Week -3
System Design Documentation

- Jai Kishan Timmapatruni

Distributed Systems:
The server or code that is executing doesn't lie in one place it is distributed all around the world.

For fault tolerance, performance.

Design pattern: It is generally a software solution for a reusable solution to a commonly occurring problem within a given context in software design.

We have a model named publisher-subscriber model where a publisher has created the event ,and subscribers able to reach the event and react to it. And our model decides the pace at which we send the notifications.

Live Streaming Application:

Initial things to do:

Product requirement doc
|
features/abstract concepts
|
Data definitions
|
Objects
|
Databases

Primary targets:
→being able to see the video
→users should be able to like or comment
→we should make sure that none of our services fail if there is an outage.
→Extensibility
→build a system that can scale and extend as amd when requirements change.
→Testing

Requirements for Live streaming application:

- Streaming video
- Processing video
- Broadcasting
- Failproof
- Advertisement
- Reactions
- Disclaimers/News Flashes
- Degradation of video quality
- Multiple Device support

The network protocol is used to define how the electronic message is taken from one place to another. Ex: GRPC, HTTP, FTP.

So we have customer→server→Databases

Based on customers API request we try to perform operations on the database.

Like if we are storing data regarding a video like video id,name, timestack, data, .
But here we are mainly trying to focus like distributed systems on the back end part so lets talk about it.

For suppose you want to get the video so in the getVideoFrame() function you would request for video id, what type of device we are trying to stream for , offset(in what time the video needs to be streamed.) return type could be frames.

To post comment(we need id,data, author video id)
And for the database we can use a sql database where we can store user data(user id, data) video(id, data) and comment table(video id, user id,data, author).

On client side the API requires a different behavior like for suppose users have posted some comments, i need not need the comments update immediately they can be periodical.(non real time). But where as if we take in case of getting the video data we need each frame to time (in real time).

The most common network protocol is http which is a stateless server. No need to store any information. While passing the request you need provide the necessary information to pull up the data and perform the operation.

Client-side all we need to do is request and we let the server handle how the request needs to be sent back to us. And if the server crashers there will be no memory lost if we save it in a database and it remains as a stateless server. We can add new servers with no issue.

So now we trying to look what kind of protocol to use to transmit the video data.
TCP protocol: Reliable protocol
UDP protocol : Real time efficient protocol
Here we will try to use the WebRTC protocol which is a peer to peer one.

Most database solutions(MySQL PostgreSQL) define how exactly the client interacts with the database.

Elastic Search has http protocol

HDFS(cheap, easy to query and store very large files,Vimeo)
We can use a file system like Amazon s3

For suppose we want rawly dump all the comments of a user in the key value then we can use NoSQL instead of MySQL.

So now how do we convert the raw video data like from 480p to 360p for the customer based on the device.

In order to convert we can create programs in such a way that when we pass the raw video data, video quality data it will be converted to the respective quality.

The most common format which is sent through network h.264.

A map-reduce pattern can be used where we can take a single piece of video data like a 10-sec frame send it to free servers try to transform them to different resolution versions and compress them using free servers. In the end, we will store the various versions of compressed output in the database.

So now after getting the response from data base the following data should be displayed using WebRTC. However, it is known that WebRTC is mainly used for video conferences than for streaming.

MPEG- DASH where dash stands for dynamic adaptive streaming over HTTP.
We also have an HLS protocol used for IOS or Mac devices.

We also have the content delivery network (CDN) which it can be useful when providing user-based data. For all the authentication we need to user-privacy we need to write it ourselves.

Things we need to do:

→Define the requirements as abstract concepts.
→Objects can be manipulated and queried using APIs on server.
→The data representation need to be stored in databases.

Like first we decide about the tools , then requirements and then how these requirements work with the API calls.

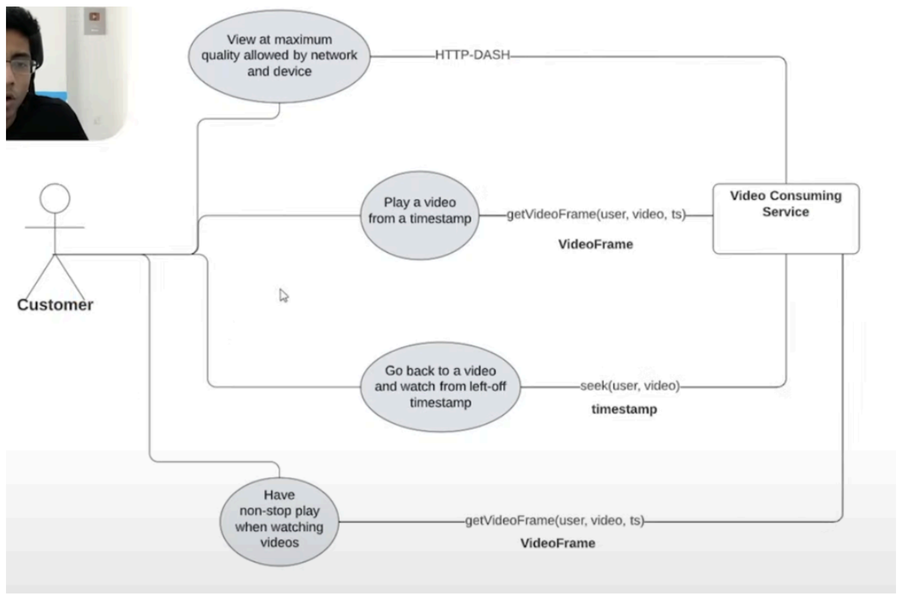
Till now we have discussed the approach by thinking about high level design.

Low level design:
Core functionality of a user is to view the video.
→fetch video
→view video
→play the video till the end

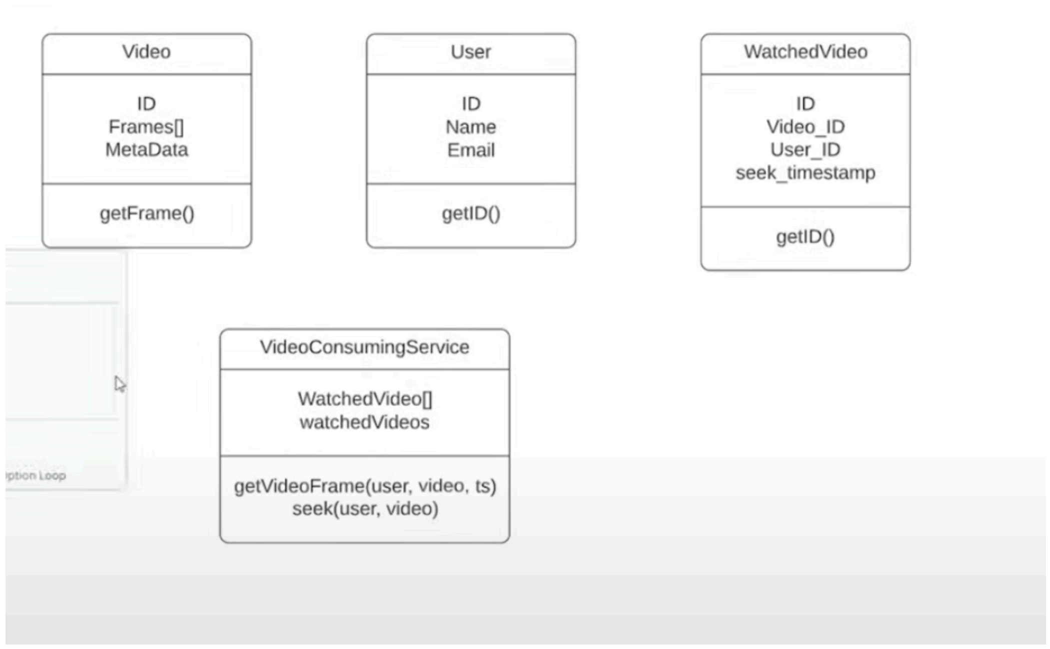
We should make sure that we get proper idea over the operations performed by the user like:
→play the video at a time stamp
→pause the video
→how the buffering of the video needs to be controlled.
→store where the particular video is stopped and the user should be able to login back and play at the respective timestamp.
→choose the quality of the video

Here we would be having three actors where
→customer- who requests for the operations.
→Admin - who makes sure that the operations are performed.
→videographer - who uploads the video with various video qualities.

A use case diagram for our design would be really helpful in putting out the usecases.

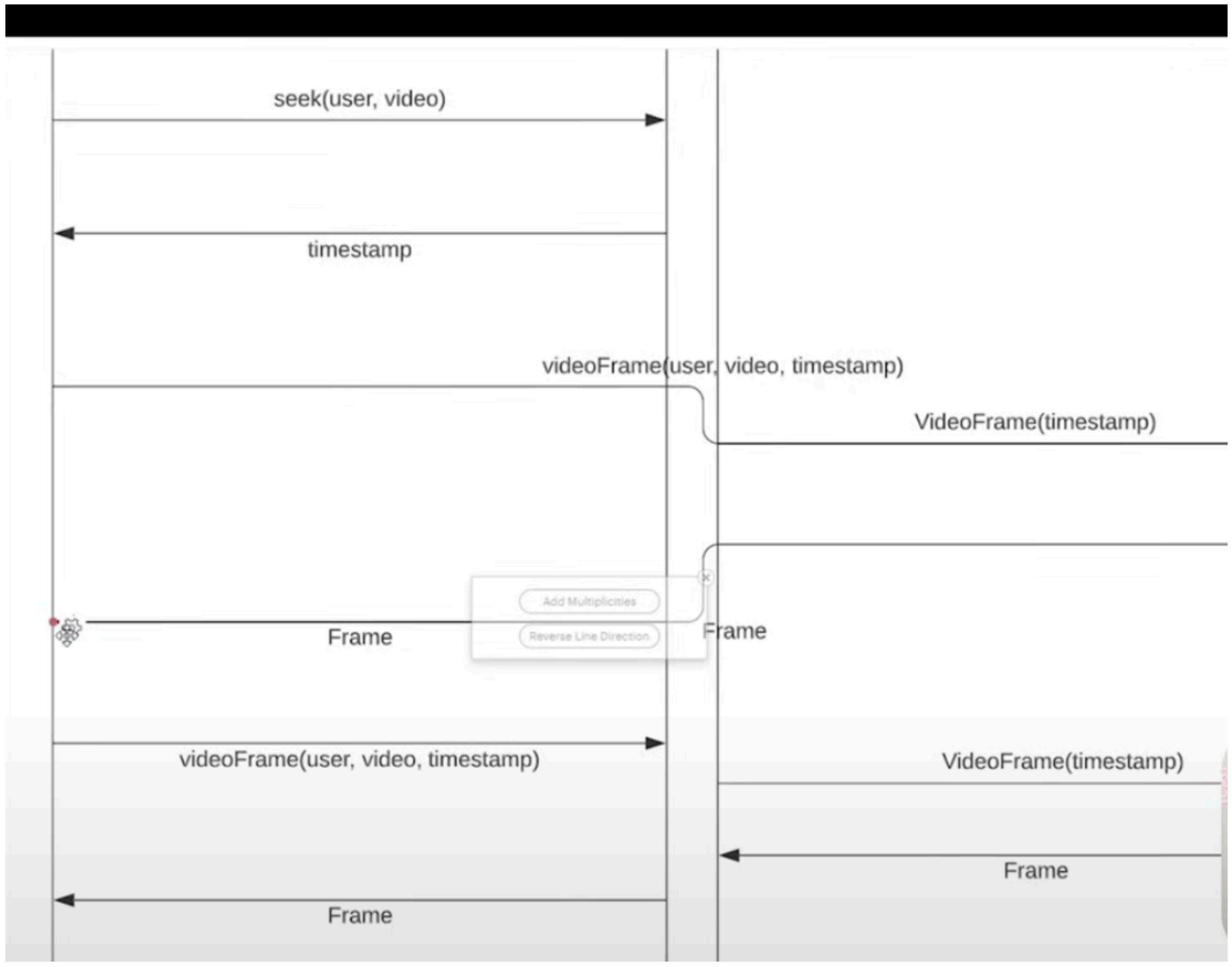

View at maximum quality allowed by network and device
HTTP-DASH
Play a video from a timestamp
getVideoFrame(user, video, ts)
VideoFrame
Video Consuming Service
Customer
Go back to a video and watch from left-off timestamp
seek(user, video)
timestamp
Have non-stop play when watching videos
getVideoFrame(user, video, ts)
VideoFrame

Class diagram:
This helps in defining what state and behaviors are performed by the user.



Sequence diagram:
We need to know the sequence of actions. What action is done first and what is done next.

User VideoConsumingservice Video


seek(user, video)
timestamp
videoFrame(user, video, timestamp)
VideoFrame(timestamp)
Add Multiplicities
Reverse Line Direction
Frame
Frame
videoFrame(user, video, timestamp)
VideoFrame(timestamp)
Frame
Frame