



Experiment No: 4

Student Name: Aditya Singh
Branch: BE CSE
Semester: 6th
Subject Name: System Design

UID: 23BCS11827
Section/Group: KRG-2_B
Date of Performance: 04/02/2026
Subject Code: 23CSH-314

1- Aim - To design an scalable OTT platform (Similar to Netflix or Amazon Prime)

2- Requirements: Functional & Non-Functional

A- Functional Requirement

- Client should be able to create account on the OTT platform.
- After the successfull login, client should be able to opt for the subscription plans.
- Client should be able to search for the shows/movies based on the video title or names.
- Client should be able to watch the videos / tv shows in multiple different resolutions (480p,720p,1080p, 4k etc.)
- Recommendation for TV shows and movies.

B- Non-Functional Requirement

- Scalability: 200-300M, for which let's say total videos we are having are 20K videos (~1 hour each)
- Consistency & Availability: Availability >>>>> Consistency.
Availability on watching TV shows and movies and Consistency in making payments and in subscription plans.
- Latency: 50 - 80 ms
Client should be able to see the video with zero or neglibile buffering.

3- Core-entities of System

- Clients
- Clients metadata
- Video / TV shows
- Video metadata: (images(Thumbnails + description)

4- API endpoint creation

1. Client-Onboarding

1. POST Call: <https://www.netflix.com/user/register>
2. POST Call: <https://www.netflix.com/user/login>
3. PUT Call: <https://www.netflix.com/user/update>
4. User Data Update: PUT API CALL: PUT / api / users / {user_id} / profile

2. Subscription

1. GET Call: <https://www.netflix.com/Get-subscription-plans>
2. POST Call: <https://www.netflix.com/subscription>

```
{
    userMetadata,
    subscriptionID
}
```

3. Searching & Video Playing

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

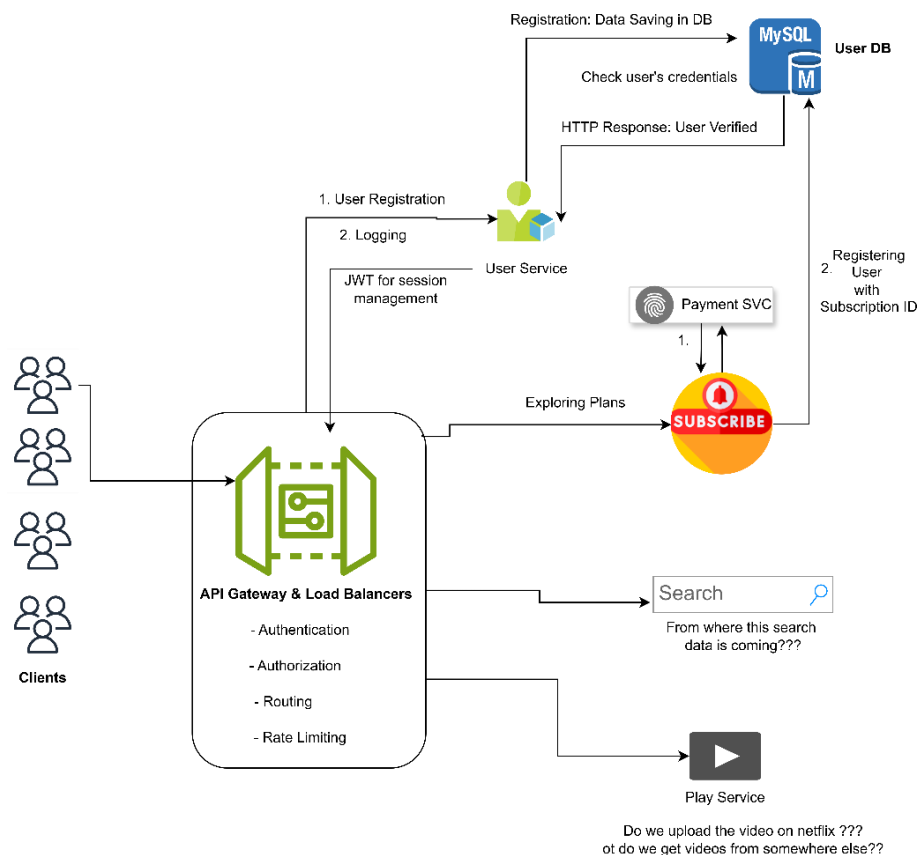
Response: List<Video_ID> + some meta data of video

2. GET Call: `https://www.netflix.com/{video_ID}`

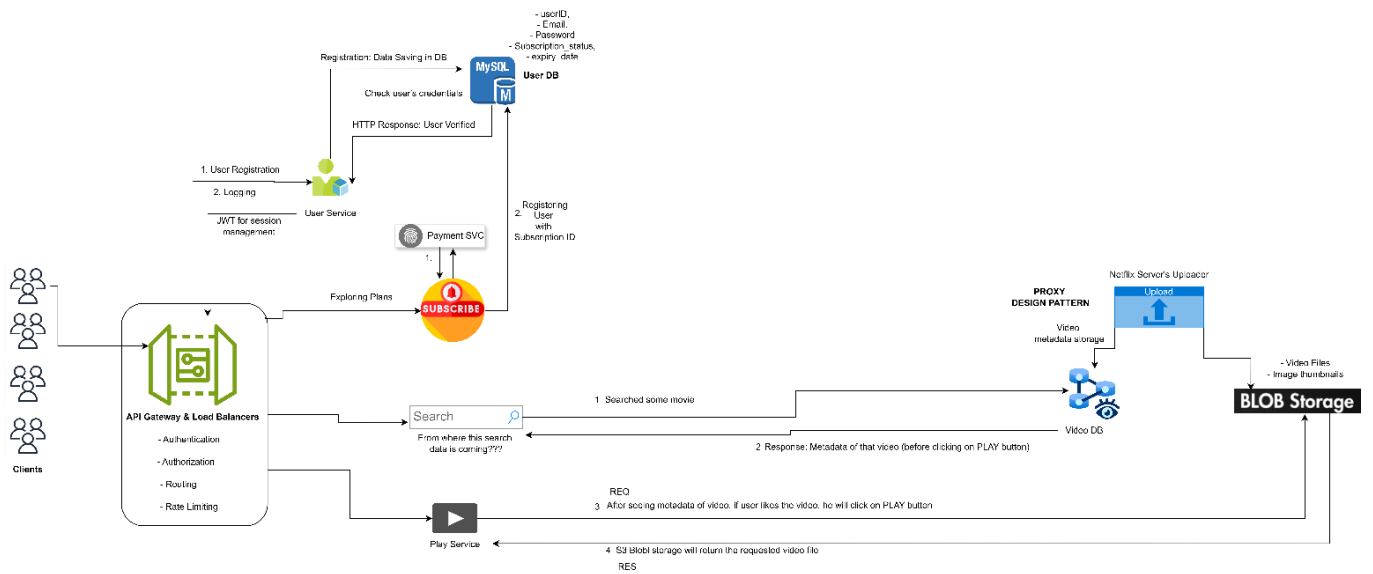
Response: Metadata of the video (JSON)

3. GET Call: `https://www.netflix.com/play/{video_ID}`

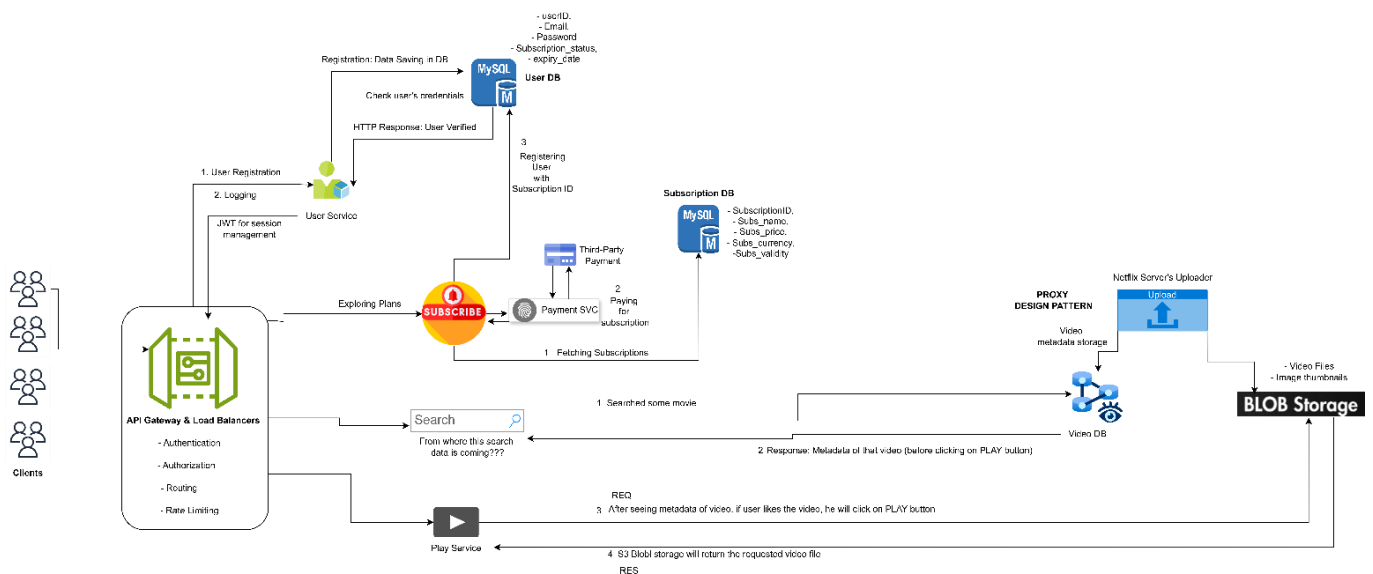
5- High-Level Design



6- Low-Level Design

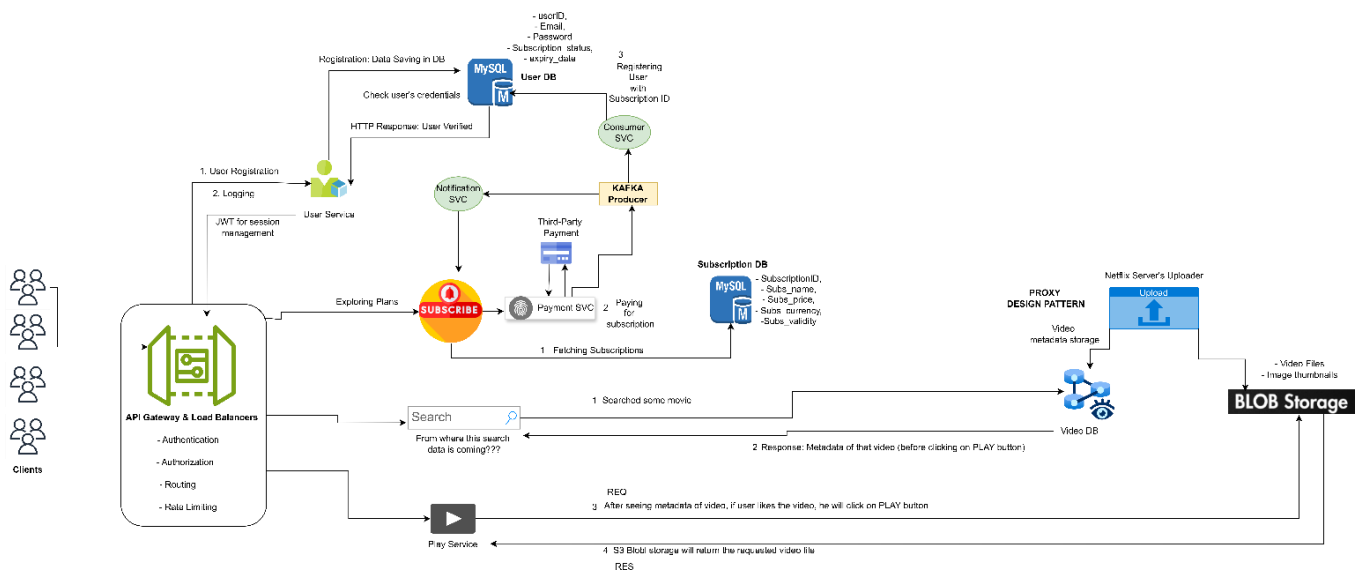


Subscription Service

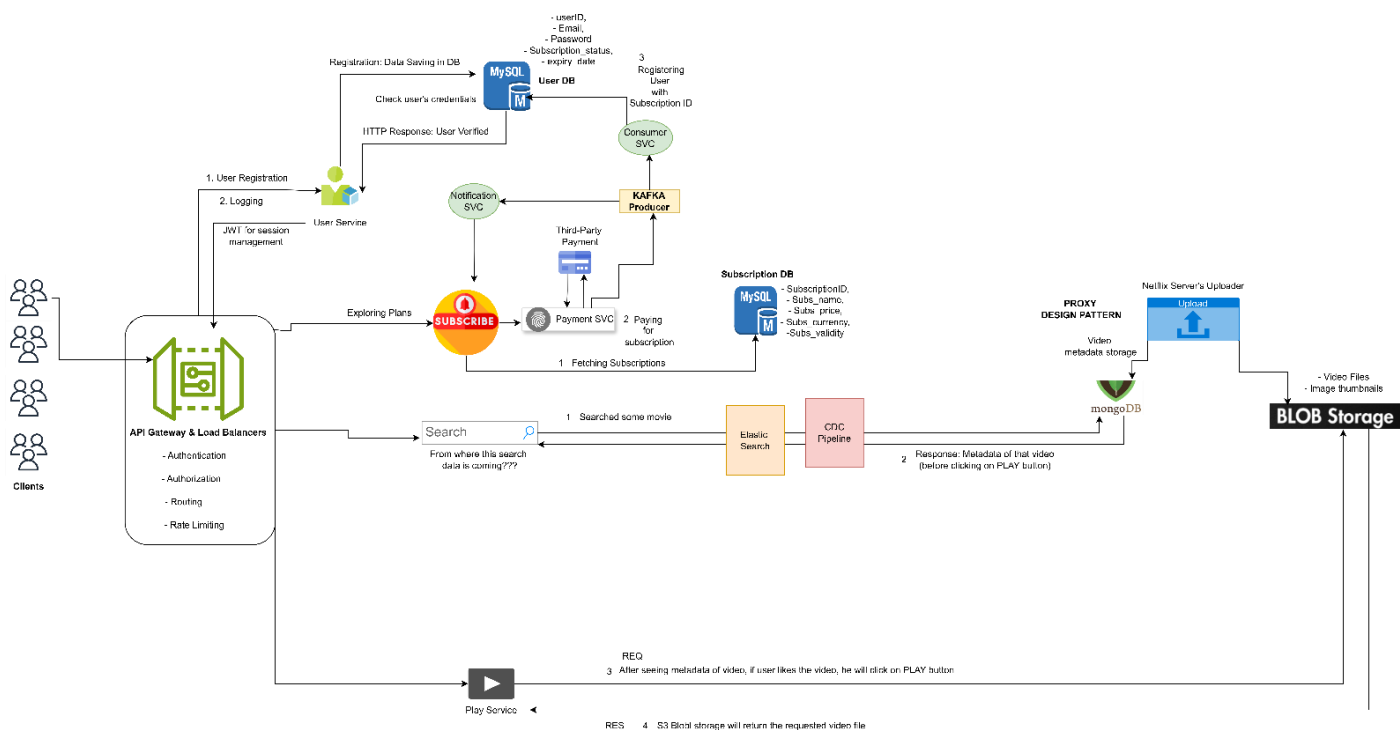


Problem:

- To many API calls with Subscription Service
- Solution is to use KAFKA (P-C Architecture)



Search Service



Play Service

For play service

- Does the whole video of 5-10 GB loads prior in your OTT???
- If we change the resolution then how it changes everything in backend.

NOTE: If you have observed, while watching any movie on OTT, there can be chances that on first-load you get blurry video for 5-6 sec, after that when internet bandwidth is good, you see 720p automatically.

before the video is displayed on our devices, it has to go from 'n' no of operations.

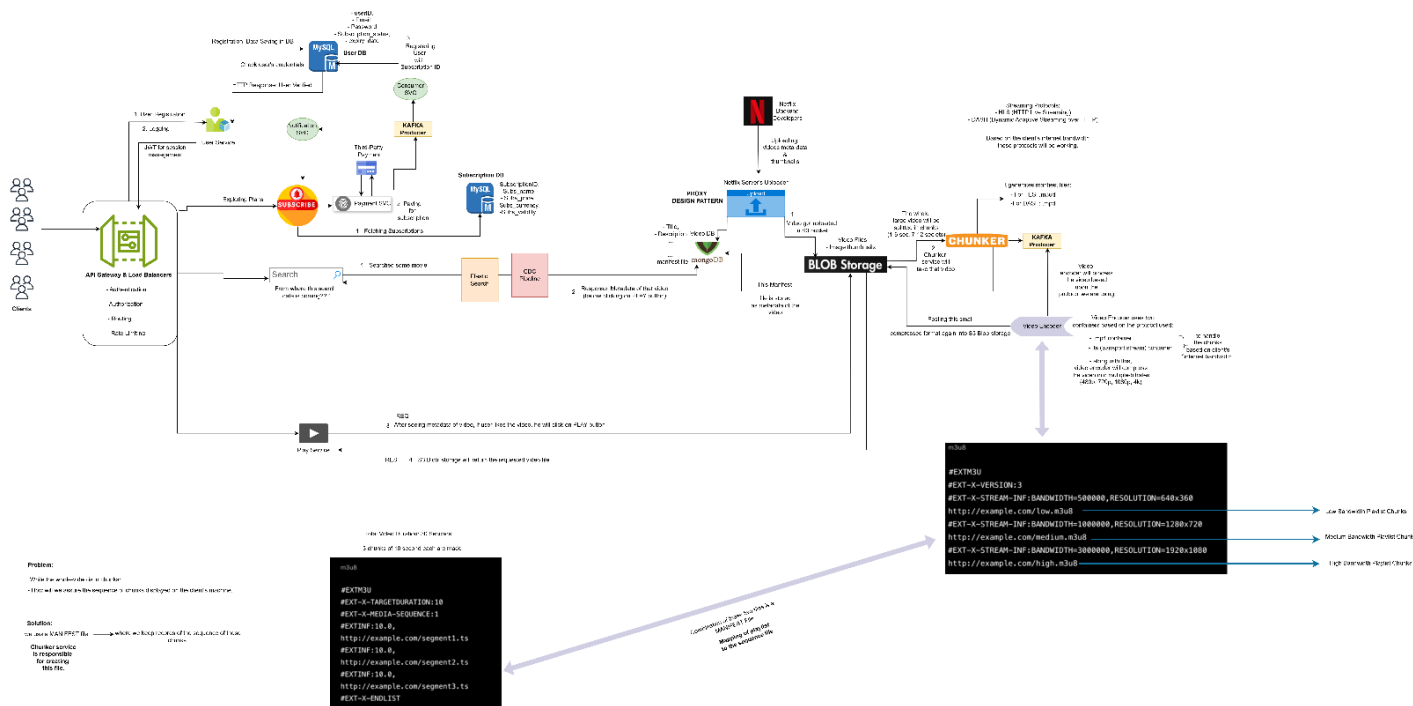
Imagine your video encoder creates **three versions** of the same movie segment:

1. **High Quality** (4K - for fast Wi-Fi)
2. **Medium Quality** (1080p - for standard 4G)
3. **Low Quality** (360p - for poor signals)

The Process:

- As you watch, your video player (the "client") constantly checks your internet speed.
- If your speed drops, the player automatically asks the server for the **Low Quality** 10-second segment instead of the High Quality one.
- **The result:** The video keeps playing without a "loading spinner," even if the picture looks a bit blurry for a moment.

Adaptive bitrate streaming



Now, Play Service will get manifest file from Video DB and will give back to the client. This manifest file will calculate the bandwidth on client side, and the video / show based upon the bandwidth will be fetched from S3 storage. But if, you observe clearly, there is a huge gap btw Clients and S3 to fetch the video / shows. **Resolution:** We'll use CDN

