System Design

Large Scale distributed Systems

Eg Google Maps ↳ to prevent ~~single~~ [illegible] failure

Design Patterns

A software design pattern is a general, reusable solution to a commonly occuring Problem within a given context in software design. ↳ { Publisher subscriber Problem }

Eg.

Live Streaming Platform

Eg. Zoom
    Yt
    etc.

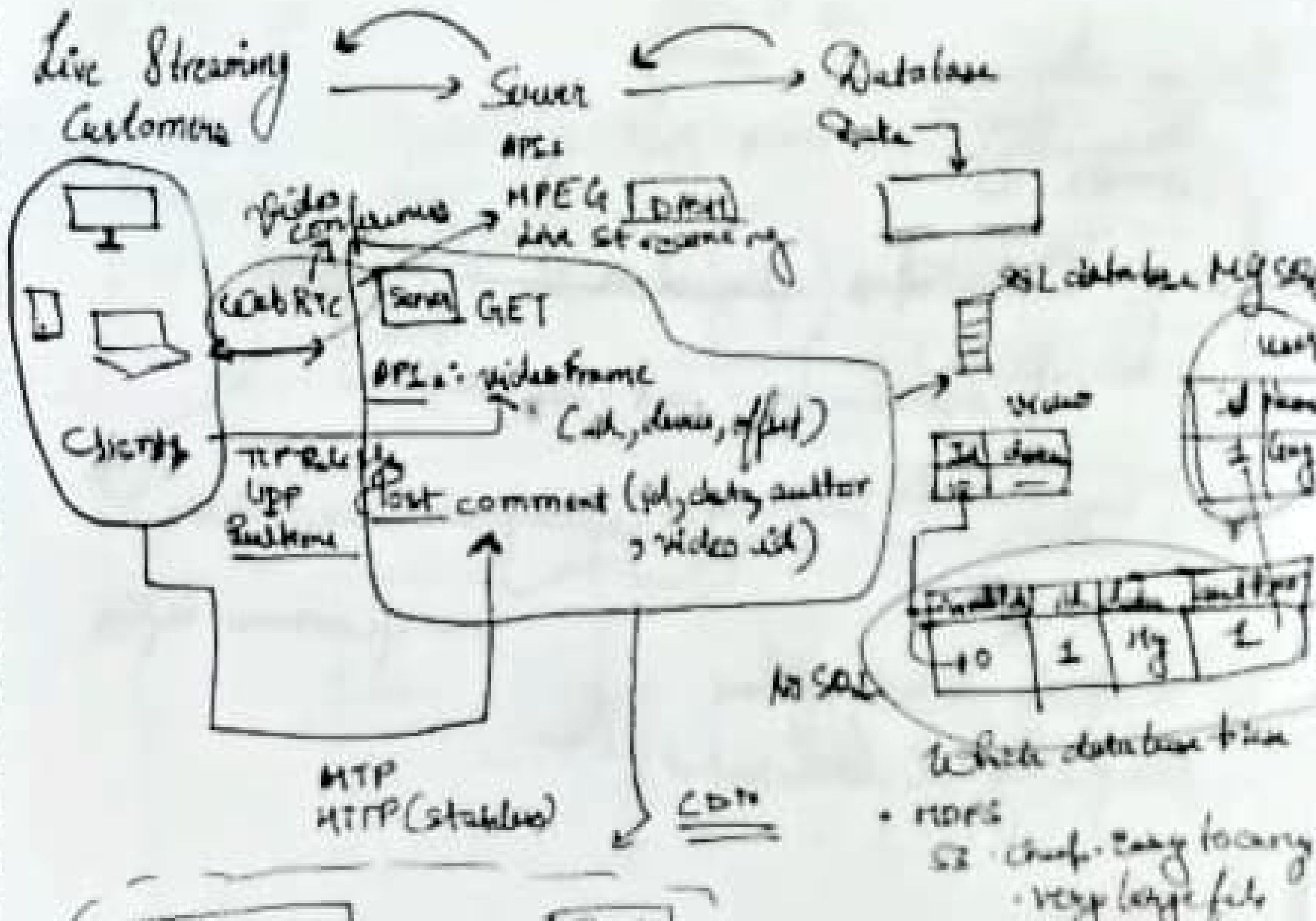
Issues
↓
Primary ⇒ [illegible] bandwidth.
↓
Secondary ⇒ ID, [illegible]
↓
etc.

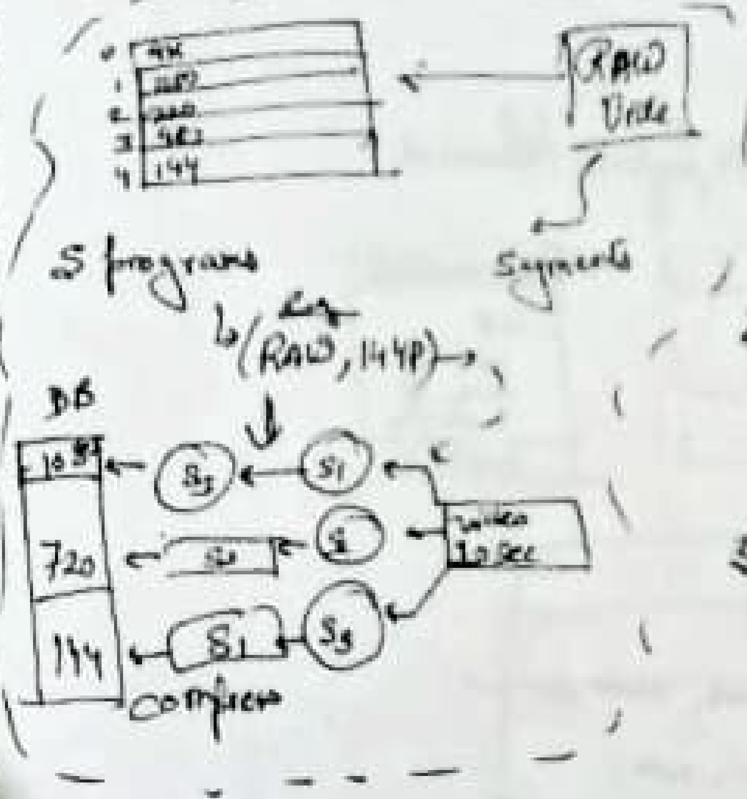
Basically

Product Requirements DOC [core]
⇓
Features/abstract concepts [core]
↳ should be extensible
⇓
Data Definitions
⇓
objects
⇓
Database

Note
• Build a system that scale & extend as and when requirements change

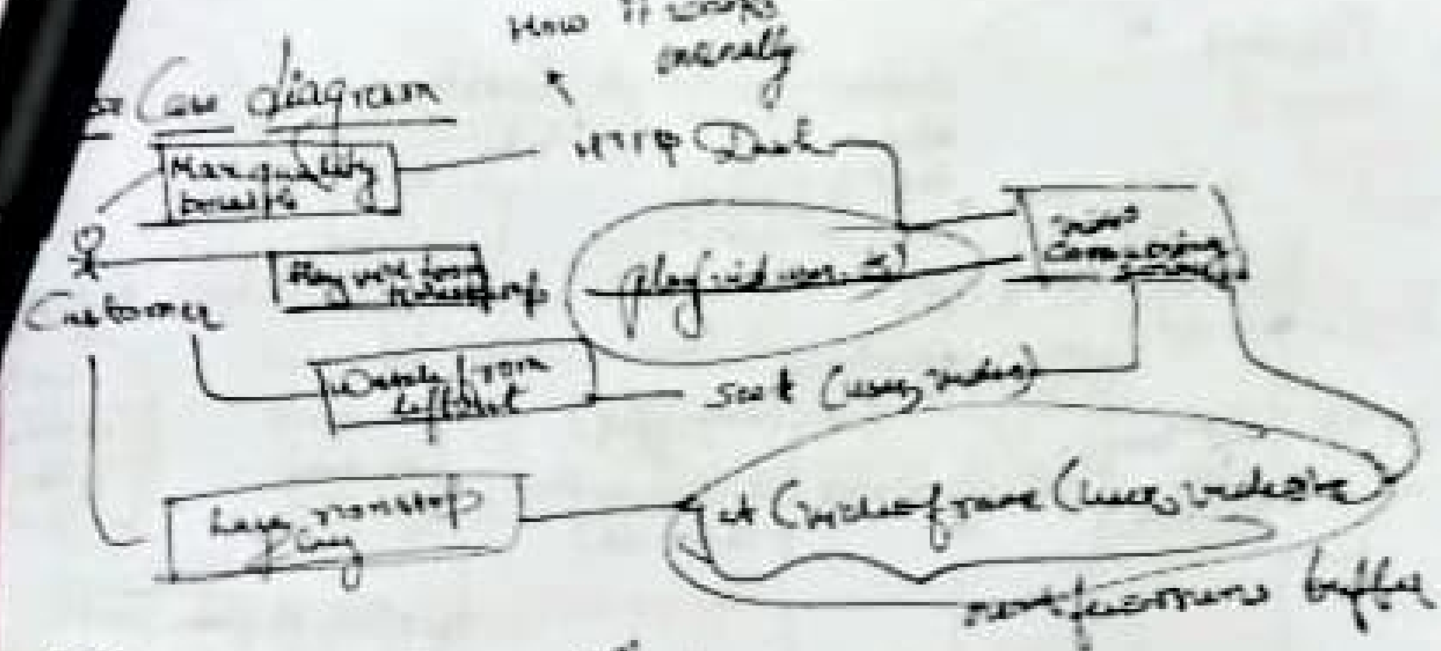
Testing ↱ flexibility





1. Define the requirements as abstract concepts (Objects)
2. Objects can be manipulated and queried using API's on server
3. The data representation need to be stored

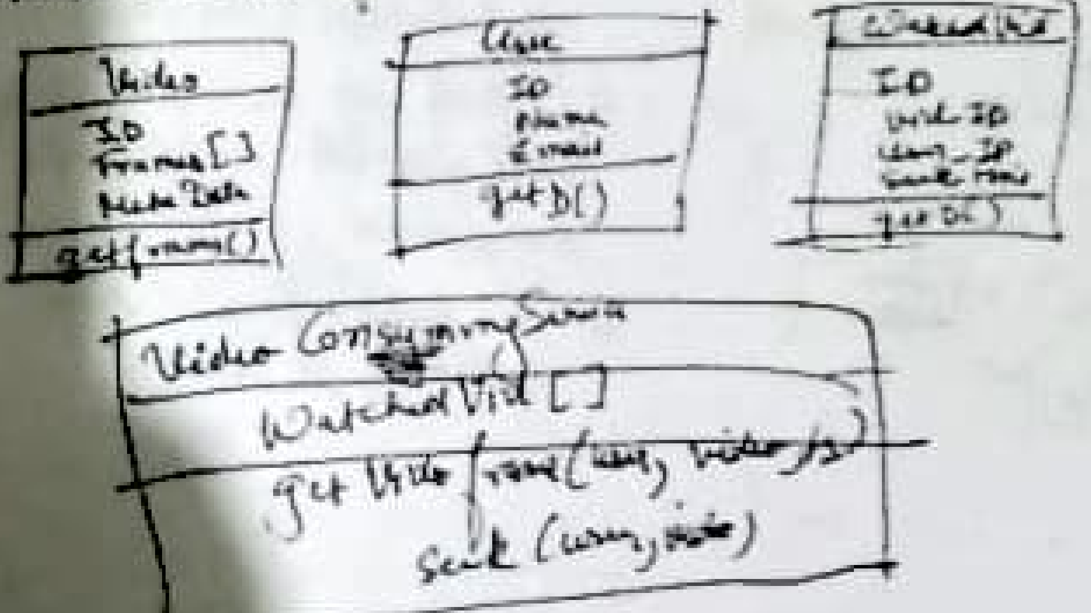
Use Case diagram



Q) Is playing the video different than fetching the feed content | Yes

Class diagram

States are data objects needed to perform behaviours


Video
ID
Frames[]
Meta Data
getFrames()
User
ID
Name
Email
getD()
WatchedVid
ID
Vid_ID
User_ID
Watch Time
getD()
Video ConsumingService
WatchedVid []
get Video frame(user, video, ts)
seek (user, video)