# Experiment-4

**Student Name:** Sujal Srivastava **UID:** 23BCS11842
**Branch:** CSE **Section/Group:** KRG-3B
**Semester:** 6th **Date of Performance:** 04/01/26
**Subject Name:** System Design **Subject Code:** 23CSH-314

1. **AIM :** To design a scalable OTT (Over-The-Top) video streaming platform similar to Netflix or Amazon Prime Video that allows users to register, subscribe, search, and stream video content efficiently using modern system-design concepts such as microservices, CDN, Kafka, Elasticsearch, and adaptive bitrate streaming.

2. **Objective:**

   • To design a high-availability video streaming system for millions of users
   • To support user registration, login, and subscription management
   • To implement video search using Elasticsearch
   • To enable adaptive video streaming (480p, 720p, 1080p, 4K)
   • To use Apache Kafka for asynchronous communication
   • To ensure low latency and high scalability

3. **Tools Required:**

   • Programming Language: Java / Python / JavaScript
   • Backend Framework: Spring Boot / Express.js / Flask
   • Database: MySQL / PostgreSQL / MongoDB
   • Object Storage: AWS S3 (Blob Storage)
   • Message Broker: Apache Kafka
   • Search Engine: Elasticsearch
   • CDN: CloudFront / Akamai
   • API Testing Tool: Postman

4. **SYSTEM DESIGN / SYSTEM SPECIFICATION:**

   **4.1. Functional Requirements:**

- User should be able to register and login
- User should be able to choose subscription plans
- User should be able to search movies / TV shows by title
- User should be able to view video metadata (thumbnail, description)
- User should be able to play videos in multiple resolutions
- User should receive recommendations based on watch history

### 4.2. Non-functional Requirements:

- Target Scale: 200–300 Million users
- Video Count: ~20,000 videos (avg. 1 hour each)
- Scalability: Horizontal scaling
- Latency: 50–80 ms
- Availability: Very High (video playback must not fail)
- Consistency: Strong consistency for payments & subscriptions

**CAP Theorem Decision:**

- Availability >>> Consistency for video streaming
- Consistency > Availability for payment & subscription

### 4.3. Core-Entites of the System:

1. User / Client
2. User Metadata
3. Subscription
4. Video / TV Show
5. Video Metadata (thumbnail, description, manifest file)
6. Payment
7. Notification

### 4.4. API Endpoints Creation:

**User APIs:**

1. POST – User Registration

`https://www.netflix.com/user/register`

2. POST – User Login

`https://www.netflix.com/user/login`

3. PUT – Update User Details

`https://www.netflix.com/user/update`

Subscription APIs

4. GET – Fetch Subscription Plans

`https://www.netflix.com/get-subscription-plans`

5. POST – Subscribe Plan

`https://www.netflix.com/subscription`

Search & Play APIs

6. GET – Search Movies / Shows

`https://www.netflix.com/search?q={movie_name}`

7. GET – Get Video Metadata

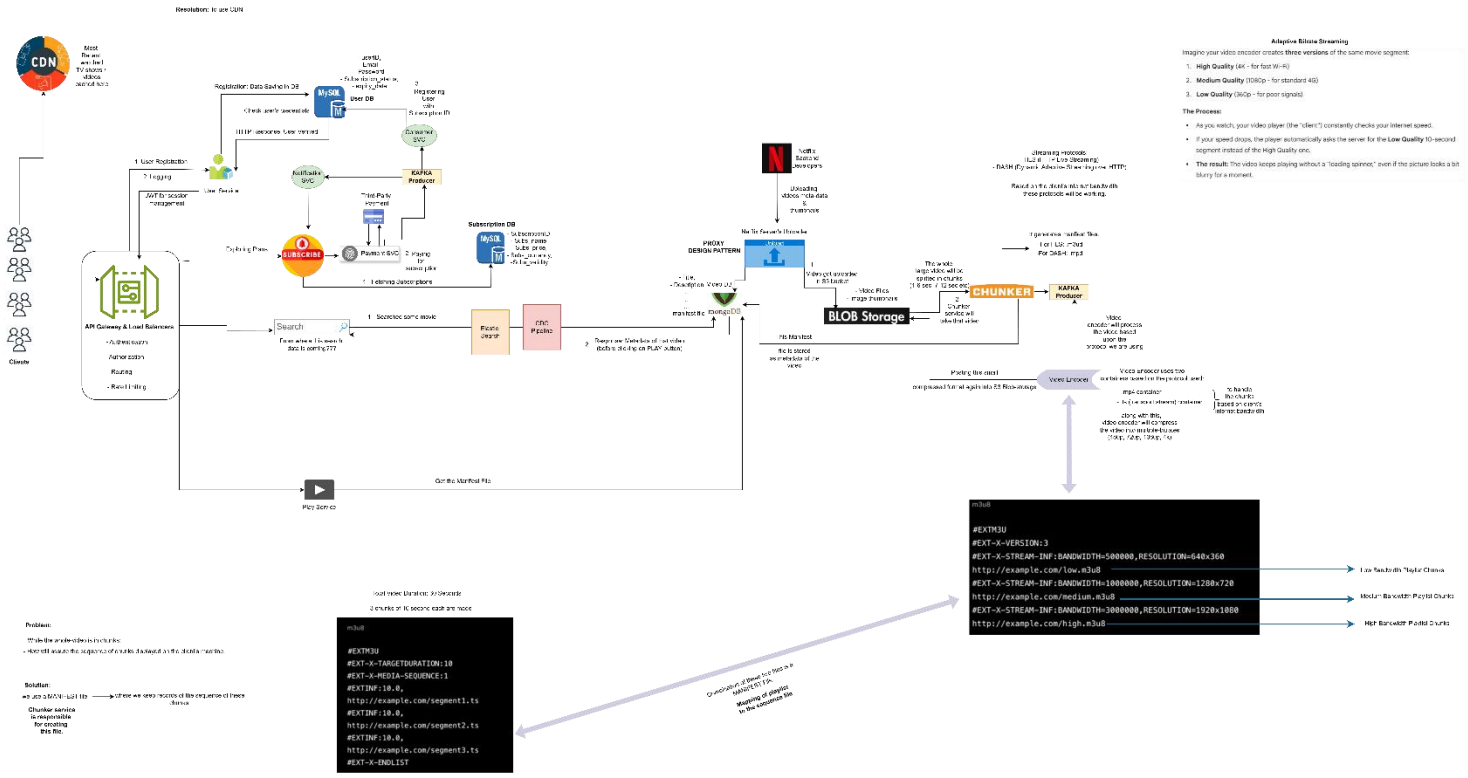
`https://www.netflix.com/{video_id}`

8. GET – Play Video

`https://www.netflix.com/play/{video_id}`

## 5. HLD(High Level Design):

We have to follow a distributed / micro-services approach not the monolithic one.




## 7. Learning Outcomes

- Learned how OTT platforms stream videos at scale
- Understood adaptive bitrate streaming
- Gained knowledge of Kafka & Elasticsearch
- Learned CDN importance in video delivery
- Understood real-world system design trade-offs