

1. Functional Requirements:

Functional Requirements:

- 1) Video Onboarding/Uploading

Functional Requirements:

- 1) Video Onboarding/Uploading
- 2) Search Video

Functional Requirements:

- 1) Video Onboarding/Uploading
- 2) Search Video
- 3) View Videos

Functional Requirements:

- 1) Video Onboarding/Uploading
- 2) Search Video
- 3) View Videos
- 4) Generate Recommendations

Functional Requirements:

- 1) Video Onboarding/Uploading
- 2) Search Video
- 3) View Videos
- 4) Generate Recommendations

Non-Functional Requirements:

|

Functional Requirements:

- 1) Video Onboarding/Uploading
- 2) Search Video
- 3) View Videos
- 4) Generate Recommendations

Non-Functional Requirements:

- 1) Low Latency|

Functional Requirements:

- 1) Video Onboarding/Uploading
- 2) Search Video
- 3) View Videos
- 4) Generate Recommendations

Non-Functional Requirements:

- 1) Low Latency
- 2) High Availability

|

Design Video Streaming Platform:

Functional Requirements:

- 1) Video Onboarding/Uploading
- 2) Search Video
- 3) View Videos
- 4) Generate Recommendations

Non-Functional Requirements:

- 1) Low Latency
- 2) High Availability

Capacity Estimations:

10^3 = 1KB = Thousand

10^6 = 1MB = Million

10^9 = 1GB = Billion

10^{12} = 1TB = Trillion

10^{15} = 1PB = Quadrillion

Day to Seconds:

24 hours * 60 mins *60 seconds

=24 * 3600

=25 * 4000

100,000

= 10^5 |

Functional Requirements:

- 1) Video Onboarding/Uploading
- 2) Search Video
- 3) View Videos
- 4) Generate Recommendations

Non-Functional Requirements:

- 1) Low Latency
- 2) High Availability

Capacity Estimation:

1) DAU: 100M Users * 2 times/day

$$\begin{aligned} \text{QPS} &= 100M * 2 \text{ times} / (24 \text{ hours} * 60 \text{ mins} * 60 \\ &\quad \text{seconds}) = 100 * 10^6 * 2 / 10^6 \\ &= 2000 \text{ QPS} \end{aligned}$$

Functional Requirements:

- 1) Video Onboarding/Uploading
- 2) Search Video
- 3) View Videos
- 4) Generate Recommendations

Non-Functional Requirements:

- 1) Low Latency
- 2) High Availability

Capacity Estimation:

1) DAU: 100M Users * 2 times/day

$$\text{QPS} = 100M * 2 \text{ times} / (24 \text{ hours} * 60 \text{ mins} * 60 \text{ seconds})$$

$$= 100 * 10^6 * 2 / 10^5$$

$$= 2000 \text{ QPS}$$

Storage:

10:1 viewers : Creators

100M viewers -> 10M Creators

Two times a week

Each video 5GB

Storage capacity =

10 M Creators * 2 times / week * 5GB video

$$= 10 * 10^6 * 2 \text{ times a week} * 4 \text{ weeks} * 12 \text{ months} * 5\text{GB}$$

$$= 10 * 10^6 * 100 * 5 * 10^9$$

$$= 10^{15} * 5000$$

$$= 5000 \text{ PB/year}$$

Database Design:

|

Database Design:

Videos -> BLOB storage

Database Design:
Videos -> BLOB storage

Video Metadata DB: SQL

Video ID(PK)
UserID (FK)
DTTM : DateTime
Visibility: Public/Private
Short Description: String
Long Description: String
Category : String

VideoURL Table:

Video ID (FK)
PartID: Numeric
Format: String(Mp3,MP4 etc)
----- Unique Key

VideoURL Table:

Video ID (FK)

PartID: Numeric

Format: String(Mp3,MP4 etc)

----- Unique Key

User Table :

UserID(PK)

UserName: String

FirstName : String

Last Name : String

CreatedDTTM: DTTM

LastLogin : DTTM

Video ID (FK)
PartID: Numeric
Format: String(Mp3,MP4 etc)
----- Unique Key

User Table :

UserID(PK)
UserName: String
FirstName : String
Last Name : String
CreatedDTTM: DTTM
LastLogin : DTTM

History Table:

UserID (FK)
Video ID (FK)
DTTM: Date Time
WatchTime : Float

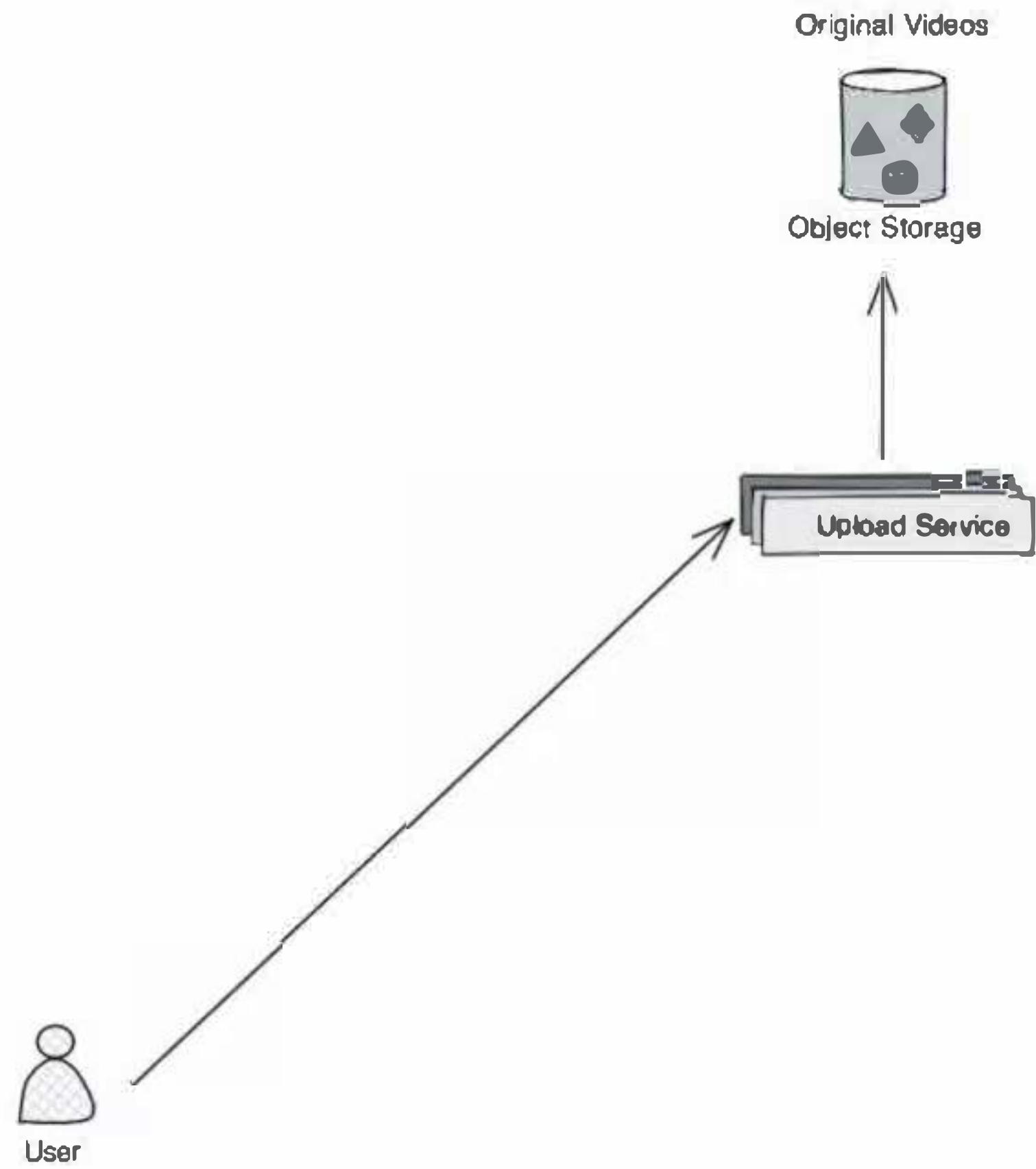
$$\begin{aligned} &= 10^{15} * 5000 \\ &= 5000 \text{ PB/year} \end{aligned}$$

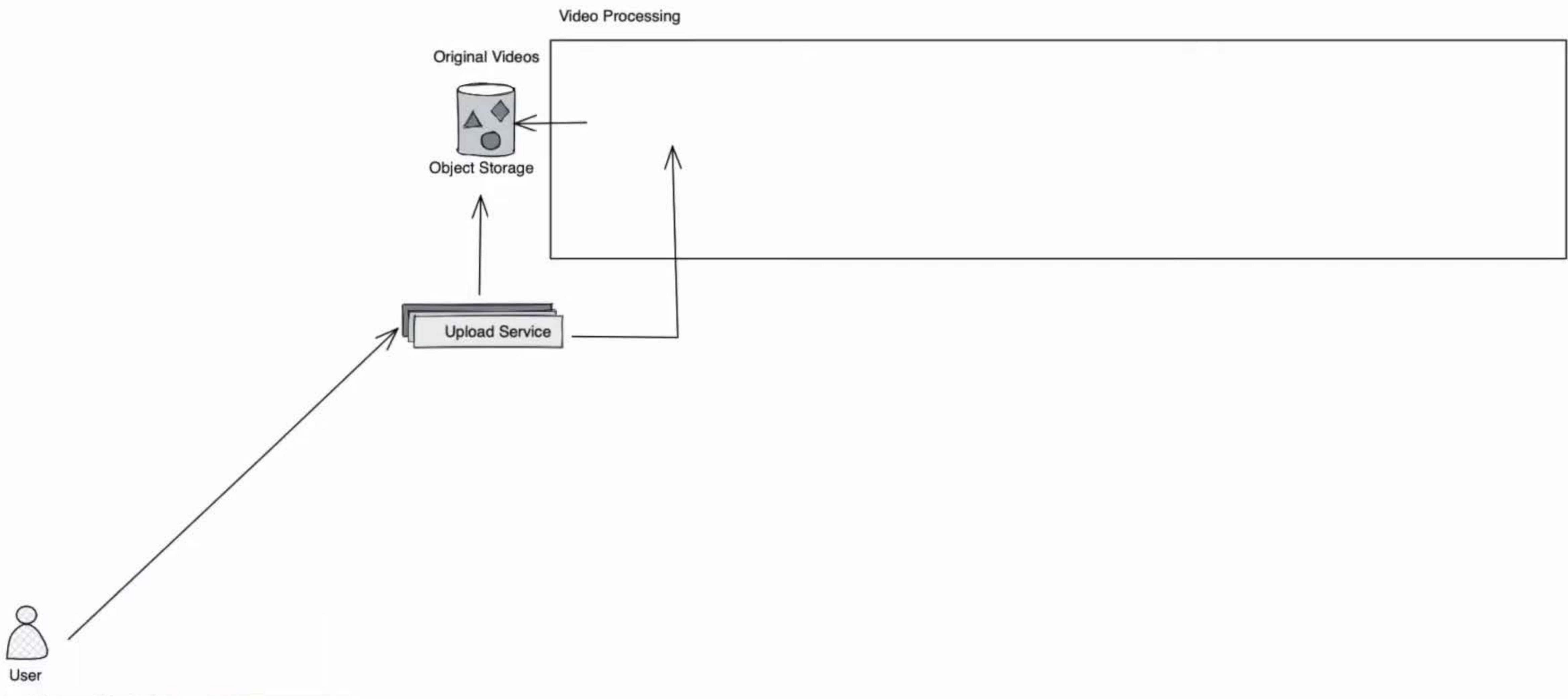
Bandwidth Server side:

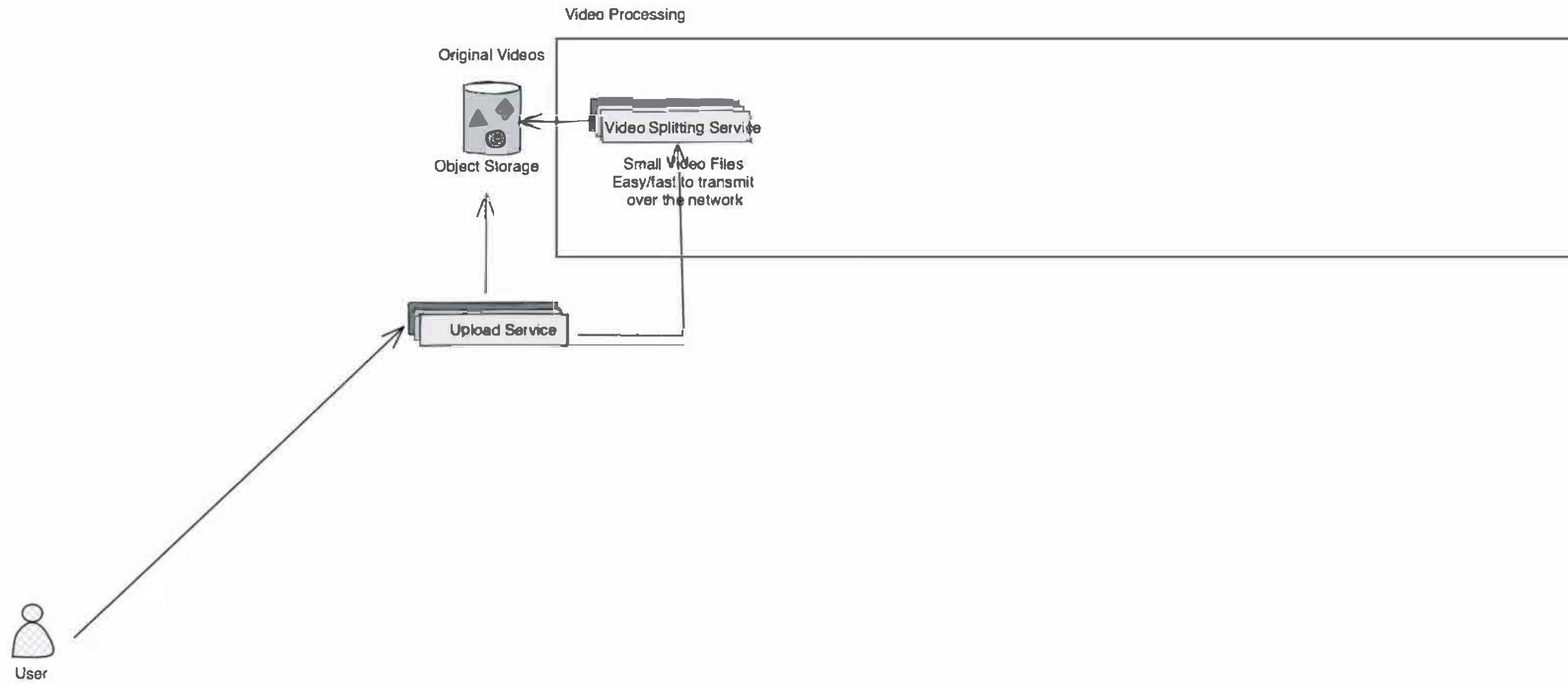
$$\begin{aligned} &2000 \text{ QPS} * 5\text{GB} \\ &= 2 * 10^3 * 5 * 10^9 \\ &\approx 10 * 10^{12} \\ &= 10\text{TB/Second} \end{aligned}$$

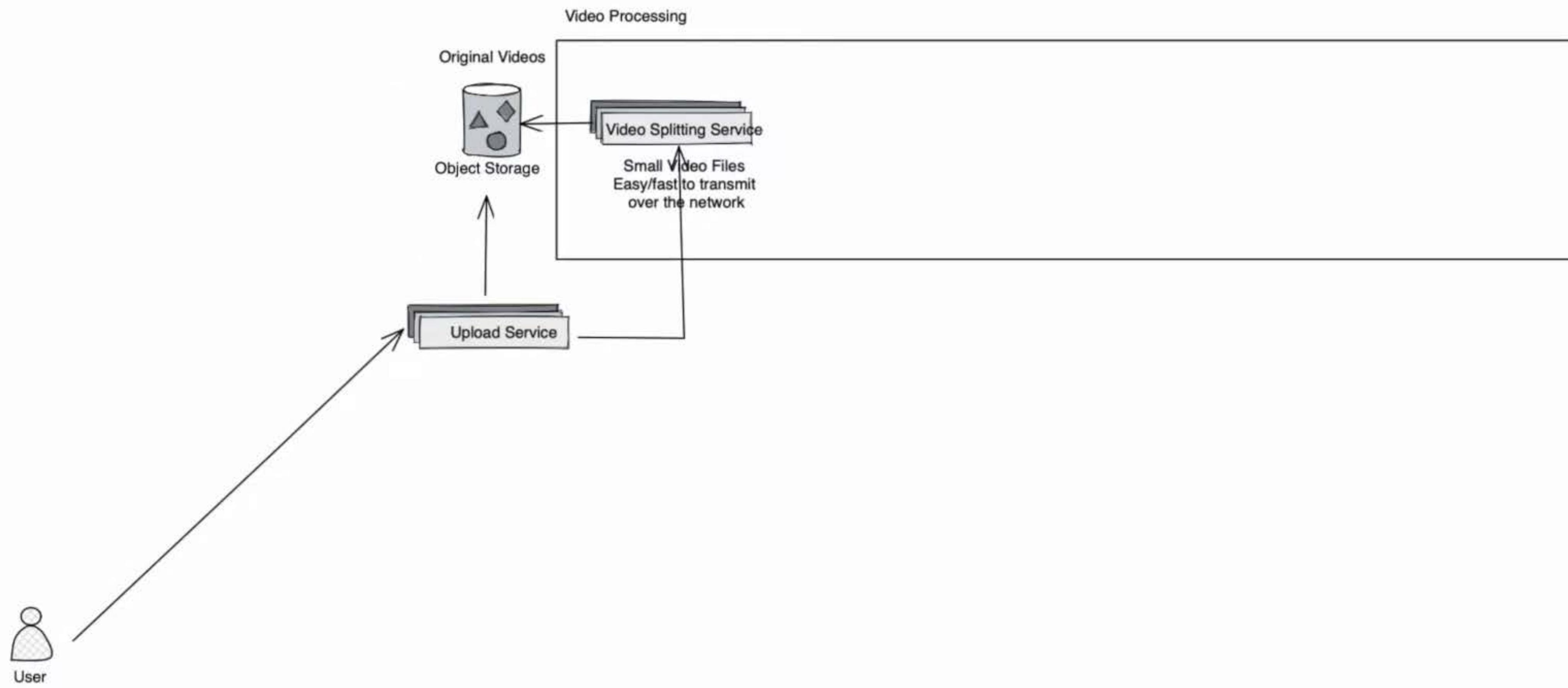
API:

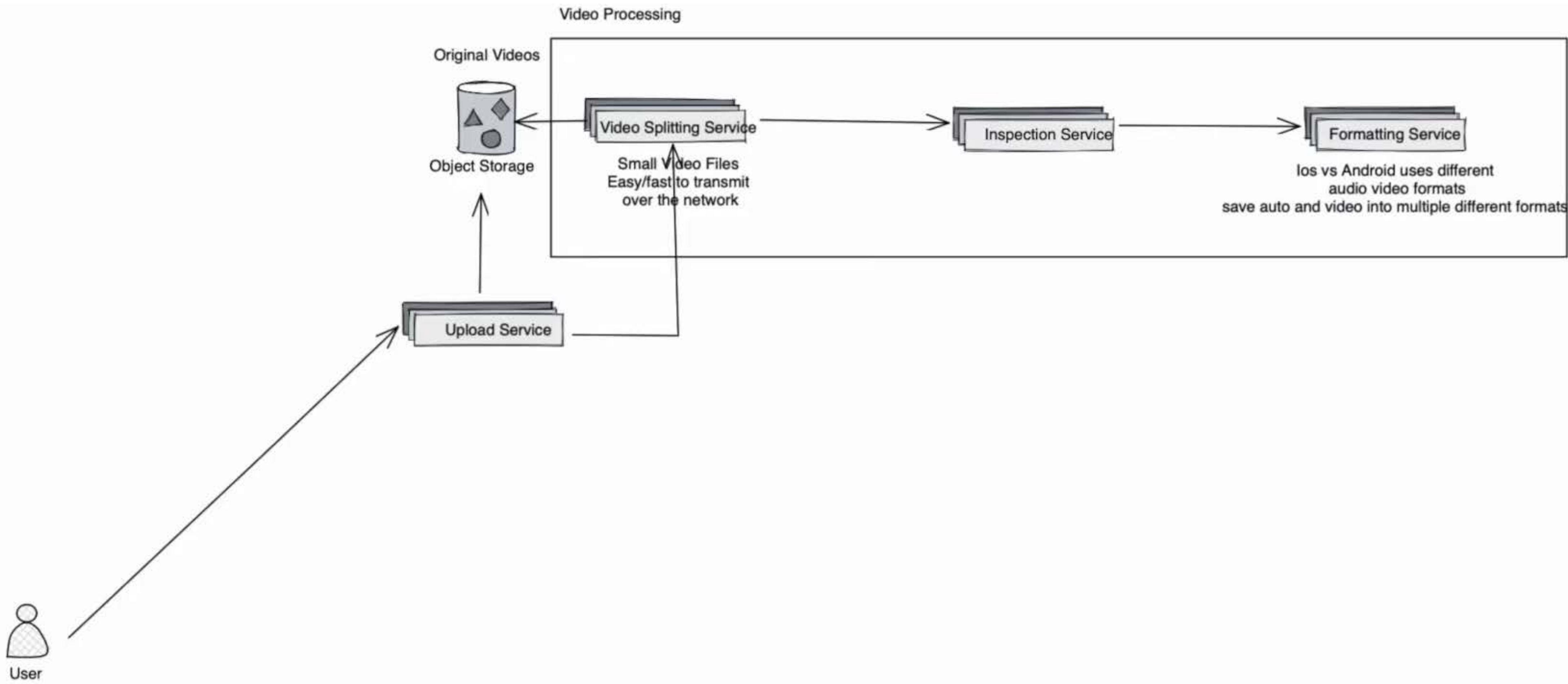
- 1) View Service(VideoID) - GET method - return video URL parts
- 2) Upload Service(Video File) - POST Method - return a message of successful upload or an error
- 3) Recommendation service(UserID) - GET - return 20 recommended videos
- 4) Update History Service (UserID, VideoID, Start DTTM, End DTTM)
POST Method

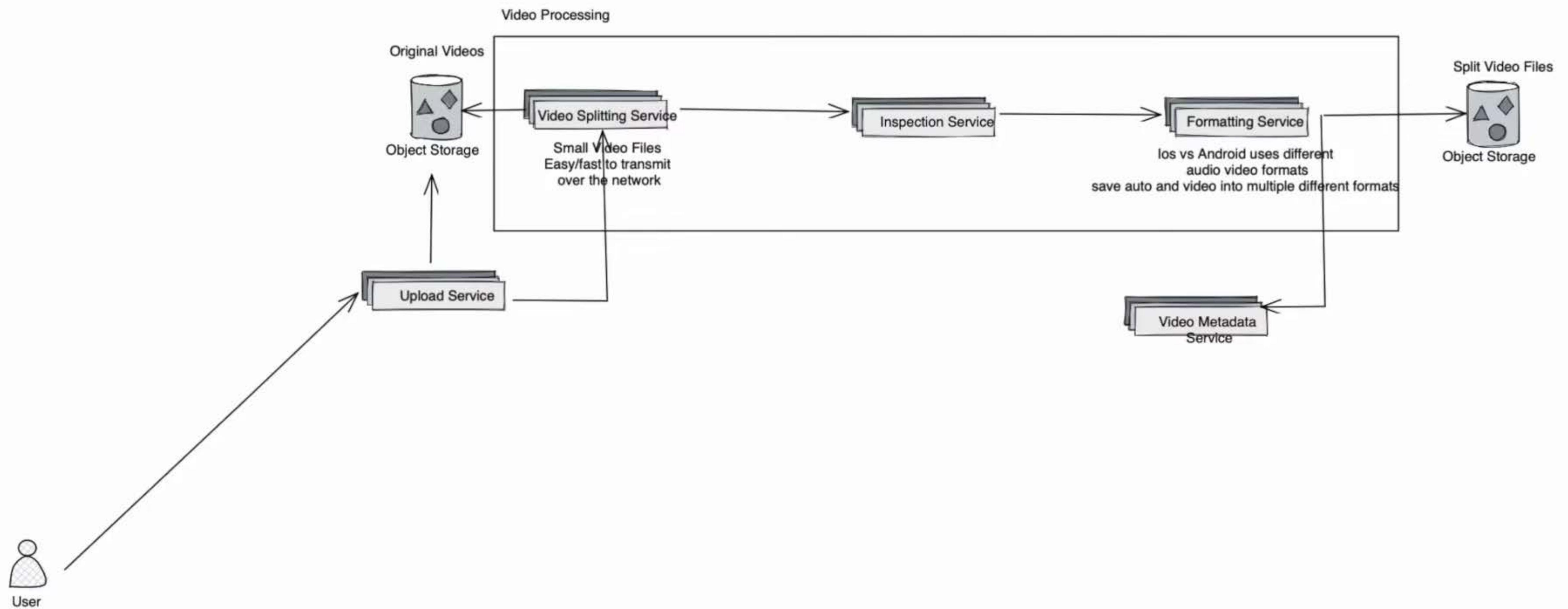


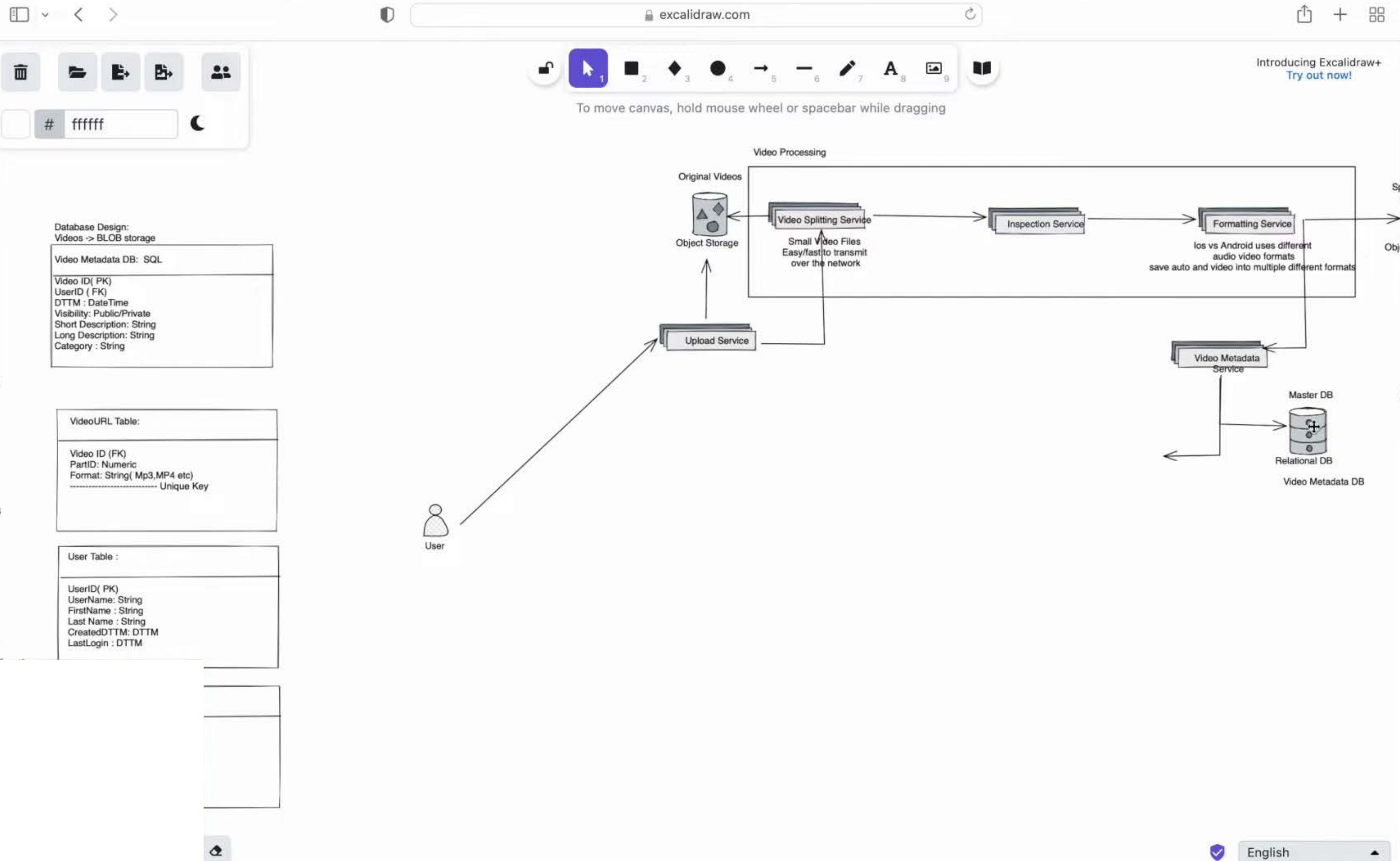


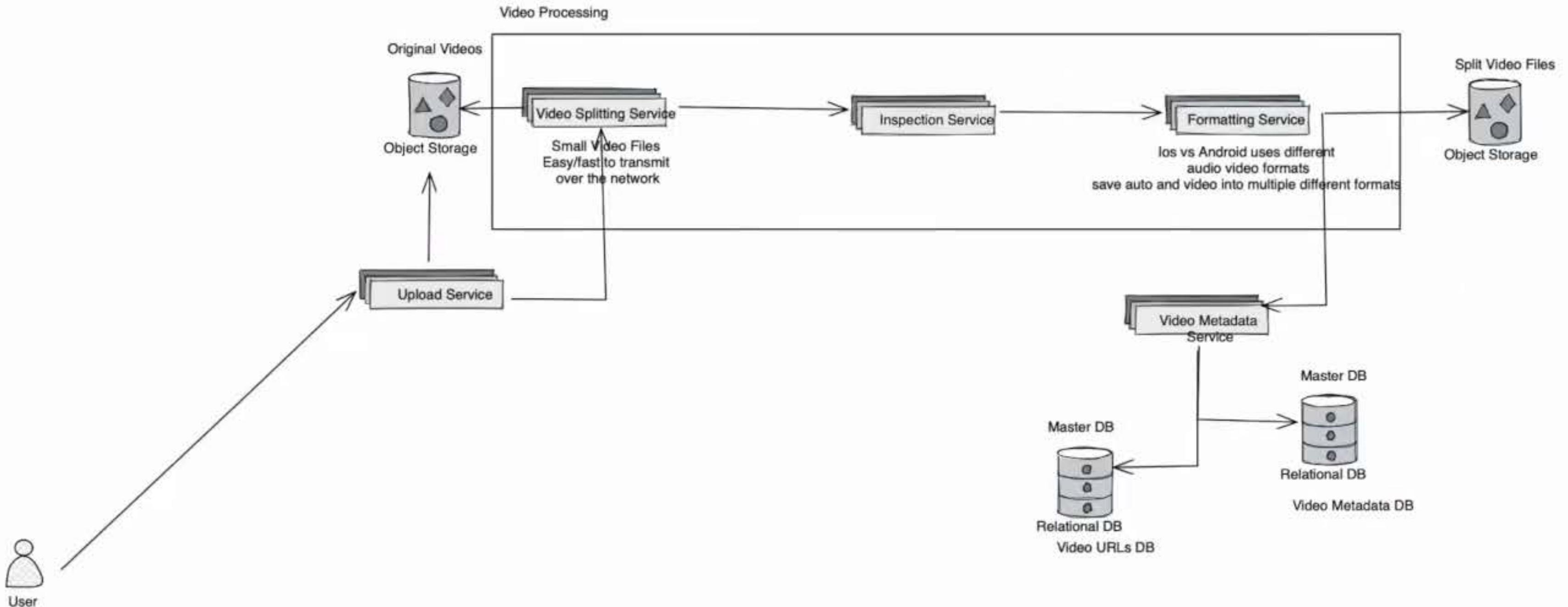




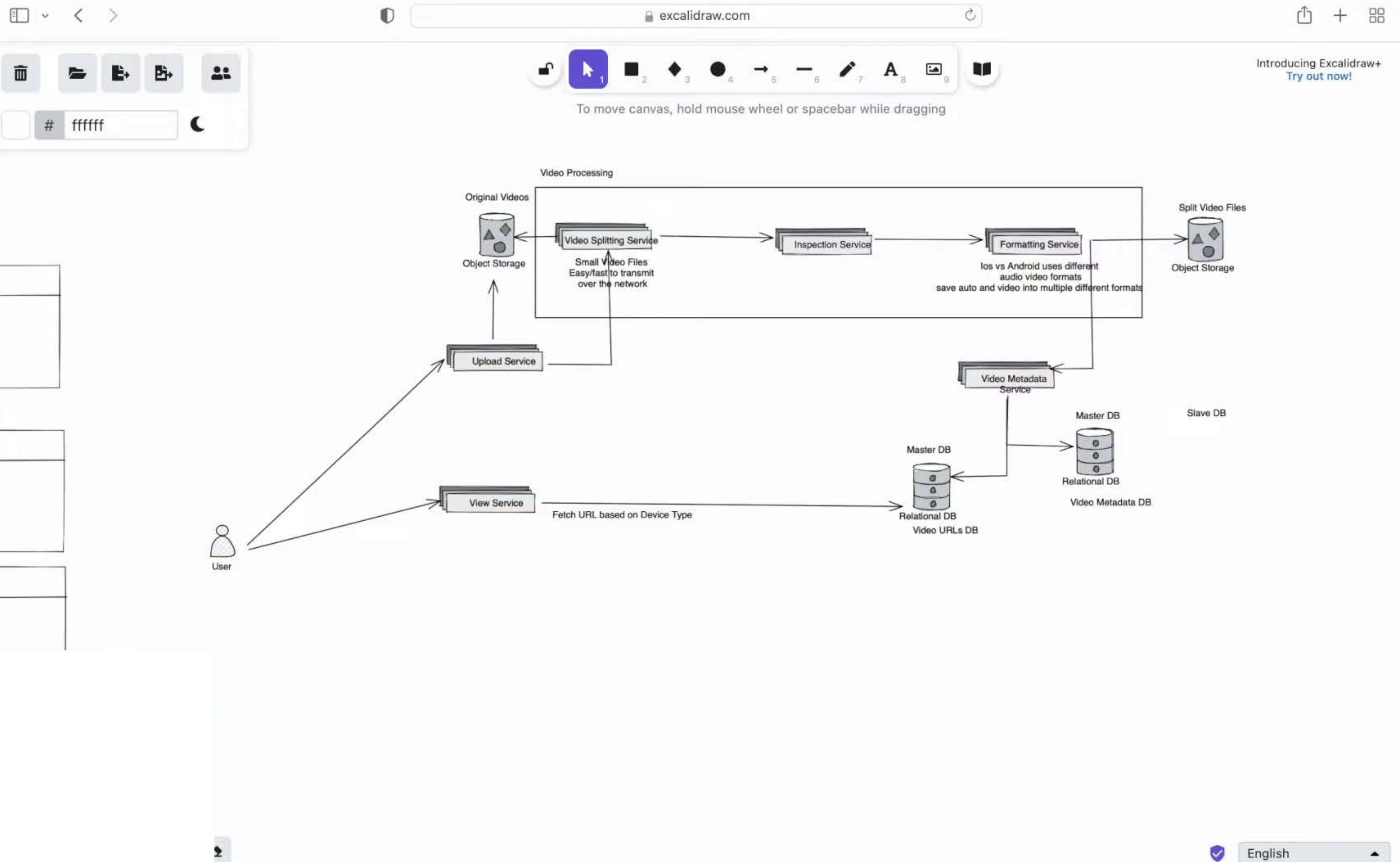


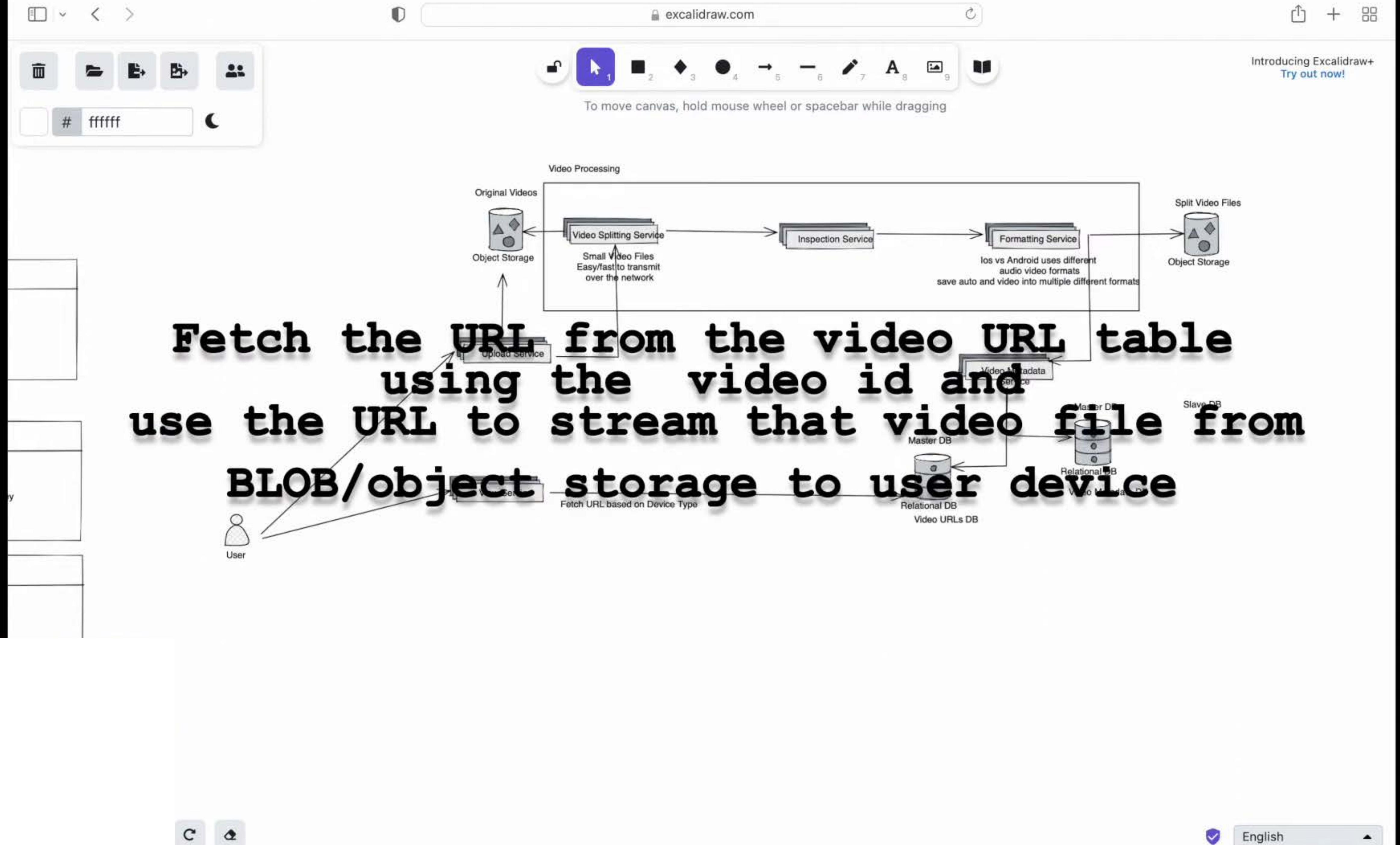


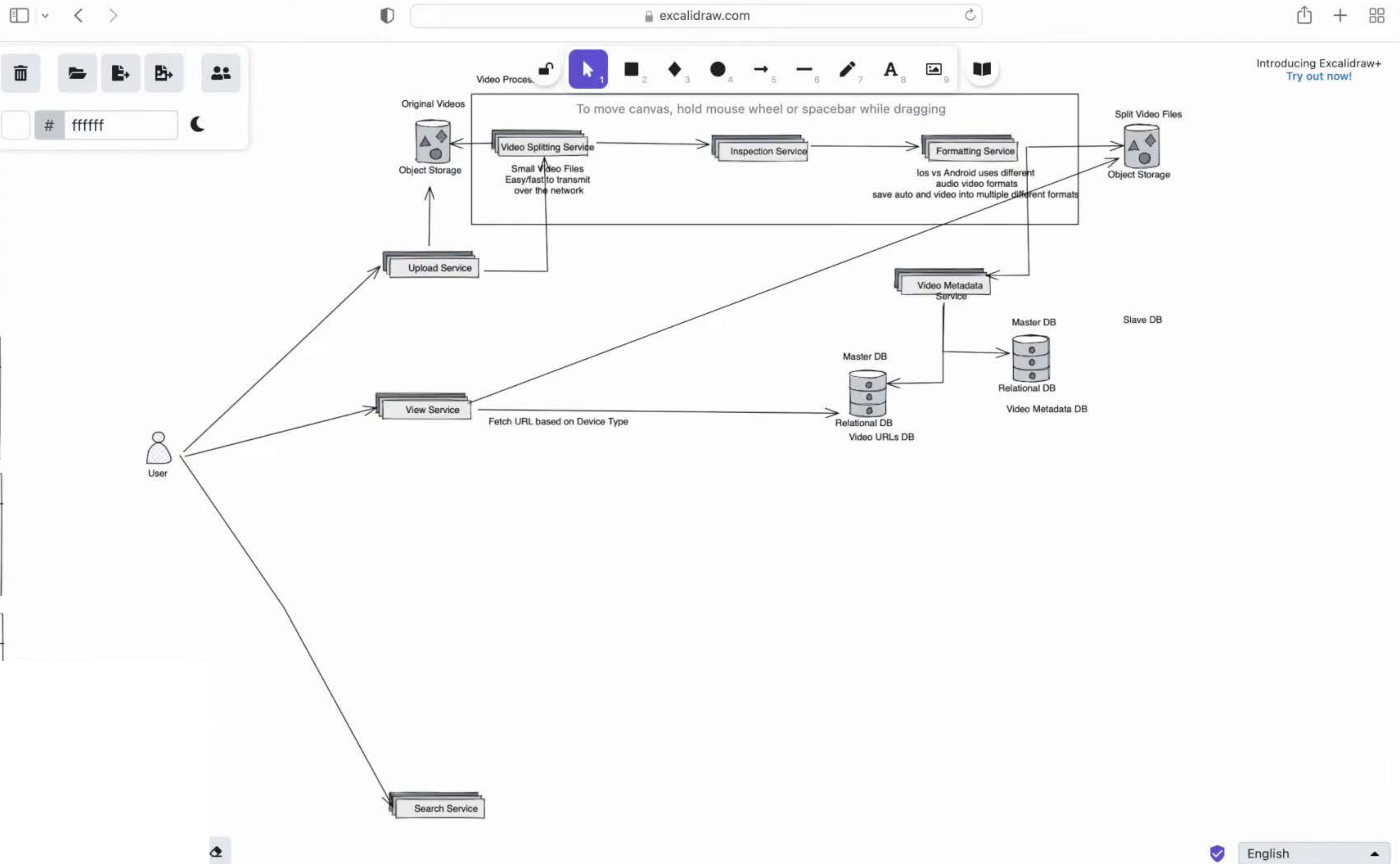


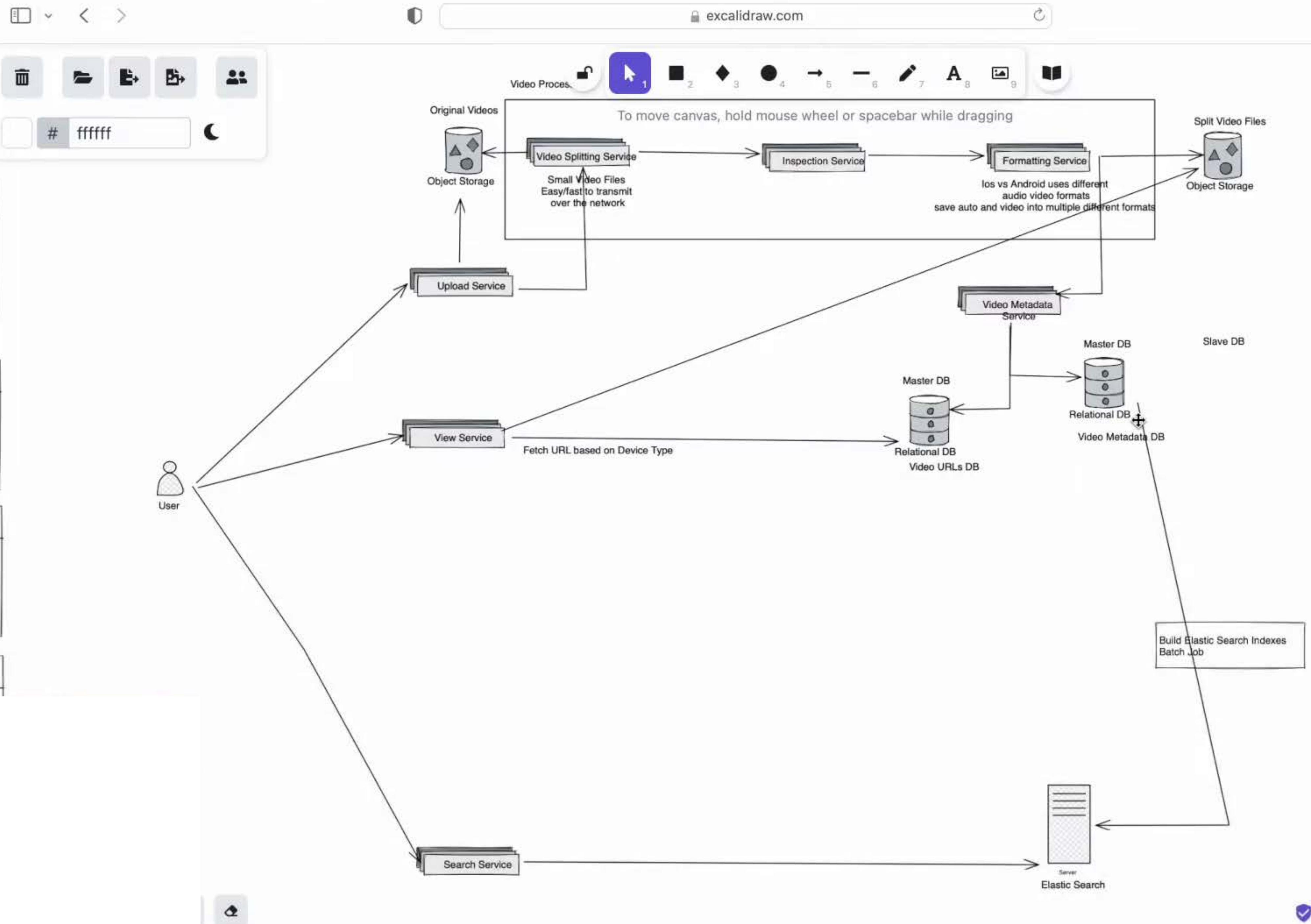


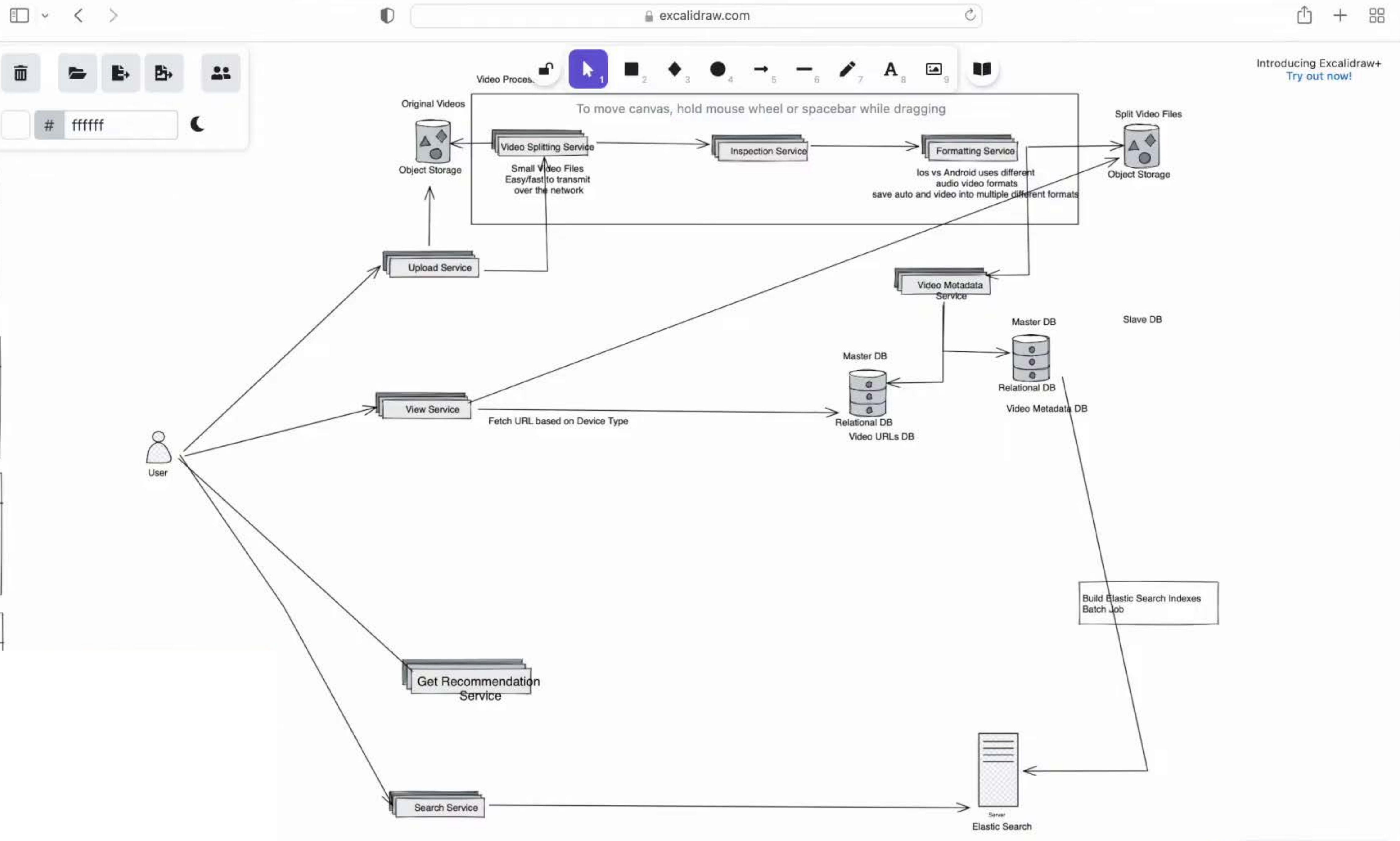
English

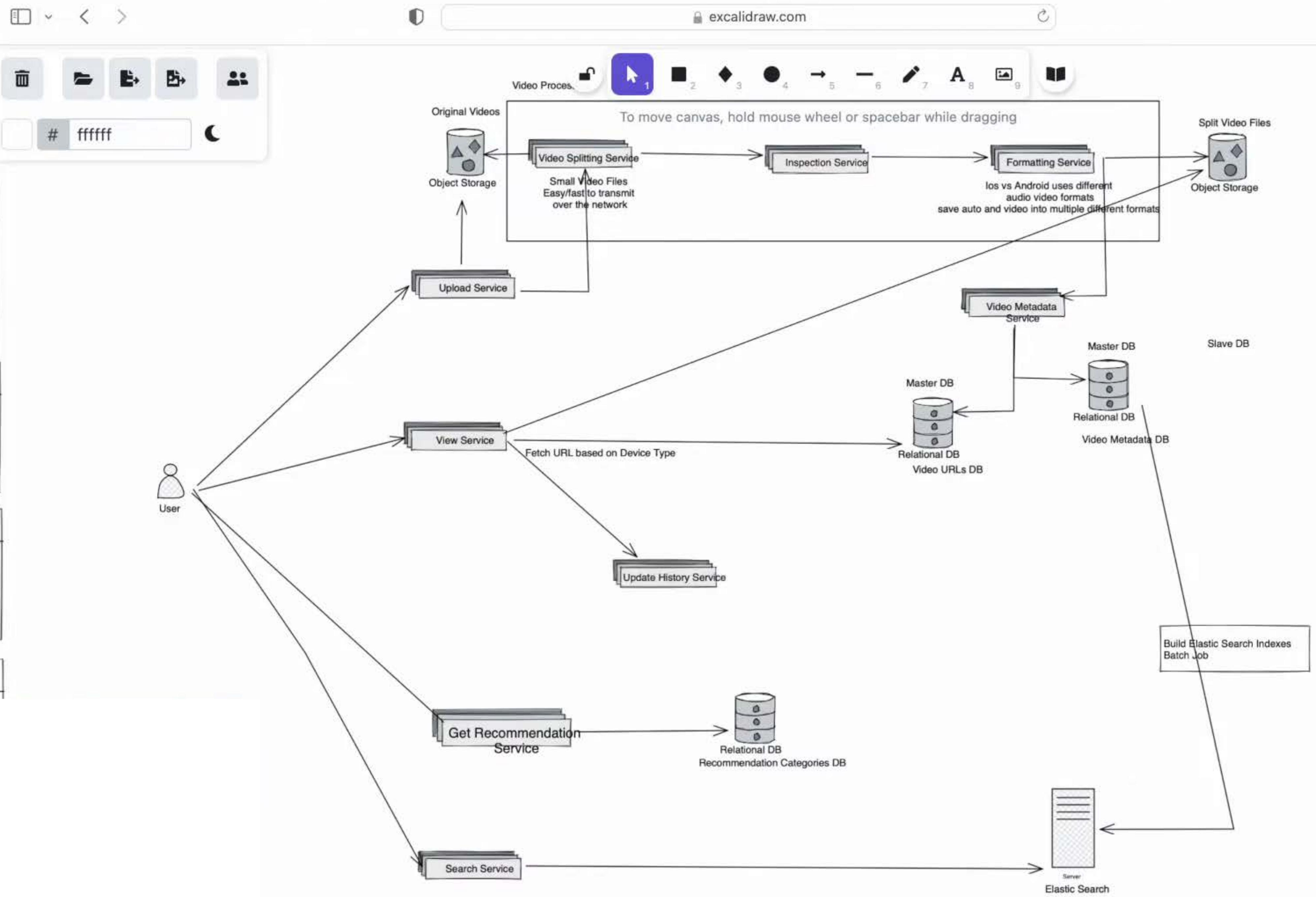


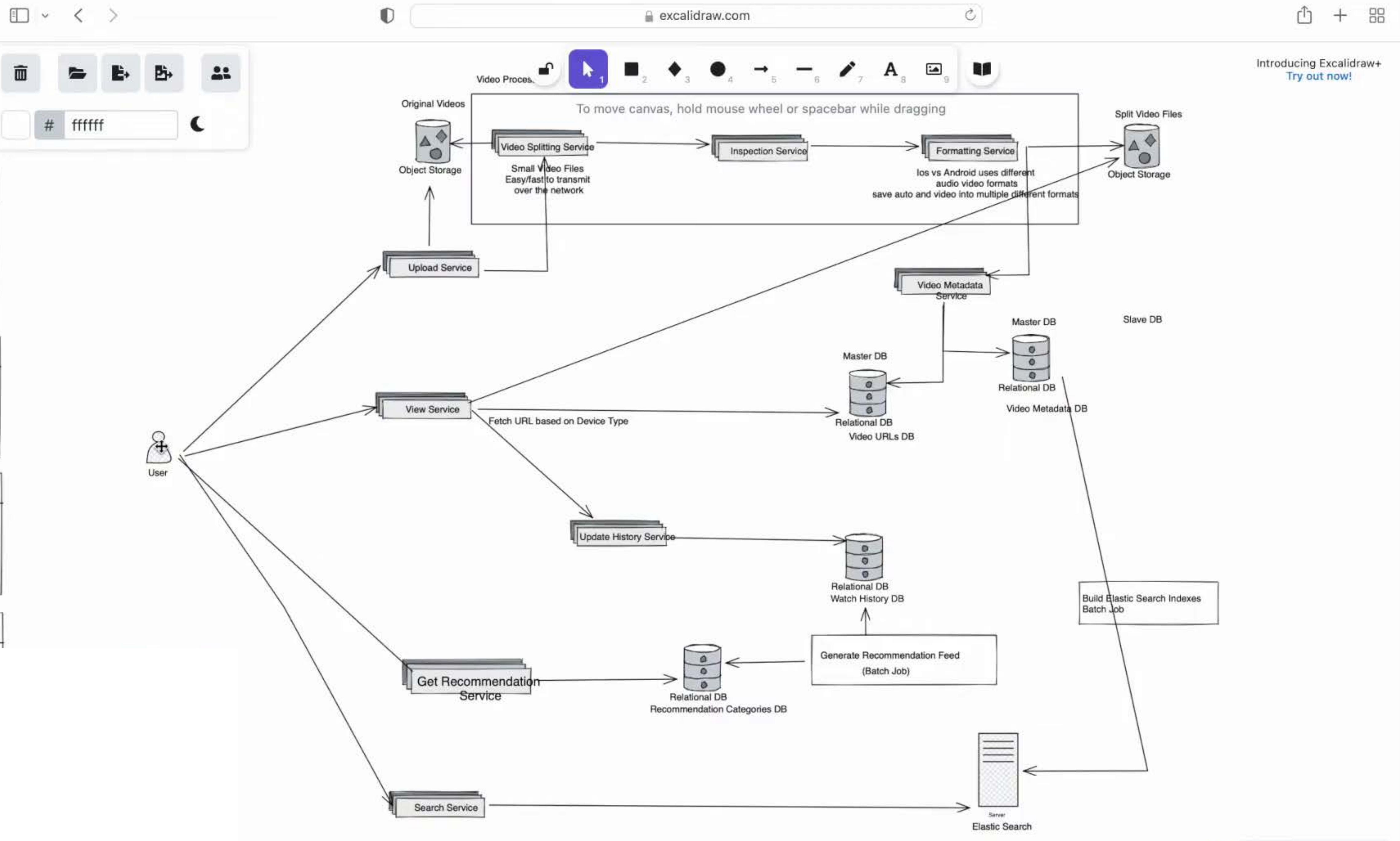






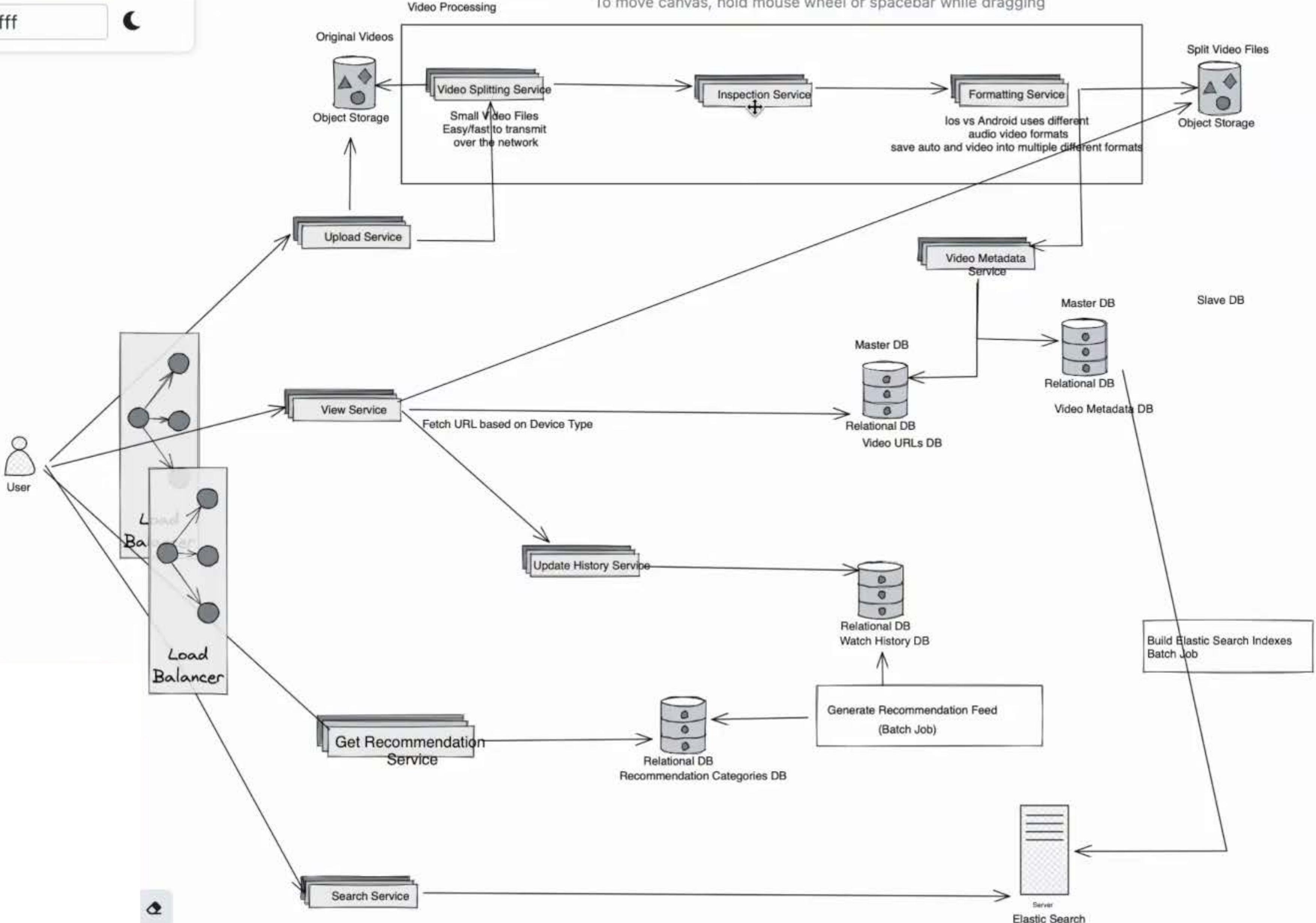


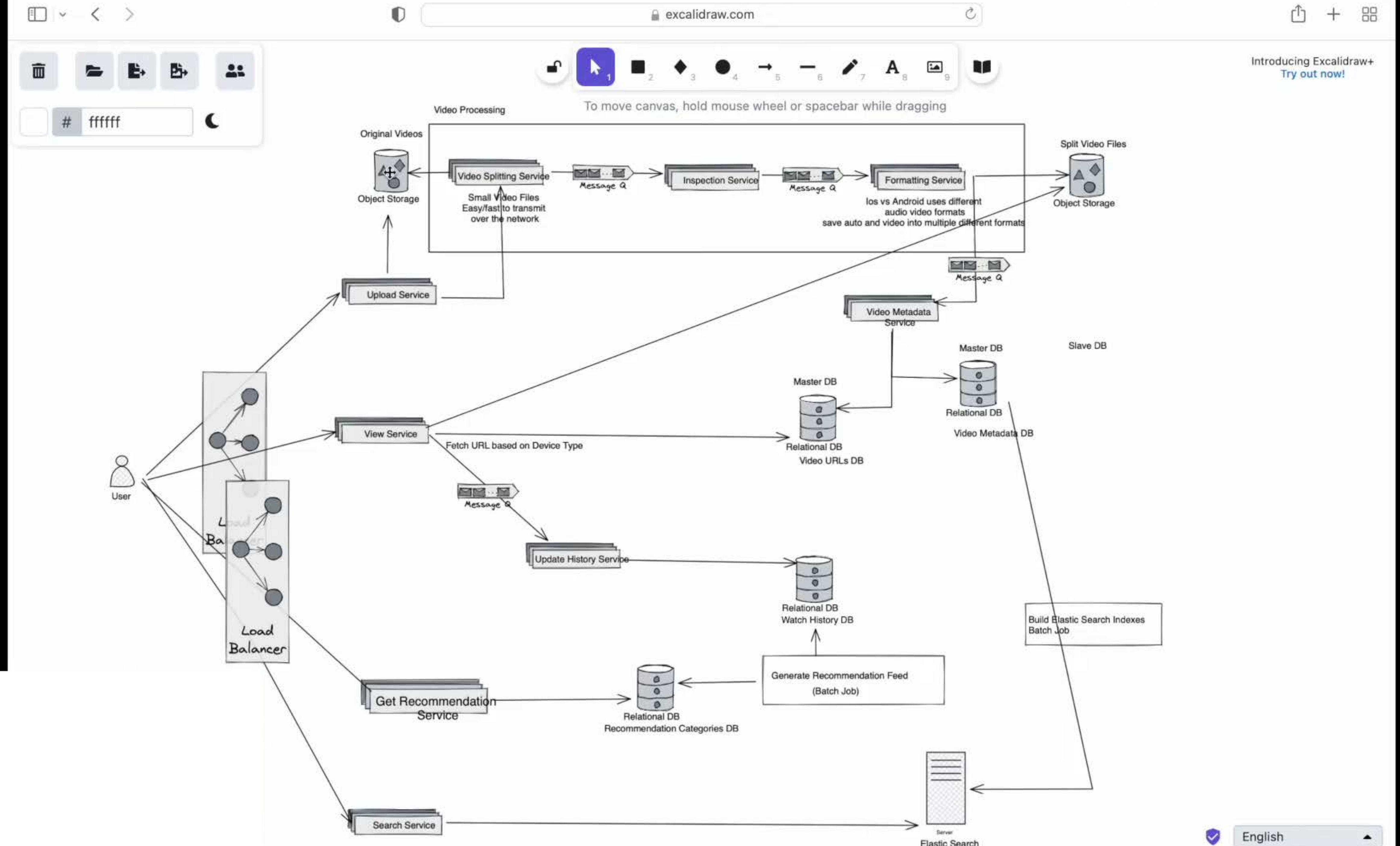




ffffff

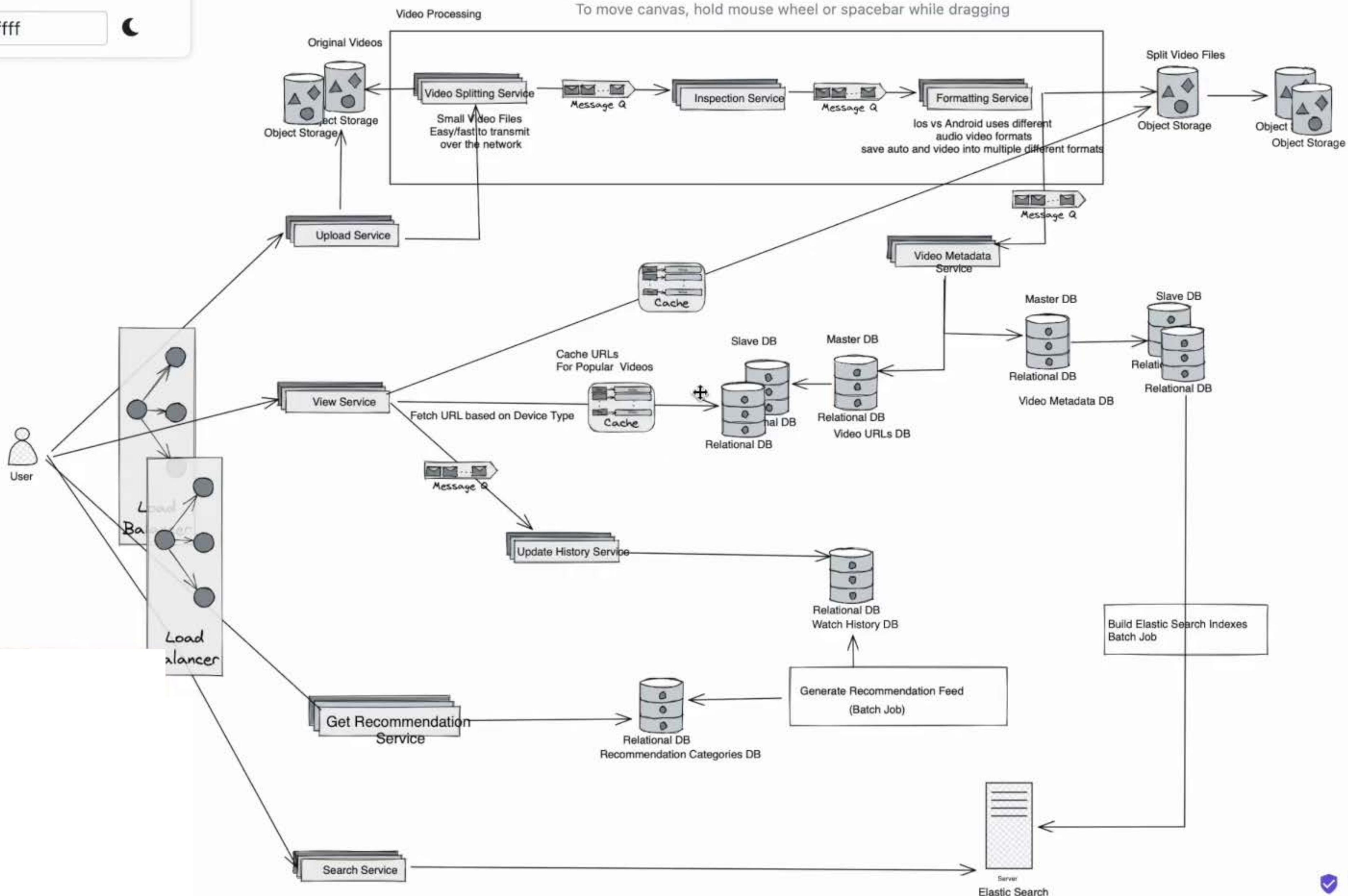
Introducing Excalidraw+
Try out now!

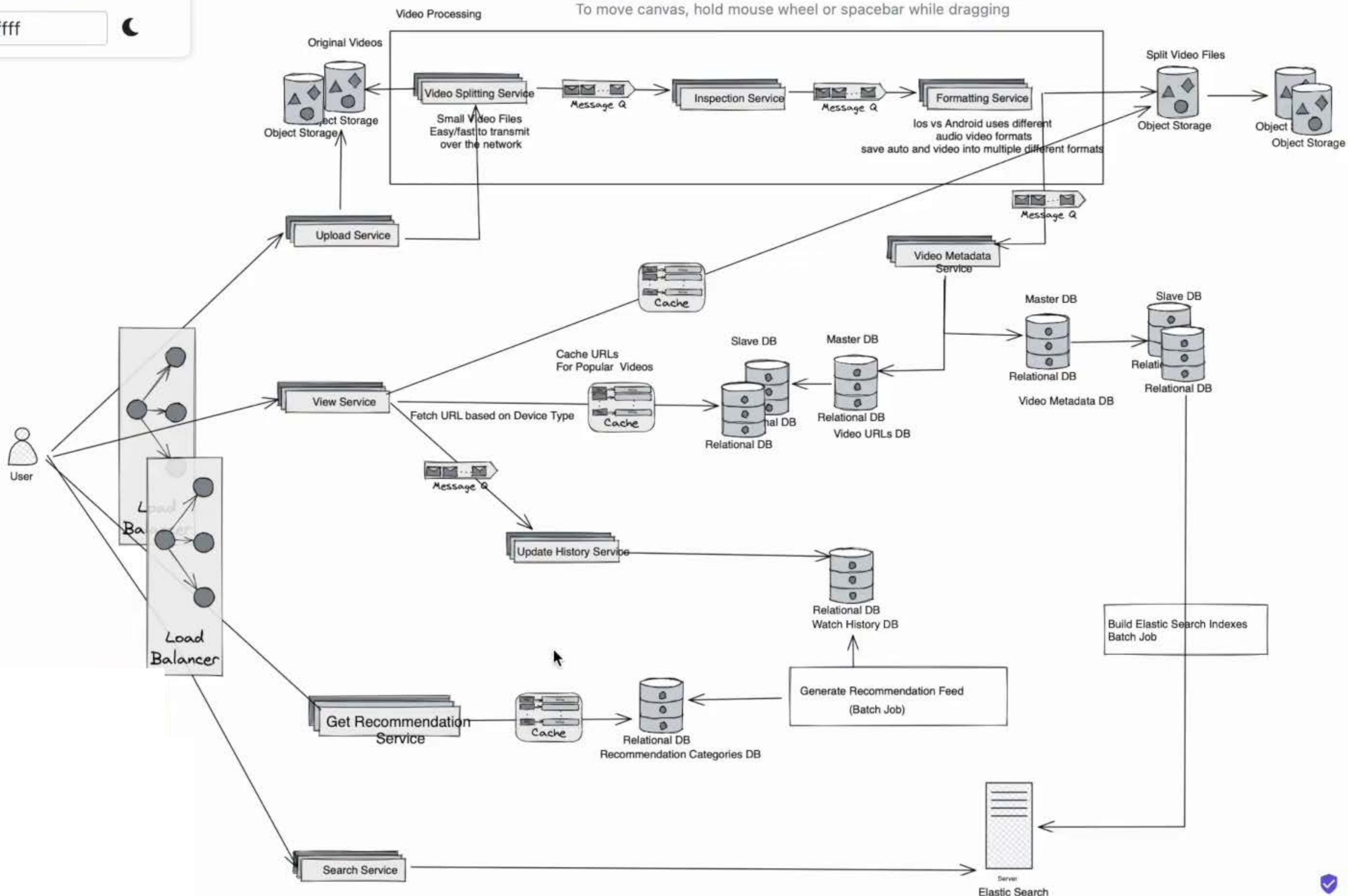
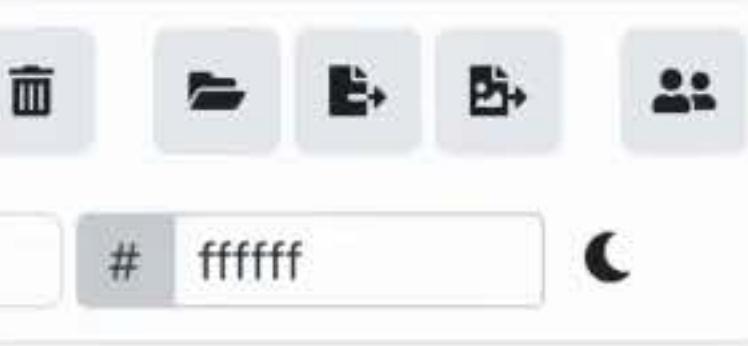


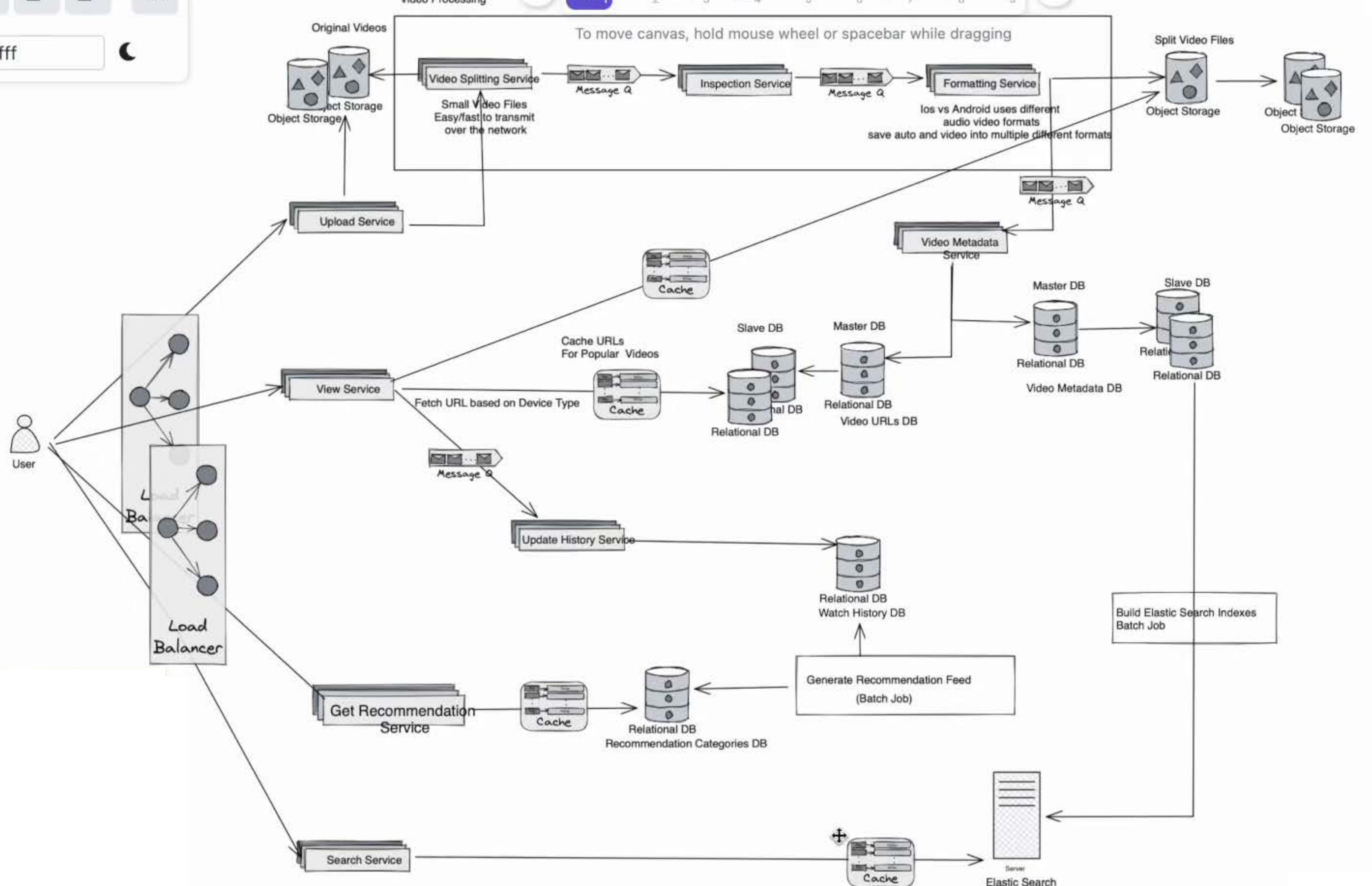




ffffff







ffffff

Introducing Excalidraw+
Try out now!

