolve abuse cases
Priority: Medium
Estimated Effort: 1 day
Acceptance Criteria:

- Admin can view, suspend, or delete user accounts.
- Actions require confirmation and are logged.
- Suspended users are blocked from login and content creation.

F. ER Design Methodology (10-Step Approach)

During Workshop 1, a preliminary ER diagram was created to represent the data model of the short-form video platform. However, this initial version was highly simplified and lacked many of the core entities and relationships required to support the business model, user interactions, and monetization features described in the project.

The Entity-Relationship (ER) diagram for this platform was developed using a 10-step design methodology. This structured approach ensures the database model accurately reflects the business logic, user stories, and system requirements, supporting both transactional integrity and future scalability.

1) **Requirements Collection and Analysis:** The platform's functional and non-functional requirements were derived from user stories, the business model canvas, and domain-specific research. Key areas included user management, video content, monetization (via

subscriptions and gifts), advertising campaigns, and content moderation.

2) **Identify Entity Types:** Entities were derived from real-world concepts central to the platform's functionality: *User*, *Video*, *Transaction*, *Subscription*, *SubscriptionPlan*, *GiftTransaction*, *VirtualGift*, *Advertiser*, *Campaign*, *ContentReport*, and *Follow*.

3) **Identify Relationship Types:** - Users upload videos: *User (1) — (N) Video*
   - Users subscribe to creators: *User (N) — (N) Subscription*
   - Users follow each other: self-relationship *User (N) — (N) Follow*
   - Users send virtual gifts to other users: *GiftTransaction (N) — (1) Transaction*
   - Advertisers manage campaigns: *Advertiser (1) — (N) Campaign*
   - Campaigns are financed through transactions: *Transaction (N) — (1) Advertiser*
   - Videos receive content reports: *Video (1) — (N) ContentReport*
   - Users review content reports: *User (1) — (N) ContentReport*
   - Subscriptions reference plans: *Subscription (N) — (1) SubscriptionPlan*

4) **Identify Attributes and Primary Keys:** Each entity was defined with meaningful attributes and keys. For example:
   - *User*: `user_id`, `email`, `role`, `username`
   - *Video*: `video_id`, `title`, `upload_datetime`, `visibility`
   - *Transaction*: `transaction_id`, `user_id`, `amount`, `type`, `advertiser_id`

5) **Map Attributes to Entities and Relationships:** - All relevant foreign keys were explicitly included (e.g., `plan_id`, `advertiser_id`, `reviewed_by`).
   - Weak entities such as *GiftTransaction* rely on *Transaction* for referential integrity.
   - Complex interactions are clearly modeled with multiple foreign keys.

6) **Draw the Initial ER Diagram:** An initial diagram was created (see section 4.2 of Workshop 1), which was later refined. The early version lacked key entities and relationships.

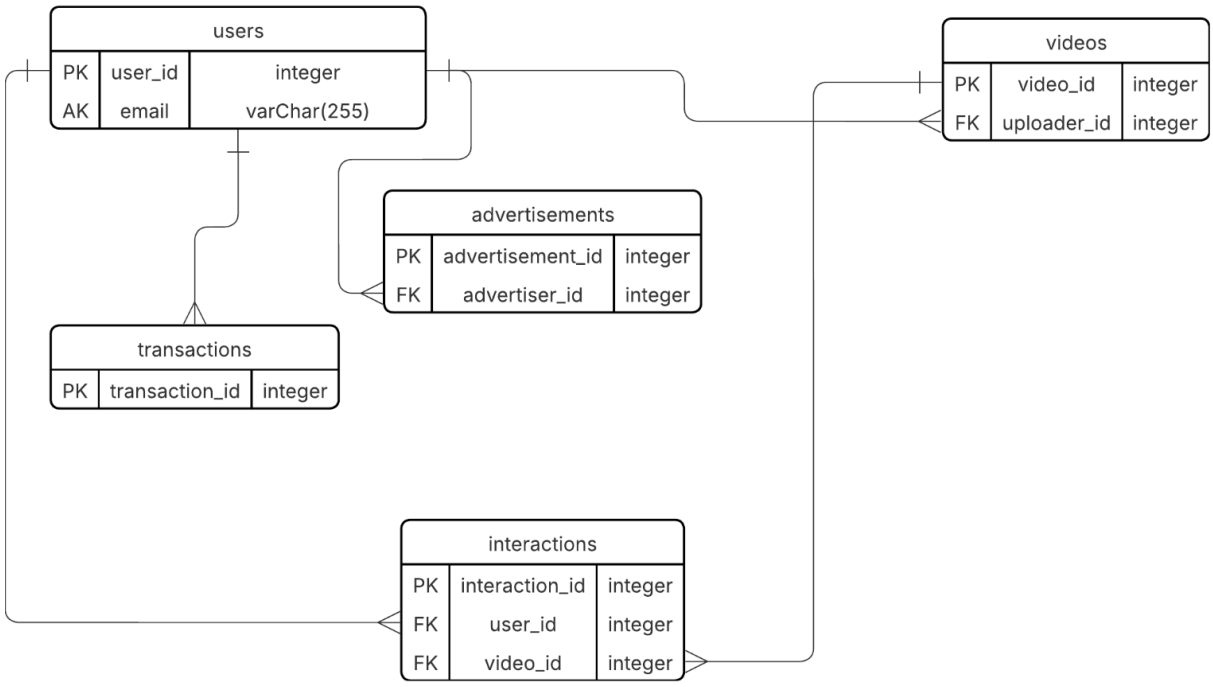

Fig. 3. Initial ER Diagram

7) **Review and Refine ER Diagram:** Based on deeper analysis and feedback, the ER model was extended to include:
   - Relationships between *Transaction* and *Advertiser* for ad payments.
   - Linking *GiftTransaction* to both sender and receiver users.
   - Content moderation with detailed mapping of reporter, reviewer, and affected video.
   - Association between *Subscription* and *SubscriptionPlan* to handle premium features.

8) **Map ER Diagram to Relational Schema:** Each entity and relationship was mapped to a relational schema. Many-to-many relations use junction tables (e.g., *Follow*, *Subscription*).

9) **Normalize to Eliminate Redundancy:** The schema was normalized to at least Third Normal Form (3NF). All non-key attributes depend on full primary keys; transitive dependencies were eliminated.

10) **Validate Against Use Cases:** The final ER model supports all major use cases:
   - Registration, following, video uploading, and real-time engagement.
   - Subscriptions and virtual gift transactions with proper billing records.
   - Full advertiser flow: campaign creation, payments, and system integration.
   - Comprehensive content reporting and moderation by administrators.

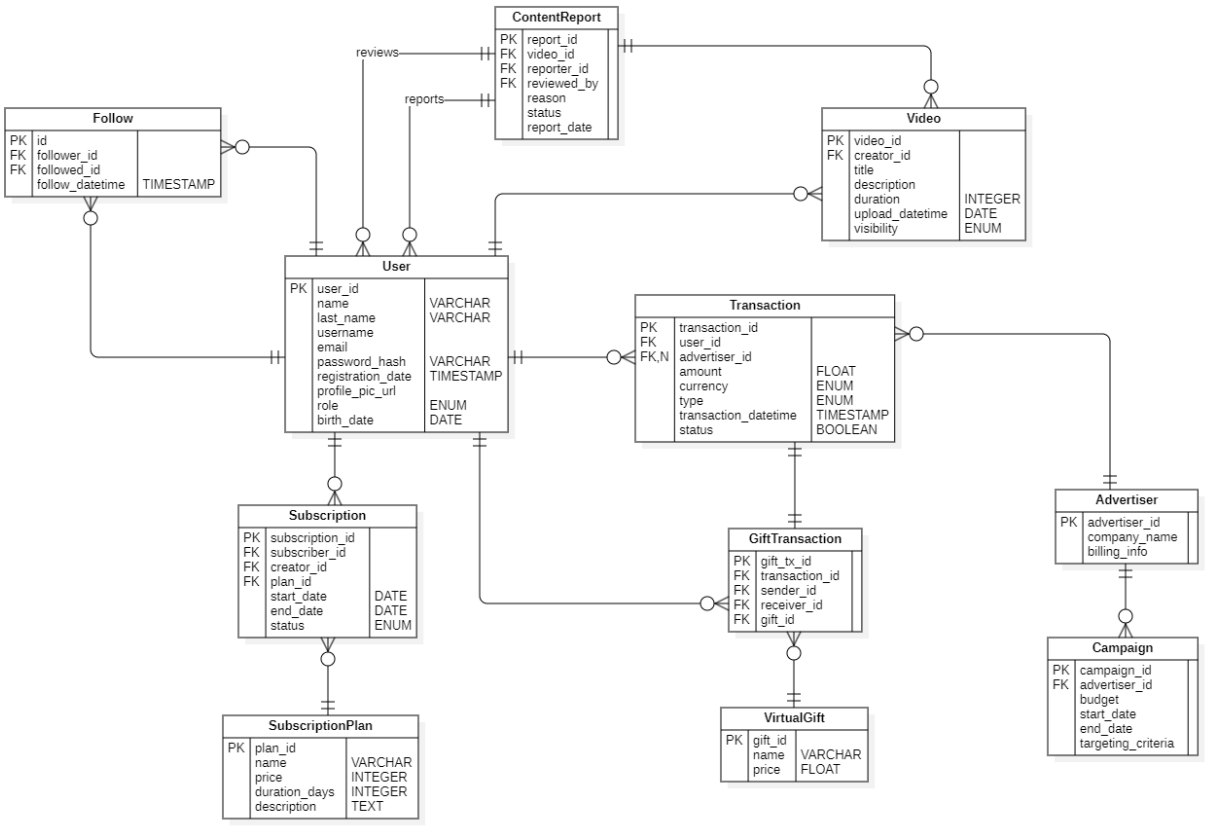

Fig. 4. ER Diagram Improved

1) Key improvements made:

- **Expanded Entity Set:** The new ER diagram introduces all essential entities aligned with the business logic, including Subscription, SubscriptionPlan, GiftTransaction, VirtualGift, Campaign, ContentReport, Follow, and a refined Transaction entity.

- **Role Management:** Instead of separating user types into different tables, the revised design uses a unified User table with a role attribute (GENERAL, CREATOR, ADMIN, etc.), simplifying access control and relationships.

- **Monetization Support:** The model now fully supports creator monetization through subscriptions and virtual gifts, along with advertiser billing via linked transactions.

- **Moderation Logic:** The new ContentReport table allows users to report inappropriate content, with the ability for administrators to review and act upon it.

- **Normalized Relationships:** Many-to-many relationships such as user follows and video interactions are handled using separate association tables like Follow, improving scalability and clarity.

- **Consistency with NoSQL Design:** The improved ER diagram complements the NoSQL

structure by maintaining structured, transactional data in SQL while allowing unstructured, high-volume interactions to be handled separately.

G. NoSQL Structure designed

During workshop 1 no NoSQL database structure was proposed, this is the initial structure that meets the criteria of our project, and complements the data in the SQL database.

Collection	**Purpose**	**Key Fields**	**Indexes**
Comments	Store all comments grouped by video	{ _id, video_id, comments:[{comment_id, user_id, text, timestamp}] }	video_id, comments.user_id
Reactions	Aggregate like/reaction counts (with optional history)	{ _id, video_id, likes_count, reactions:[{user_id, type, timestamp}] }	video_id
Views	Log view events for analytics	{ _id, video_id, view_events:[{user_id, duration, timestamp}] }	video_id, view_events.timestamp
Sessions	Sequence of user actions per session	{ _id, user_id, session_id, events:[{type, video_id, timestamp}] }	user_id, session_id
FeedCache	Cached personalized feed	{ _id (as user_id), feed:[{video_id, rank, score}], last_updated}	_id
AdImpressions	Log ad impressions/clicks for campaign analytics	{ _id, campaign_id, advertiser_id, user_id, video_id, timestamp, type, device, location, duration_viewed, interacted}	campaign_id, user_id
SystemMetrics	Technical system logs (latency, errors, etc.)	{ _id, timestamp, metric, value, endpoint, status, server_id}	timestamp, metric

TABLE II
NOSQL DATABASE COLLECTIONS