DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
Discover. Learn. Empower.

Experiment No:4

Student Name: Sumanyu Singh
UID:23BCS10801
Branch: BE CSE
Section/Group:KRG-3B
Semester: 6th
Date of Performance:04/02/2026
Subject Name: System Design
SubjectCode:23CSH-314

Aim: To design an scalable OTT platform (Similar to Netflix or Amazon Prime).

Objective:

- To understand the architecture of large-scale OTT video streaming platforms.
- To analyze system components required for delivering uninterrupted video playback.
- To define functional and performance requirements for millions of concurrent viewers.
- To design service communication using APIs within a microservices environment.
- To ensure scalability, low buffering latency, and high availability for global users.

Tools and Technologies Used:

- Architecture diagram tools such as Draw.io or Lucidchart for system visualization.
- Programming languages including Python, Java, or JavaScript for backend services.
- MySQL database for storing user profiles and subscription data.
- MongoDB or NoSQL storage for video metadata management.
- Elasticsearch for high-speed content searching.
- Apache Kafka for asynchronous video processing workflows.
- Redis Cache or CDN for reducing latency during streaming.
- Blob Storage (S3 equivalent) for storing large video files.
- Development environments such as VS Code and modern web browsers.

Functional Requirements :

- Users must be able to register accounts and securely log into the platform.
- The system should allow users to subscribe to different streaming plans.
- Users should search movies or TV shows using keywords or categories.
- Video playback must support multiple quality levels such as 480p, 720p, 1080p, and 4K.
- Adaptive bitrate streaming should automatically adjust video quality according to network speed.
- Users should view thumbnails, trailers, and metadata before streaming.
- The system may provide personalized recommendations based on viewing history.

Non - Functional Requirements :

- **Scalability**
  The platform should efficiently support hundreds of millions of users and millions of simultaneous streaming sessions through horizontal scaling and distributed services.
- **Availability and Consistency (CAP Perspective)**
  High availability is prioritized over strict consistency. Video playback must continue smoothly even if minor delays occur in metadata updates or recommendation synchronization. However, payment transactions require strong consistency.
- **Latency**
  Video loading time and playback startup delay should remain minimal, ideally within 50–80 milliseconds, ensuring reduced buffering and uninterrupted streaming

Core-Entites of the System:

The major data elements involved in the OTT streaming system include:

- User accounts and authentication details
- Subscription plans and payment transactions
- Video metadata including titles and descriptions
- Video media files and encoded chunks
- Streaming manifest files (HLS/DASH)
- Watch history and playback progress
- Recommendation data

API Designing (Low Level Design – LLD)

1. POST API – User Registration

https://localhost/users/register

- Allows new users to create an OTT account.
- Stores credentials and profile information in the user database.
- Validates email and password before account creation.

2. POST API – User Login

https://localhost/users/login

- Authenticates registered users securely.
- Generates JWT authentication tokens for accessing services.

3. GET API – Browse or Search Content

https://localhost/content/search?query={movie_name}

- Enables users to search movies or TV shows.
- Uses search indexing for faster response.
- Returns titles, thumbnails, and metadata.

4. GET API – Video Metadata Details

https://localhost/content/{video_id}

- Fetches information such as description, cast, duration, and thumbnail.
- Retrieves metadata stored in NoSQL databases.

5. POST API – Subscription Purchase

https://localhost/subscription/subscribe

- Allows users to select and activate subscription plans.
- Integrates with payment validation service.
- Ensures transactional consistency during purchase.

6. POST API – Payment Processing

https://localhost/payment/process

- Handles online payment authorization.
- Updates subscription status after successful payment.

7. GET API – Streaming Manifest Request

https://localhost/stream/{video_id}/manifest

- Generates HLS or DASH manifest files (.m3u8 or .mpd).
- Provides different bitrate URLs for adaptive streaming.

8. GET API – Fetch Video Chunk

https://localhost/stream/chunk?segment_id={segment_id}

- Client downloads small video segments dynamically.
- CDN caching improves playback speed and reduces latency.

9. GET API – Recommendation Feed

https://localhost/recommendations?user_id={user_id}

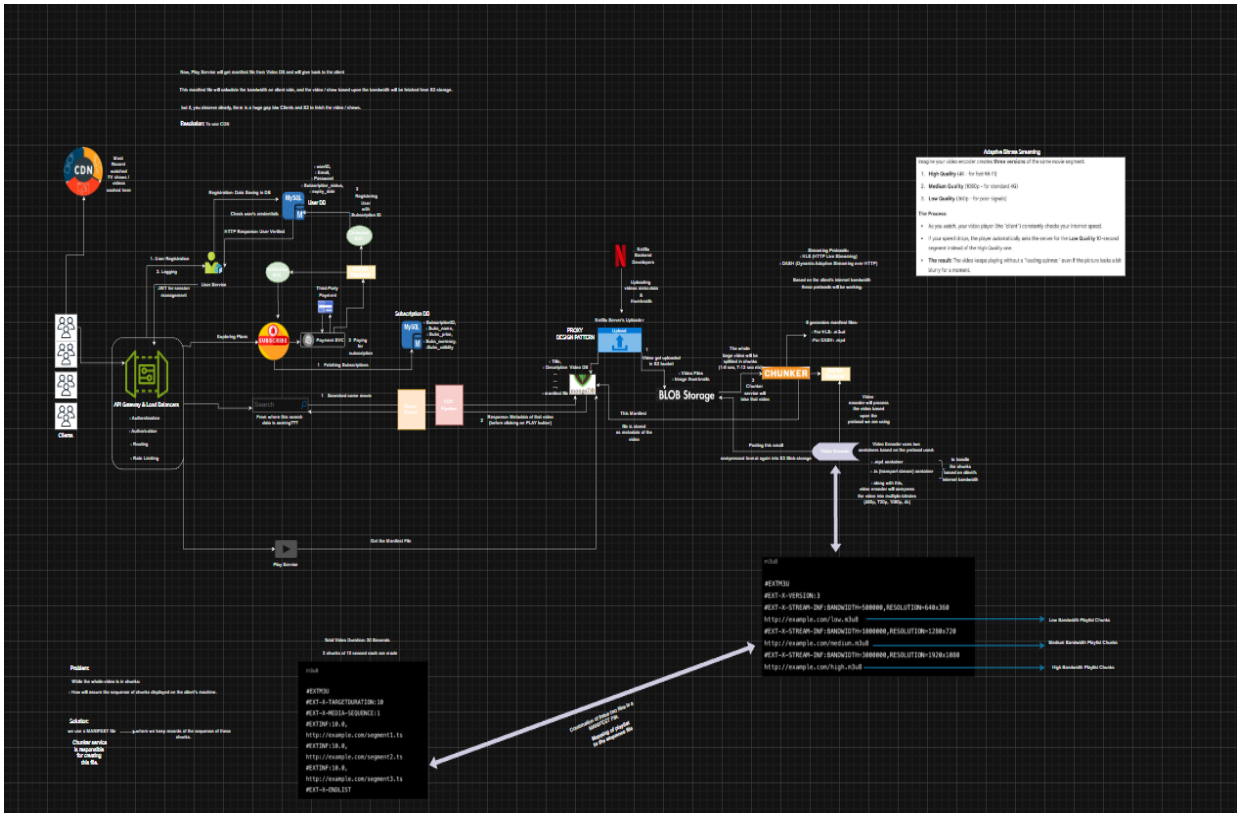
- Provides personalized content suggestions.
- Based on watch history and viewing patterns.

10. GET API – User Watch History

https://localhost/users/watch_history

- Retrieves previously watched content.
- Used for resume playback and recommendations.

HLD(High level Diagram) -> FINAL :



Learning outcomes:

- Understanding real-world OTT platform architecture.
- Learning adaptive bitrate streaming concepts using HLS and DASH protocols.
- Understanding the role of CDN and caching in reducing buffering delays.
- Gaining knowledge about distributed event processing pipelines.
- Developing skills in designing scalable and highly available systems.