Experiment 4

Student Name: Astik Joshi
UID: 23BCS10627
Branch: CSE
Section/Group: KRG 3-B
Semester: 6th
Date of Performance:04/02/2026
Subject Name: System Design
Subject Code: 23CSH-314

1. Aim:

To design and analyze a **scalable OTT (Over-The-Top) video streaming platform** similar to **Netflix / Amazon Prime**, which allows users to register, subscribe, search, and stream video content efficiently while ensuring **high availability, low latency, and massive scalability** using modern distributed system concepts.

2. Objective:

- To determine and document both functional and non-functional requirements of a large-scale OTT platform
- To create a High-Level Design (HLD) representing distributed system components
- To study adaptive bitrate streaming mechanisms using HLS and DASH protocols
- To evaluate CAP theorem trade-offs in real-world streaming systems
- To analyze the usage of Kafka, CDN, object storage, encoding workflows, and video segmentation
- To understand why distributed streaming systems prioritize availability over strict consistency

3. Tools:

- Draw.io – For designing system architecture diagrams
- MySQL – To manage structured data such as users, subscriptions, and billing
- MongoDB (NoSQL) – For storing video-related metadata
- ElasticSearch – To implement high-performance search capabilities
- Apache Kafka – For event-driven asynchronous communication
- Redis / CDN Cache – For reducing latency through caching
- Cloud Object Storage (S3-like Blob Storage) – For storing raw and processed video files

4. System Requirements:

A. Functional Requirements:-

- Users must be able to create and manage their accounts.
- Secure login and session handling should be supported.
- Users should be able to purchase and manage subscription plans.
- The platform must provide search functionality based on keywords or titles.
- Videos must be available in multiple quality formats (480p to 4K).
- The system should automatically adjust video quality according to network speed.
- Users must be able to view video details including thumbnails and descriptions.
- (Optional) Personalized recommendations may be generated using watch hist

B. Non-Functional Requirements:-

1. Scalability

- Expected user base: 200–300 million active users
- Content library: Approximately 20,000 videos (~1 hour each)
- Infrastructure must support millions of simultaneous video streams

2. Availability vs Consistency (CAP Consideration)

- The system emphasizes **high availability over strict consistency**.
- Video playback must remain uninterrupted even if minor system components fail.
- Minor delays in metadata synchronization are acceptable.
- Financial transactions (subscriptions/payments) require strong consistency guarantees.

3. Performance & Latency

- Target response latency: 50–80 milliseconds
- Playback must start quickly with minimal or zero buffering

5. High Level Design (HLD):

The architecture follows a **microservices-based distributed model**:

API Gateway

- Central entry point for all client requests
- Handles routing, authentication, authorization, and request throttling

User Management Service

- Manages registration and login
- Uses JWT tokens for authentication
- Stores user records in MySQL

Subscription & Billing Service

- Processes payments and manages plans
- Ensures ACID properties for transaction safety

Search Service

- Integrates ElasticSearch for real-time search results

Video Metadata Service

- Stores content information (title, genre, description, thumbnails)
- Uses NoSQL database for scalability

Video Ingestion & Processing Pipeline

- Uploaded videos are saved in object storage
- A Chunking Service divides videos into smaller segments
- Encoding Service converts videos into multiple resolutions
- Apache Kafka coordinates asynchronous communication between pipeline components

Streaming Service

- Generates streaming manifests (.m3u8 for HLS / .mpd for DASH)
- Delivers adaptive bitrate streams to clients

CDN Layer

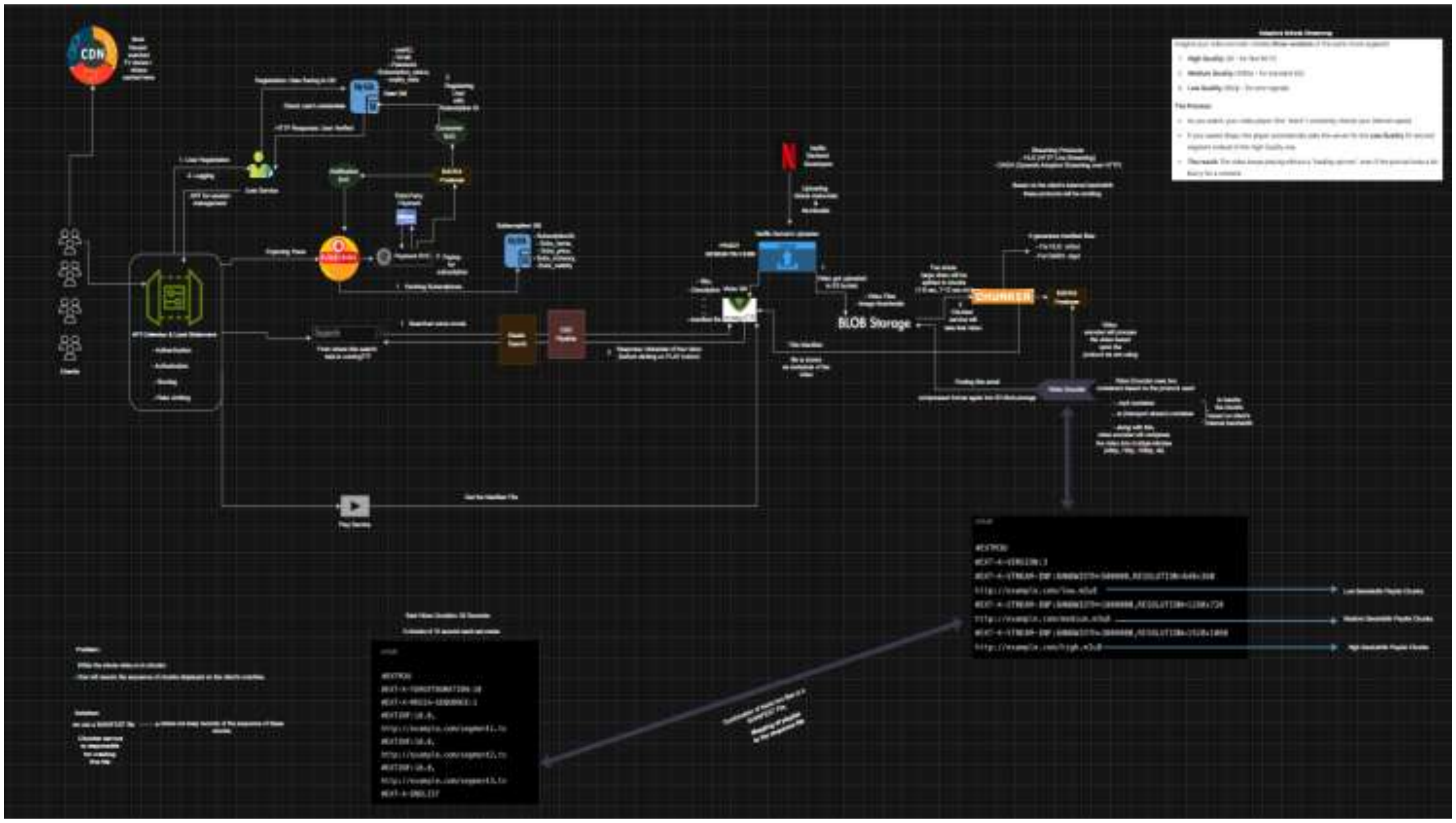
- Caches popular video chunks closer to users
- Reduces load on origin servers and improves performance

6. Video Streaming Process Flow:

- User selects a video and clicks play.
- The client requests the manifest file.
- The manifest contains URLs for different quality segments.
- Client evaluates current bandwidth conditions.
- Suitable resolution segments are fetched dynamically.
- CDN serves cached segments when available.
- Adaptive bitrate logic ensures smooth playback without interruption.

7. Scalability Solution:

- All microservices are horizontally scalable.
- Load balancing is managed via API Gateway.
- Kafka ensures decoupling between processing components.
- CDN reduces repeated origin fetch requests.
- Blob storage provides efficient large-file storage.
- Services are stateless to allow easy replication and scaling.



```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-STREAM-INF:BANDWIDTH=500000,RESOLUTION=640x360
http://example.com/low.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=1000000,RESOLUTION=1280x720
http://example.com/medium.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=3000000,RESOLUTION=1920x1080
http://example.com/high.m3u8
```

```
#EXTM3U
#EXT-X-TARGETDURATION:10
#EXT-X-MEDIA-SEQUENCE:1
#EXTINF:10.0,
http://example.com/segment1.ts
#EXTINF:10.0,
http://example.com/segment2.ts
#EXTINF:10.0,
http://example.com/segment3.ts
#EXT-X-ENDLIST
```

7. Learning Outcomes (What I Have Learnt):

- Gained practical understanding of OTT platform architecture
- Distinguished clearly between functional and non-functional requirements
- Understood adaptive bitrate streaming techniques
- Applied CAP theorem concepts in distributed design
- Learned the interaction between Kafka, CDN, and encoding workflows
- Developed insight into designing highly scalable, fault-tolerant systems