**Created By: Siddhant Jain**

**Email: siddhantjn11@gmail.com**

***VoD Requirements:***

1. Should be able to upload videos

2. Overlaid subtitle on videos

3. Viewers should be able to view content with varying screen size and internet speed

4. Encrypt videos

5. Video search functionality

6. Authenticate users

uploads the raw video file

Storage service



Content uploader



Segment video file into chunks

* Store the segmented file into the file storage system (AWS S3, reliable & cheaper)
* Encode each segment into different codec (Cinepak, MPEG-2, VP8) and resolution(240p,360p,480p). It would help with viewers having different Internet speed.

1. URLs will be stored in database schema.
2. Should distribute encoded file segments on CDN (Content distribution network)

***Video metadata***

|  |
| --- |
| **VideoID (PK)** |
| Title |
| Description |
| Cdn\_urls |
| Creator |

* ***Overlaid subtitles***

OPEN TSDB (Time Series Database)

“events”: [

{

StartTime: T0

EndTime: T1

Metadata: {

Subtitle: “This is some subtitle”

}

}

]



HTTP content play Asynchronous Request

Playback service

Hit play button



Authenticate user’s request



Request the content from the CDN (Stream requested MEDIA file)

Pick best CDN URL for content playback



* ***Video search feature:***

Make search request ( in content discovery service which make request to AWS ES)

Use AWS ElasticSearch Service (we can store all keywords associated with video title)

Can also fetch details of similar titles (If video title doesn’t exist in search-index)



***Capacity Planning:***

Since there is a heavy load, we need **horizontal scaling.** Many web servers will be handling the requests. In doing this we need to have a load balancer, which will distribute the request among the servers.  
This approach gives us a flexibility that when the load increases, we can add more web servers to handle the increased load.

***Authenticate Users:***

1. Register New Users
2. Send confirmation email

Encryption is bijective but hashing is not. Hence, we must store hash password in database.

A random string can also be appended in the password hash. We can avoid SQL injection attack by using an appropriate table name.

***“User\_login\_data” schema***

|  |
| --- |
| UserID |
| LoginName |
| PasswordHash |
| PasswordRandomString |
| HashAlgoID |
| EmailAddress |
| StatusID |

***Send Confirmation Email:***

The way to do this email verification is through a random and unique token, long enough so that it cannot be guessed by brute force in limited time.

***Email\_validation\_status schema:***

|  |
| --- |
| StatusID |
| StatusDescription |

***Encryption of videos using multi-DRM: (digital rights management):***

For users who have the right to use the content, the ‘encryption key’ and ‘usage right information’ will be transmitted separately from the content.

This will help in protecting videos from unauthorized users.

**DRMs**: PlayReady, Widevine Modular and FairPlay Streaming (These DRMs Also supports HTTP live streaming )