Arquitetura de Software

YouTube System Design

Architectural Requirements

- Functional Requirements:
  - User Features:
    1. An authenticated user can upload a video with various resolutions, such as 360p, 480p, 720p, 1080p, 4k and formats such as .avi, .mov or .mp4 with a maximum size of 1GB, these videos during the upload process are encoded in possible encoding formats, like container that support various formats already referred and codecs like H.264, VP9 and HEVC, and persistently store the results of the encoding so it can make the video available for the end user in the different formats and resolutions of different user device conditions.
    2. An authenticated user during the upload of a video must have his metadata that contains the file name, size and format supported by the system in parallel to make the upload process much faster.
    3. A user can watch videos and change their formats and resolutions dynamically depending on the uploaded video, the video when requested to be streamed will be available immediately by loading a small portion of it, making it streamed continuously.

- Non-Functional Requirements:
    - Usability:
        1. A user from a specific part of the world wants to stream a very popular video that was uploaded in a different region of the world and this user needs to have the same viewing experience as a user that is located in the same part of the world as the uploader.
        2. A user from a specific part of the world wants to upload a video and have the same upload experience as another user from a different region of the world.
    - Modifiability:
        1. The user, when the video playback is not optimal for example the internet speeds are slower in the user region,the system will try to make the watching experience better for the user by changing the video quality to a lower resolution making it run smoother and still minimally enjoyable for the user while affecting the quality of the video, in this case, resolution.
        2. The user when located in a region with a high speed network wants to watch a selected video, the system will automatically stream the video in the highest resolution possible, defined during upload, supplying the best user experience possible.
        3. The developer wants to change the system for it to support 5 million users in a day without the system going into overload.
        4. A user tries to watch a video that was uploaded in a format incompatible with his current device but the system changed the video, by having different versions stored, so it can be streamed in any device.
        5. The system administrator wants to upgrade the system for it to be able to store 150 Terabytes of data daily.
        6. When a user watches a video, the system should provide a smooth video streaming experience for a minimum of 30 minutes.
        7. The developer wants the system to support up to 100.000 Users watching videos simultaneously without issues.

8. The developer wants the system to support up to 500.000 video uploads daily without the system having issues.

- Availability:

  Video Transcoding:

  1. A user tries to upload a video with a size that exceeds the size limit, a fault occurs, but the system responds promptly to the failure or bottleneck by denying the upload request and continuing to operate normally.

  2. If a service provided by a third party provider such as the CDN is not available, the system will recover in accord with the SLA defined between the service provider and the builder of the system where it should provide availability of 99%.

  3. A user requests some functionality such as upload and its respective processes, during its processing should an error be encountered, the system must recover so the process in question is successful, it may take some time depending on the network traffic and amount of requests made to the system. During this process the rest of the system should function normally.

4. A user requests some functionality such as upload and its respective processes, during its processing should an error be encountered and the system is not able to recover from it, the system should notify the user right away he can act accordingly while maintaining normal functionality for the rest of the other users.

- Security:

  1. An unauthorized user tries to upload a video,the system detects the user is not who he appears to be in less than a second and does not let him access the supposed account of the user, while the system maintains normal functionality for the other users.

  2. A user tries to plagiarize a video but the system detects that there's unauthorized usage of the video and its authenticity is verified leaving the system to block the upload request made almost immediately.

- Performance:

  1. The user requests for an upload of a video, the system should fulfill the request and after completing the upload that should take on average: for HD videos, depending on size, from 30 seconds to 1 minute all the way up to 10 minutes if it is a 10 minute video; for 4k videos from 10 minutes all the way up to 4 hours if it is a hour long. The video should be available to all users of the system independent of region so if users from different regions try to access a video, the time it takes must be short and equal for every single one.

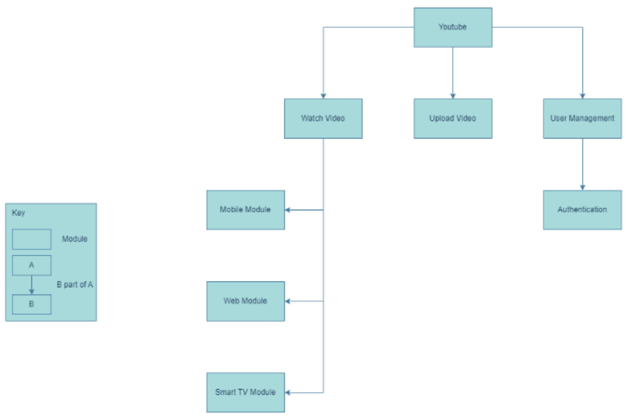
  2. A great number of users access the same video at the same time, the video should be watchable for each user simultaneously which means that if a user tries to load and watch a video, the system should fetch it with a latency of <= 1 second 90% of the time, 1 to 2 seconds 9% of the time, more than 2 seconds 1% of the time, and watches it continuously without interruption.

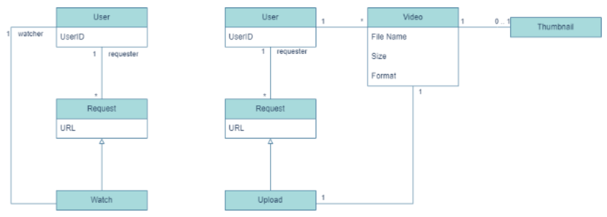
- Constraints:
    - Mobile App, Web App, Smart TV App
    - Recommendation to leverage existing cloud infrastructures provided by Amazon, Google, or Microsoft.
    - Use of CDN

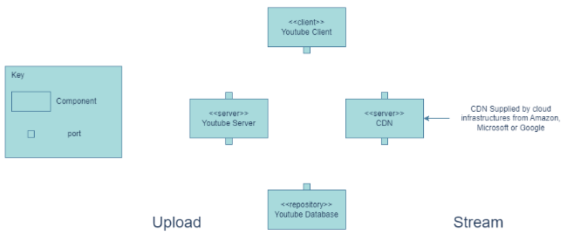
- Architectural Concern:
    - The increment in a large scale of the number of users.
    - Lower infrastructure cost.

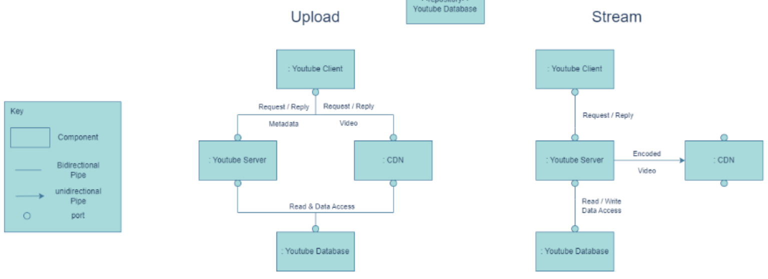
Round Definition

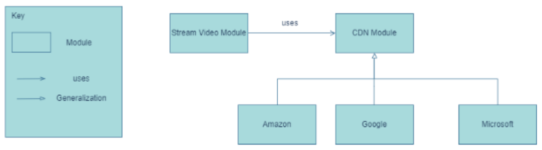
- 1º Round
  - Purpose of Design: Fully Functional System for Upload and Stream without Scalability and Availability
  - Iteration 1: Primary Functionalities (user features)(UF1 - M4, UF2, UF3,)



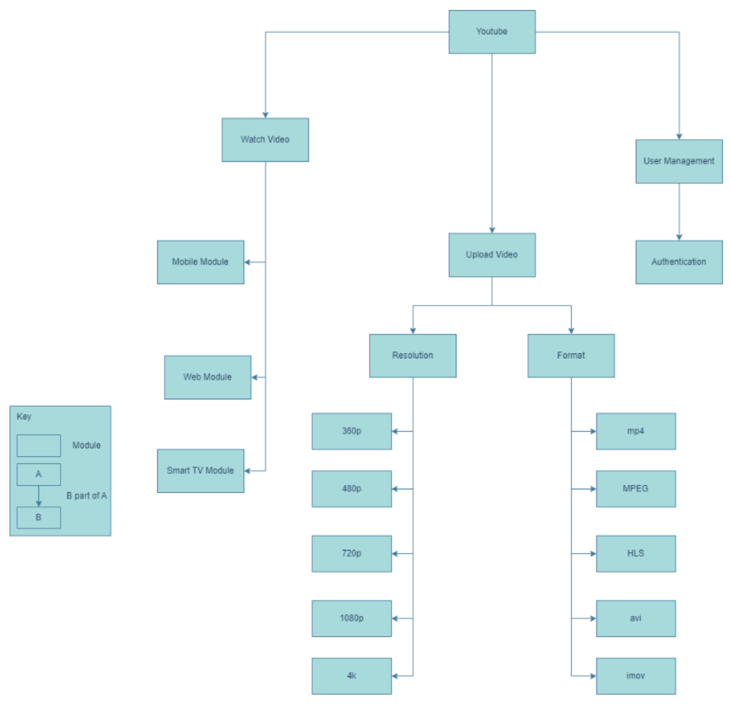


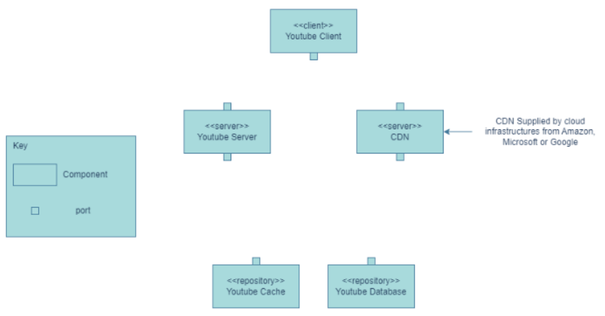


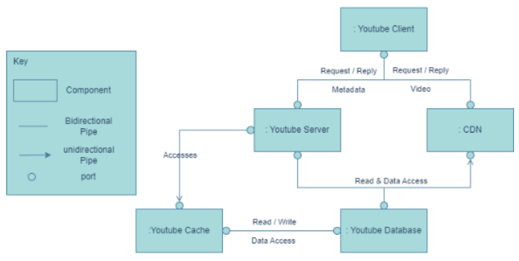


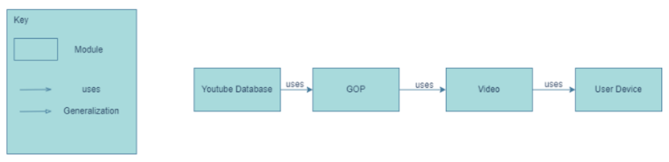


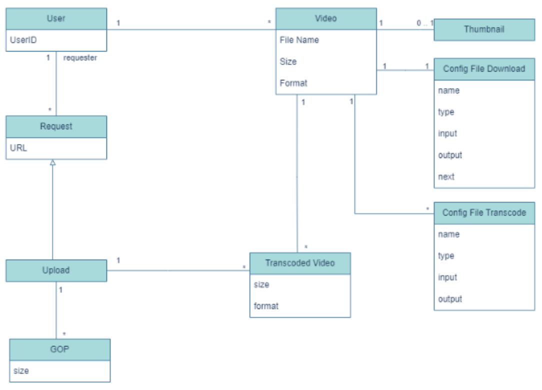
- Iteration 2: Performance for uploading videos (Preprocessor of the DAG video transcoding architecture, GOP, DAG generation, cache data) (P1, P3)



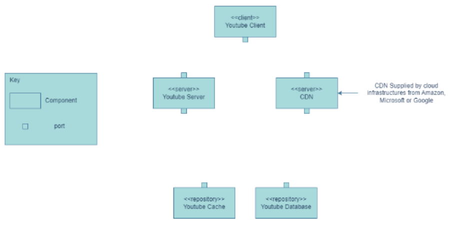


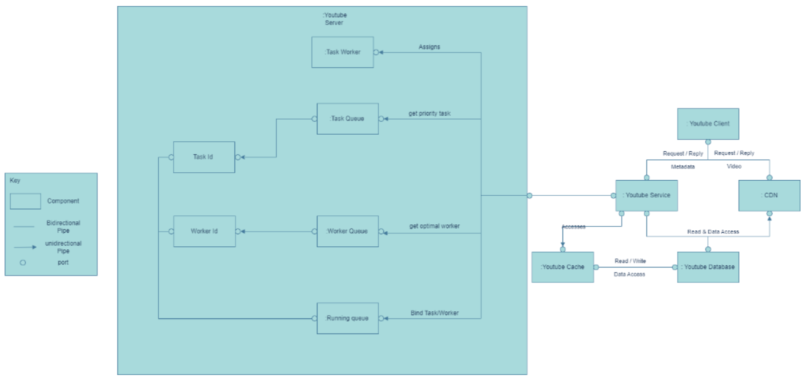




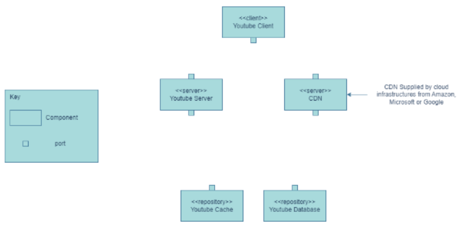


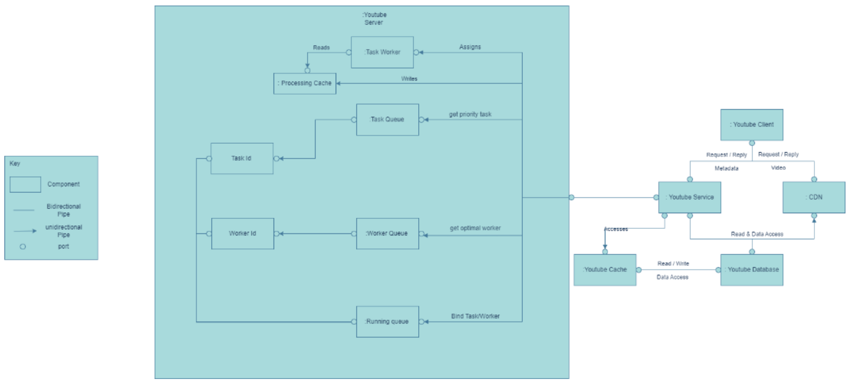
- Iteration 3: Performance of organizing tasks queues and manage resources for task workers (DAG scheduler, resource manager, tasks workers from DAG video transcoding architecture (P1)


Key
Component
port
<<client>> Youtube Client
<<server>> Youtube Server
<<server>> CDN
CDN Supplied by cloud infrastructures from Amazon, Microsoft or Google
<<repository>> Youtube Cache
<<repository>> Youtube Database

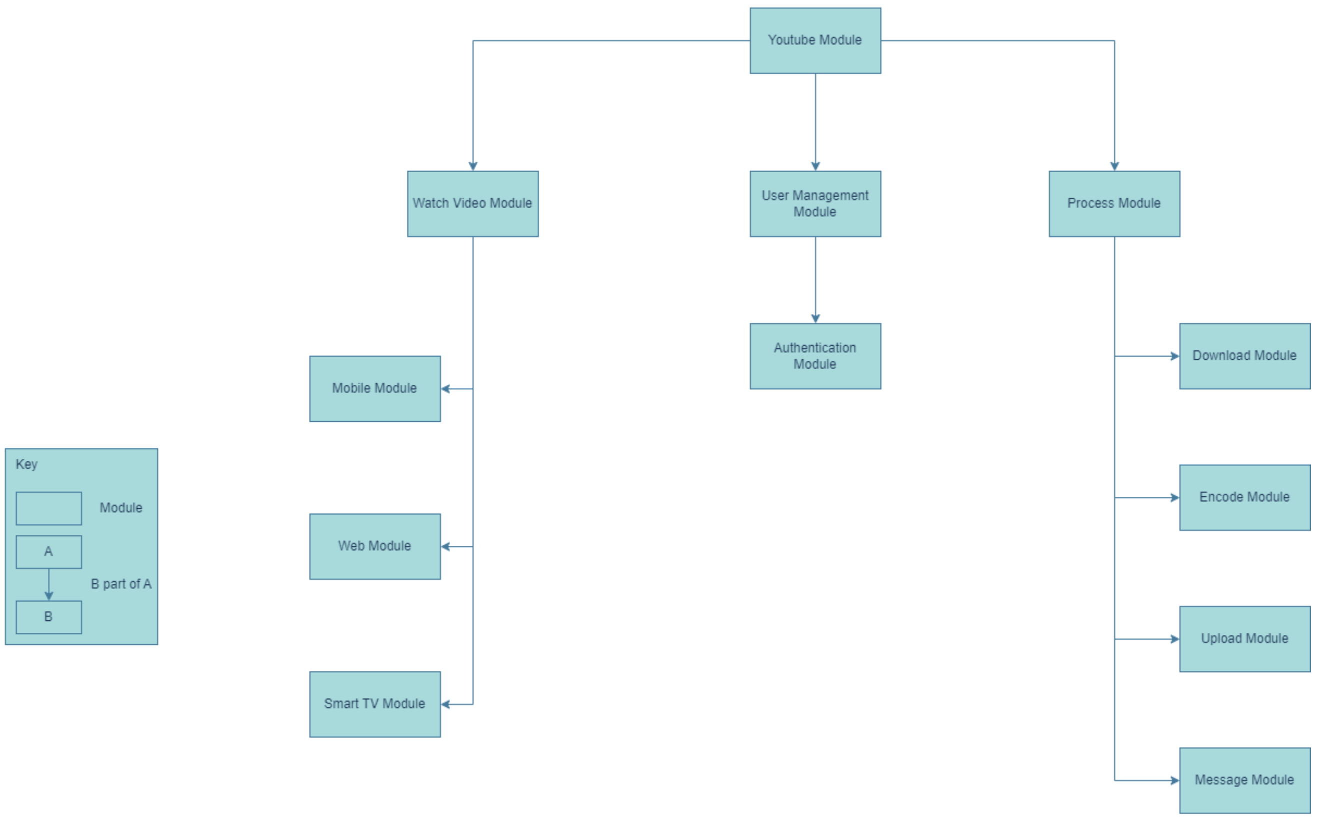


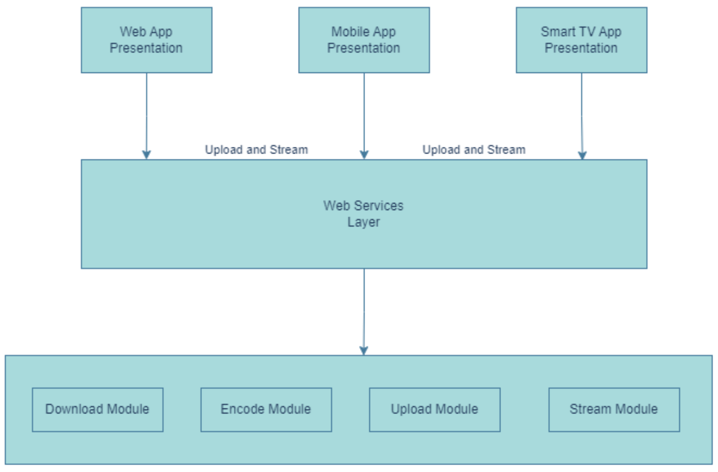
- Iteration 4: Performance for accessing storage and temporary storage (Temporary storage of the DAG video transcoding architecture) (P1)

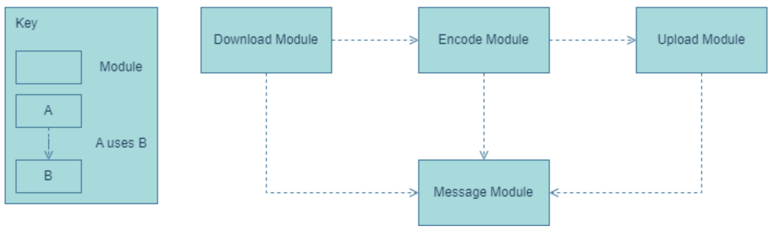

Key
Component
port
<<client>> Youtube Client
<<server>> Youtube Server
<<server>> CDN
CDN Supplied by cloud infrastructures from Amazon, Microsoft or Google
<<repository>> Youtube Cache
<<repository>> Youtube Database



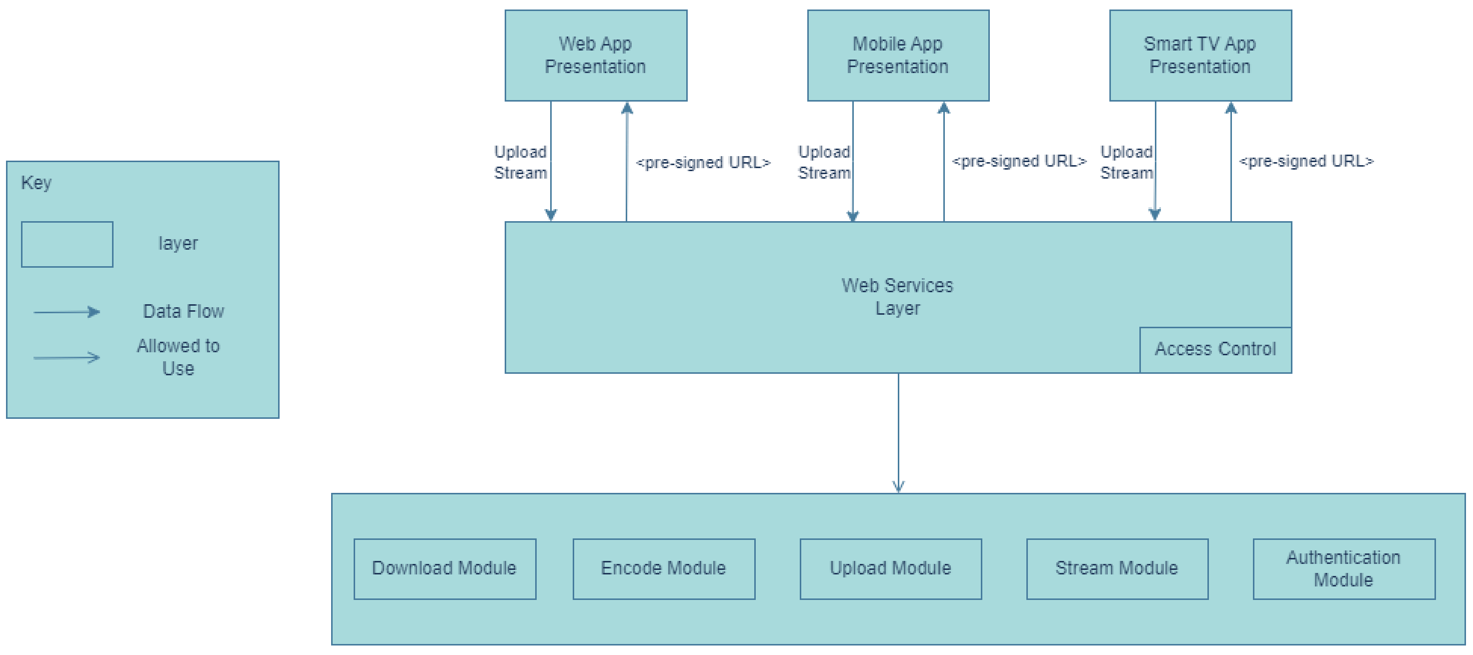
- Iteration 5: Performance of encoding videos (Encode Video from DAG video transcoding architecture, using message queues to build a loosely coupled system and enable high parallelism.) (P1)



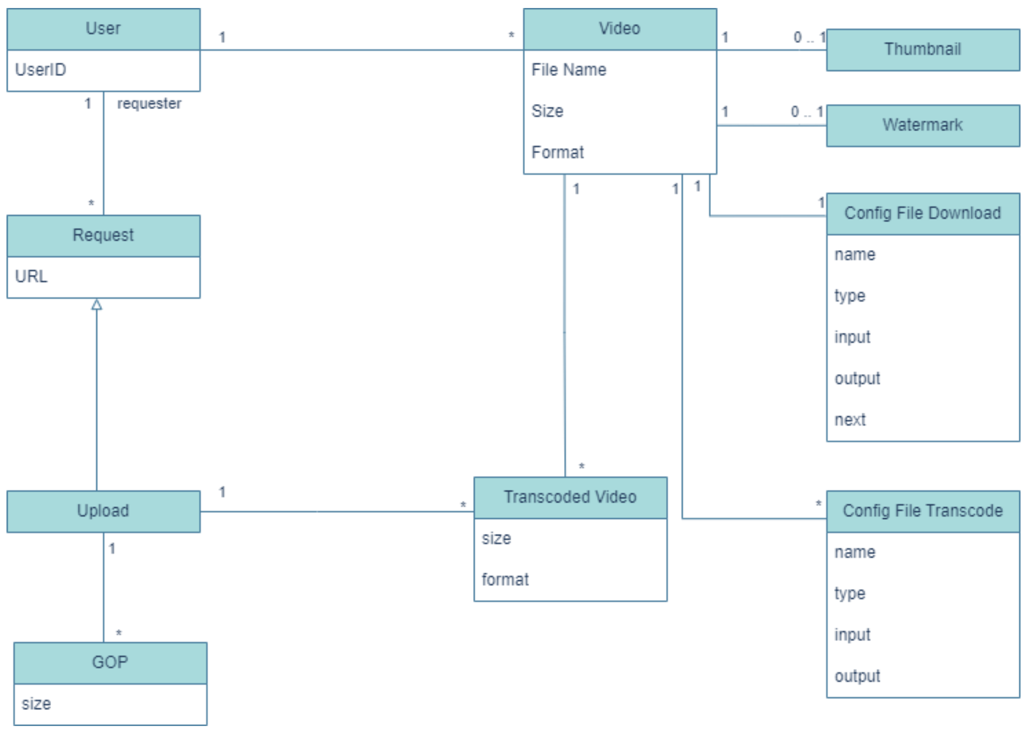




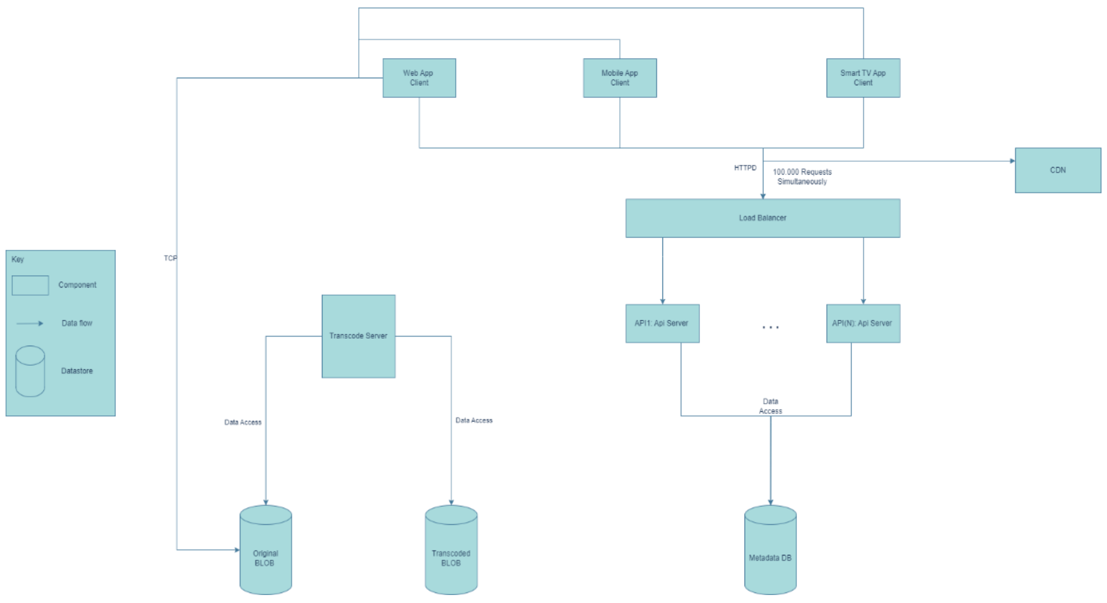
○ Iteration 6: Security for uploads (User Authentication, User request Authenticity using pre-sign URL) (S1)



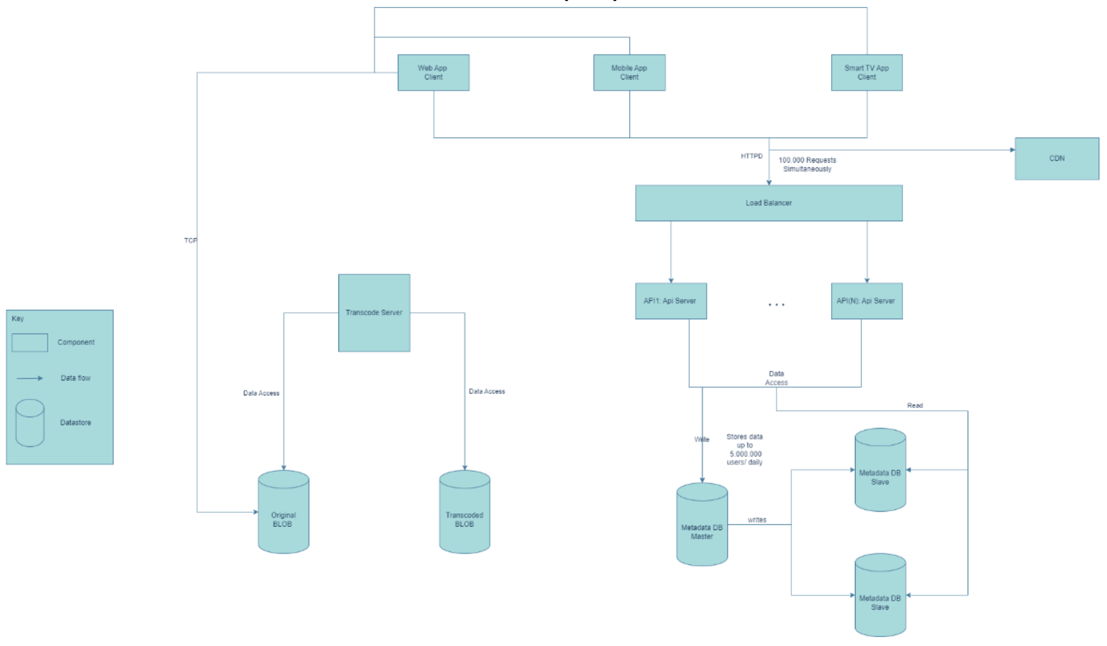
○ Iteration 7: Security against plagiarism(video watermarking) (S2)



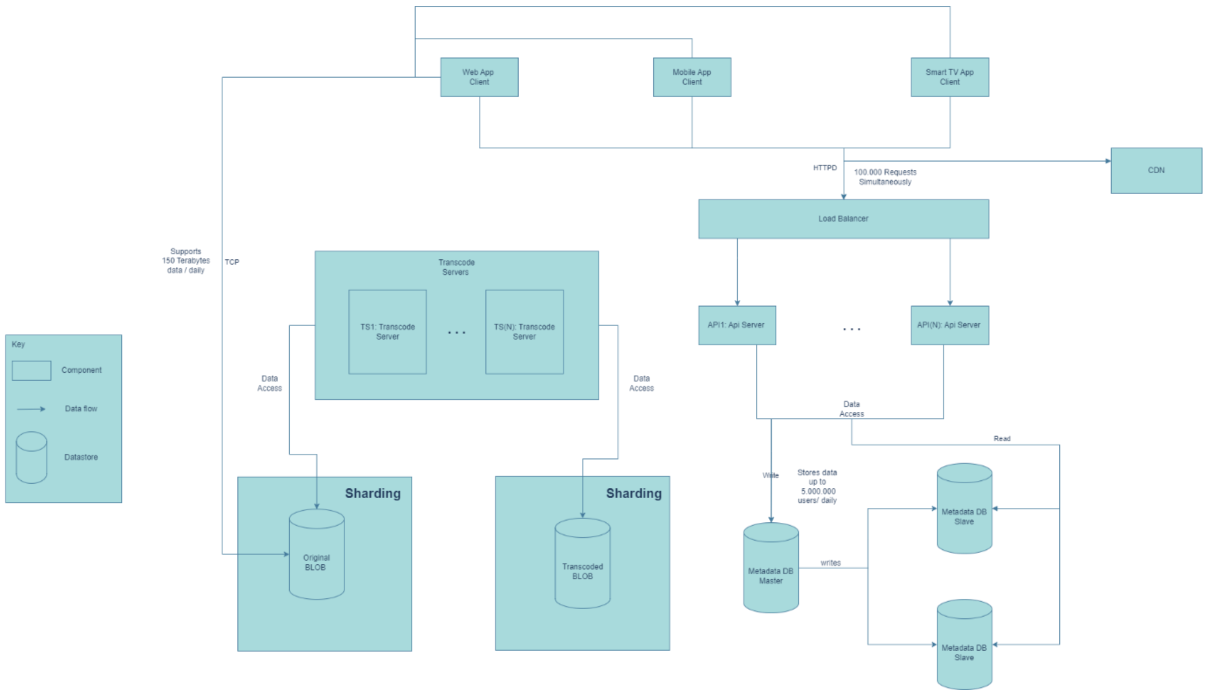
- 2º Round
  - Purpose of Design: Add Scalability (M3, M5, M6, M7, M8)
  - Iteration 1: Increase Requests (M7)



  - Iteration 2: Increase Users (M3)



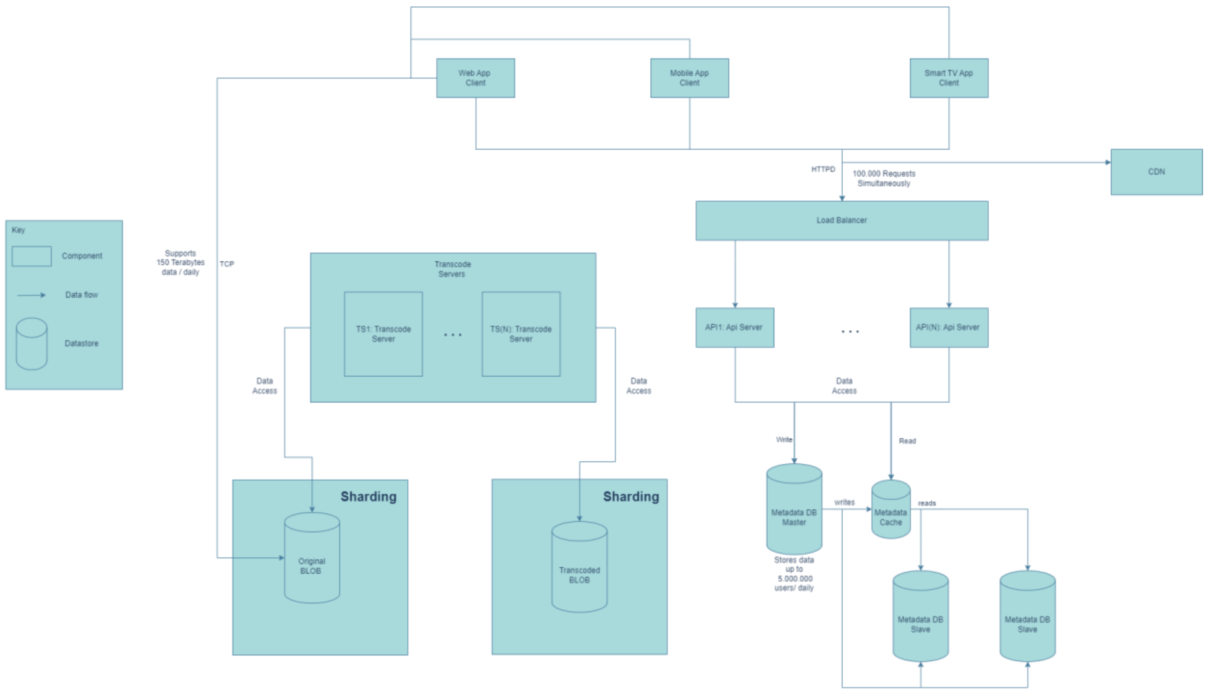
- Iteration 3: Increase Storage (M5, M8)



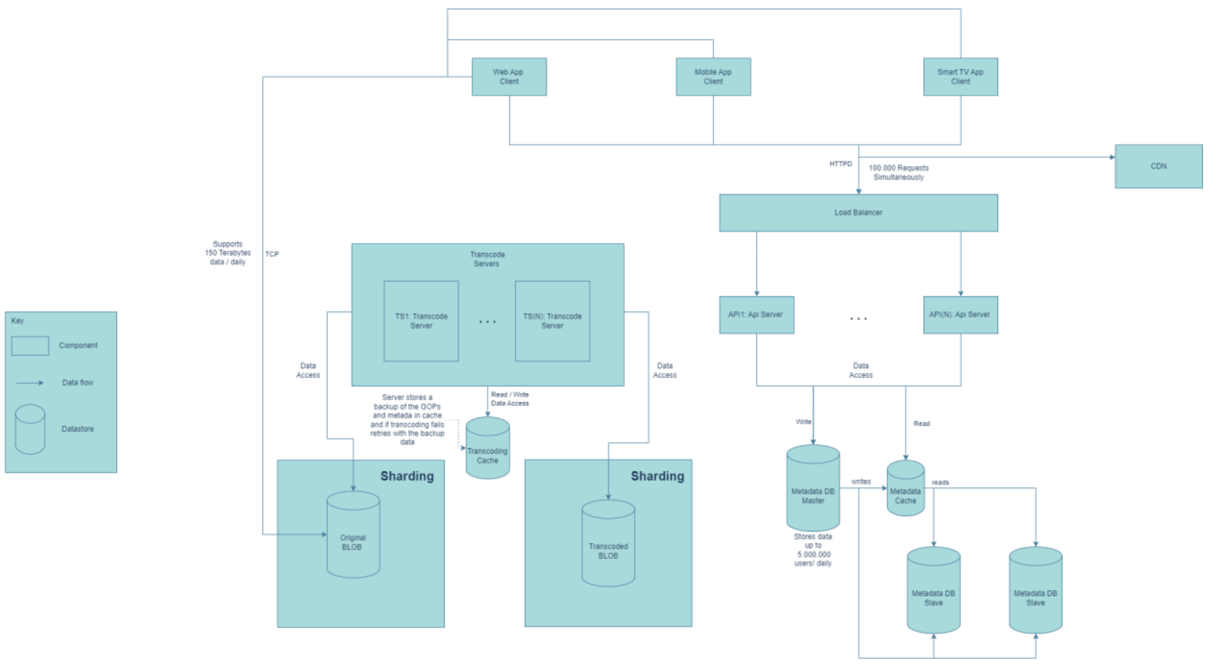
- 3º Round

  - Purpose of Design: Add availability.(M1, M2, A1, A2)

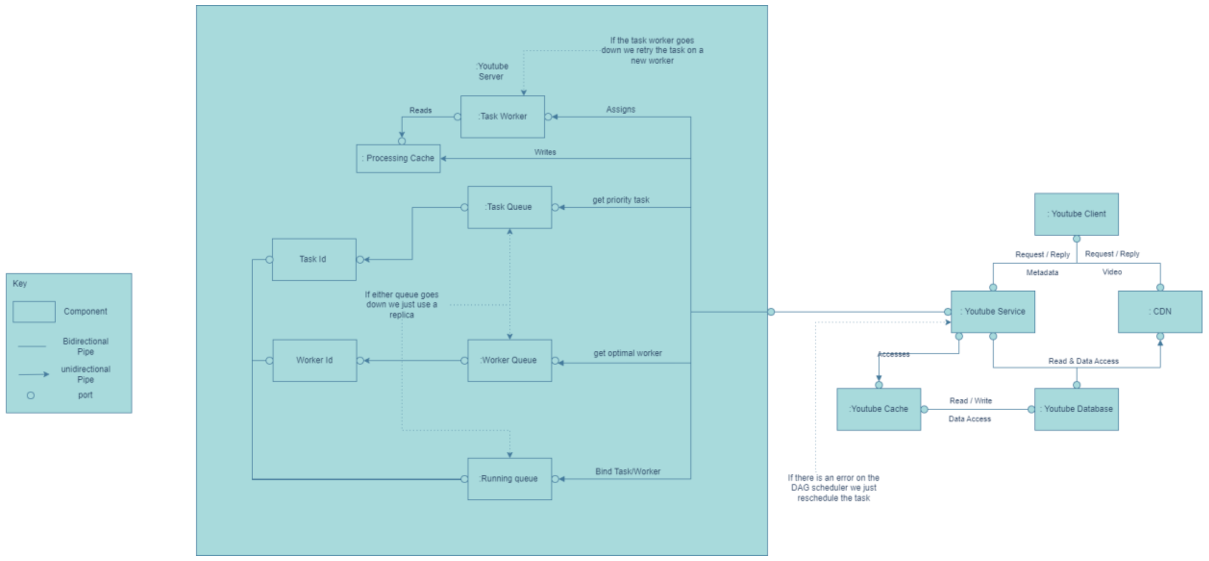
  - Iteration 1: Decrease inaccessible time. (A2)



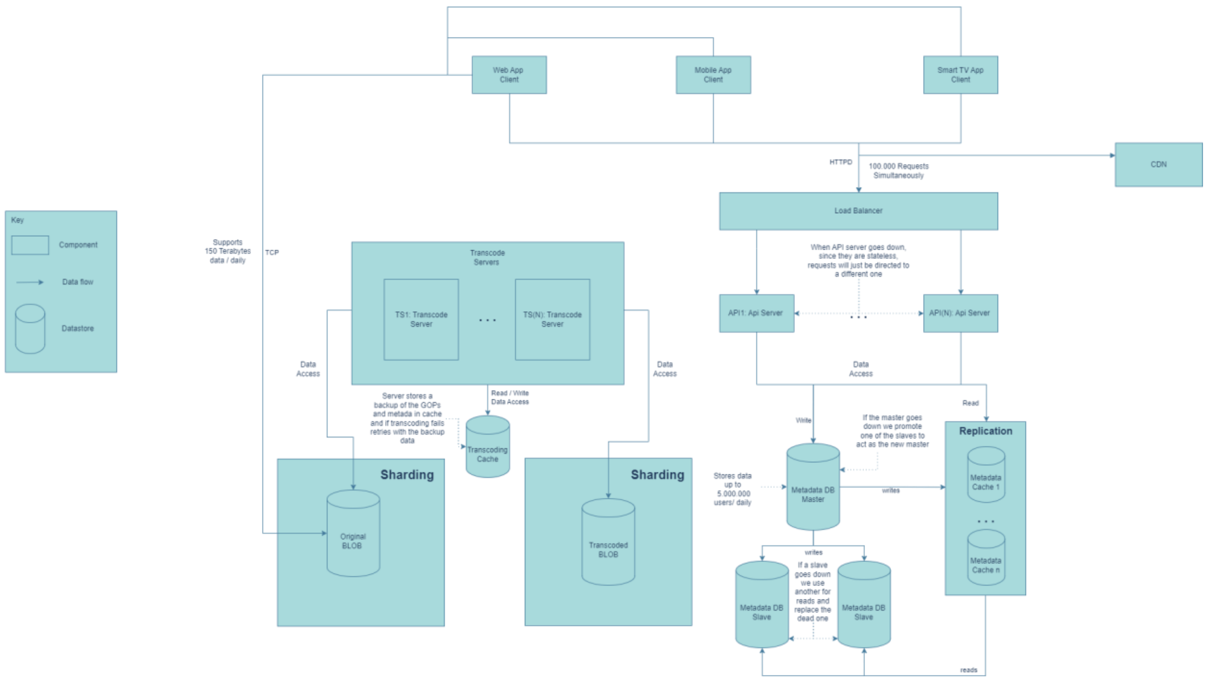
- Iteration 2: Lower transcoding failure (A1)



- Iteration 3: Recover progress from DAG scheduler, resource manager and task worker. (A3)

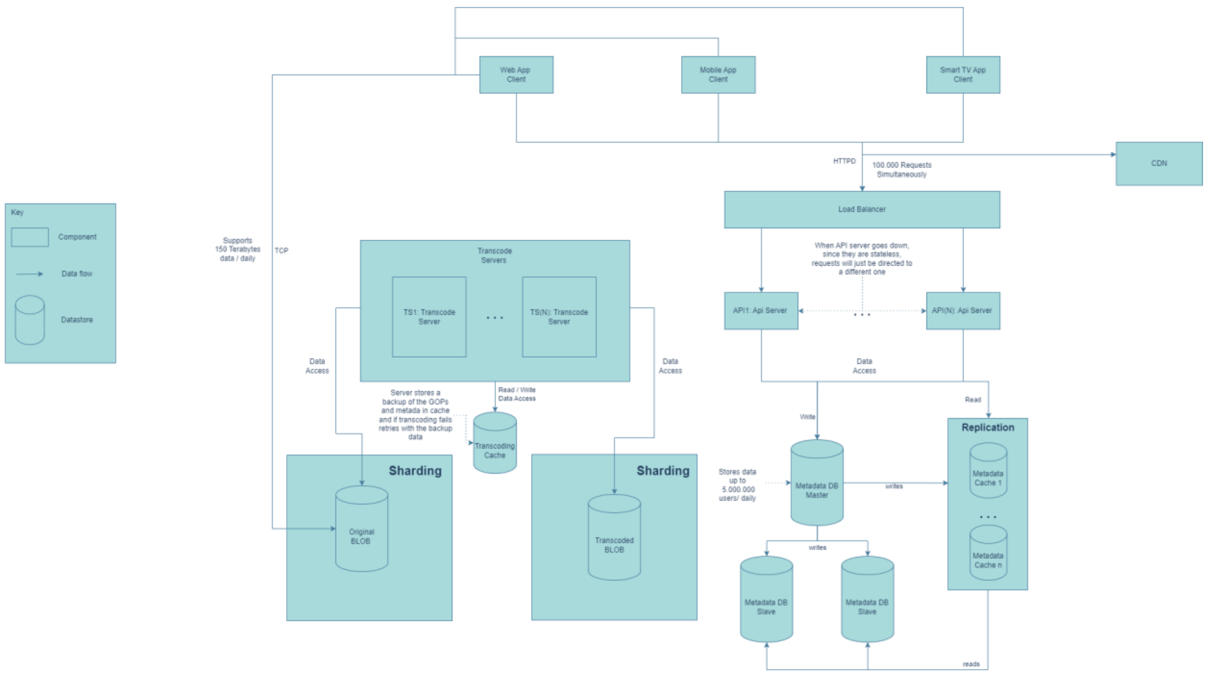


- Iteration 4: Maintain user accessibility to the API. (A1, A3)

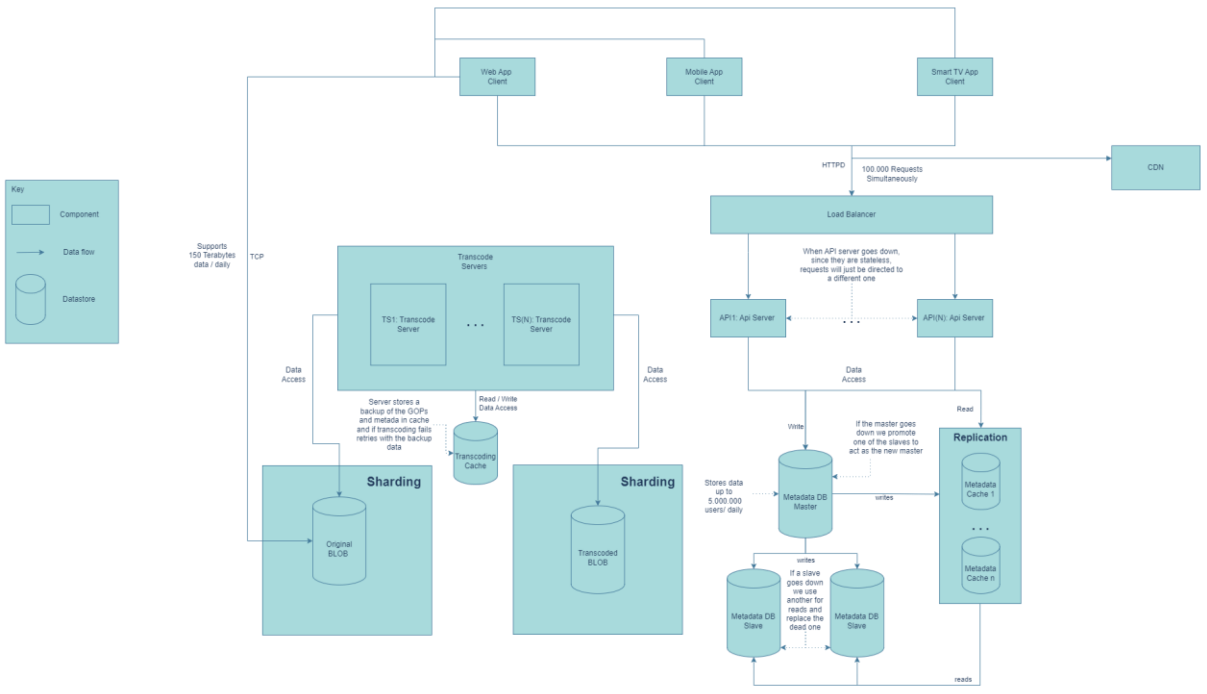


Key
Component
Data flow
Datastore
Web App Client
Mobile App Client
Smart TV App Client
HTTPD
100.000 Requests Simultaneously
CDN
Load Balancer
When API server goes down, since they are stateless, requests will just be directed to a different one
API1: Api Server
API(N): Api Server
Data Access
Read
Write
If the master goes down we promote one of the slaves to act as the new master
Replication
Metadata Cache 1
Metadata Cache n
Stores data up to 5.000.000 users/ daily
Metadata DB Master
writes
If a slave goes down we use another for reads and replace the dead one
Metadata DB Slave
Metadata DB Slave
reads
Supports 150 Terabytes data / daily
TCP
Transcode Servers
TS1: Transcode Server
TS(N): Transcode Server
Data Access
Data Access
Read / Write Data Access
Server stores a backup of the GOPs and metada in cache and if transcoding fails retries with the backup data
Transcoding Cache
Sharding
Original BLOB
Sharding
Transcoded BLOB

- Iteration 5: Add cache rerouting for videos and metadata.(A3)



- Iteration 6: Recover database functionality using warm redundant spare and maintain master-slave structure. (A3)


Key
Component
Data flow
Datastore
Web App Client
Mobile App Client
Smart TV App Client
HTTPD
100.000 Requests Simultaneously
CDN
Load Balancer
Supports 150 Terabytes data / daily
TCP
Transcode Servers
TS1: Transcode Server
. . .
TS(N): Transcode Server
When API server goes down, since they are stateless, requests will just be directed to a different one
API1: Api Server
. . .
API(N): Api Server
Data Access
Data Access
Data Access
Server stores a backup of the GOPs and metada in cache and if transcoding fails retries with the backup data
Read / Write Data Access
Transcoding Cache
Sharding
Original BLOB
Sharding
Transcoded BLOB
Write
Read
If the master goes down we promote one of the slaves to act as the new master
Replication
Stores data up to 5.000.000 users/ daily
Metadata DB Master
writes
Metadata Cache 1
. . .
Metadata Cache n
writes
If a slave goes down we use another for reads and replace the dead one
Metadata DB Slave
Metadata DB Slave
reads