

0. System Design Strategy

Wednesday, July 28, 2021 9:57 AM

1. Requirement -> Functional
2. Constraints, Assumption, Risk, Scope
3. API
4. Data - Estimation - SLO, SLA, RTO, RPO
5. NFR
6. Data Modeling
 1. CA - SQL
 2. CP -> Mongo
 3. AP -> Dynamo, Casandra
 4. RPU / WPU
 5. Read vs Write
 6. PK, Sort, Secondary Index
 7. Shading, replication
 8. TTL
7. Tradeoff - Storage vs Compute
8. Scalability
9. Resiliency

System Design

Thursday, December 12, 2019

3:48 PM

System Design

Introduction

- Vertical vs Horizontal scaling
- CAP theorem
- ACID vs BASE
- Partitioning/Sharding Data
 - consistent hashing
- Optimistic vs Pessimistic Locking
- Strong vs Eventual consistency
- Relational DB vs NoSQL
- Types of NoSQL
 - key value
 - wide column
 - document based
 - graph based
- Caching
- Data center/Racks/hosts
- CPU/Memory/Hard drive/Network bandwidth
- Random vs Sequential read/write on disk

- http vs http2 vs websockets
- TCP/IP model
- ipv4 vs ipv6
- TCP vs UDP
- DNS lookup
- Https & TLS
- Public key infrastructure & Certificate Authority
- Symmetric vs asymmetric key
- Load Balancer → L4 vs L7
- CDNs & Edge
- Bloom Filters & Count-min sketch
- Paxos - Consensus over distributed hosts
 - leader election
- Design patterns & object oriented design
- Virtual machines & containers
- Publisher-Subscriber or Queue
- Map reduce
- multi threading, concurrency, locks, synchronization, CAS

- Solr, Elastic Search
- Blobstore like Amazon S3
- Docker
 - Kubernetes
 - Mesos
- Hadoop/Spark
 - HDFS

- Cassandra
- MongoDB / Couchbase
- MySQL
- Memcached
- Redis
- Zookeeper
- Kafka
- NGINX
- HAProxy

Capacity Planning

Monday, July 26, 2021 10:26 PM

1. Storage
2. Network bandwidth

Scale:

How much per document -
Bandwidth - Concurrent User
Read or Write Intensive -

Database:

CP: Document based
MongoDB -> Limitation - 16 MB

AP: Columnar
Cassandra, dynamoDB

Partitioning, Sharding?
Cross Partition Query

Tradeoff:

Redundant Data (Storage) / Scanning Query (Compute)

```
Scale
- 500M MAU
- Distributed across globe
- Equally load across the day
- 24M daily active users
- 1M users per hour
- .1M users creating board
- .9M users would see the boards sharing
- Read heavy service

- .1 M users per hour
- .1M
- 100K/60 new board per min
- 1.6k boards per min
- per board 5 links
- 1.6k * 5 links save per min
- 900K/60 view boards per min
- 15k boards per min
```

Capacity Planning

$10^3 = \text{KB}$
 $10^6 = 1 \text{ MB}$
 $10^9 = 1 \text{ GB}$

.4 GB = 400 MB

.4 * $10^3 = 400 \text{ MB}$

$500 * 10^6 = 500 \text{ Million}$

$1 \text{ B} = 1 * 10^9 = 1 \text{ Billion}$

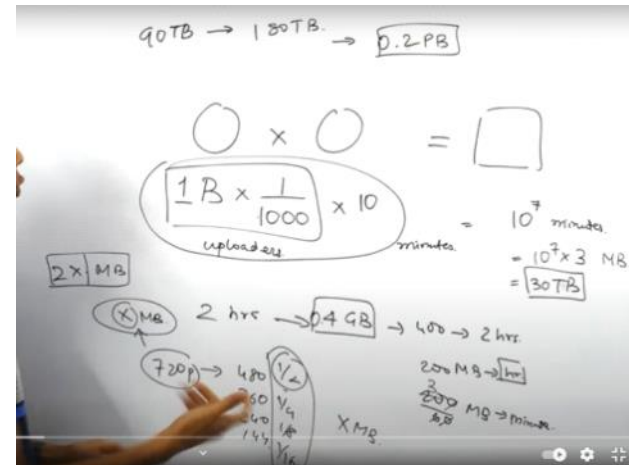
$10^9 \text{ users} * 1/1000 \text{ Uploads} * 10 \text{ minutes} = 10^7 \text{ minutes}$

Lets Assume:

2 hours video = 4GB => Filter => .4 GB = 400 MB per 2 hours
= $400 \text{ MB} / 2 * 60 = 3 \text{ MB per minutes}$

Total Size required = $10^7 * 3 * 60 = 3^7 * 10^3 = 30 \text{ TB per minutes}$

Fault + Reduandancy => $3 * 30 \text{ TB} = 90 \text{ TB}$
Different resolution => $90 * 2 = 180 \text{ TB} = .2 \text{ PB}$

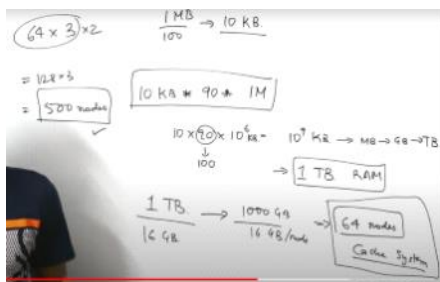


Estimate YouTube's daily video storage requirements

NO. OF USERS = 1B
RATIO OF UPLOADERS:USERS = 1:1000
NO. OF UPLOADERS = $1\text{B} / 1000 = 1\text{M}$
AVG. LENGTH OF VIDEO = 10 MINUTES
TOTAL UPLOADED VIDEO = $1\text{M} * 10 = 10\text{M MINUTES}$
TOTAL VIDEO SIZE = $10\text{M} * 3\text{MB} = 30 \text{ TB}$

2 HOUR MOVIE AVG. SIZE = 4 GB
OPTIMISED CODEC AND COMPRESSION SAVINGS = 90%
2 HOUR YOUTUBE VIDEO AVG. SIZE = $4 / 10 \text{ GB} = 0.4 \text{ GB}$
1 MINUTE YOUTUBE VIDEO AVG. SIZE = $0.4 / 120 \text{ GB} = 3 \text{ MB}$

Caching



Per original image = 1 MB filter image 10 KB

$10 \text{ KB} * 90 \text{ days most recent} * 1 \text{ M image}$
=> $10^3 * 100 * 10^6 = 10^{11} \sim 1 \text{ TB}$

Image Capacity Planning

Example used Twitter
Total users : 500 million
Active users : 2 million

Use case-3

Use case-3

- Database Capacity planning

Let's assume you have a table with columns as **ID, Name, Date of Birth**

Field Storage : **ID=3**, **Name=21**, **Date of Birth=6** Total = **30**

Table size: $500000 \times 30 \times 1.5 = 22500000 = 22 \text{ MB}$

Index size = number of records in the table being indexed $\times (7 + \text{number of fields indexed} + \text{field storage}) \times 2$

$= 500000 \times (7 + 1 + 30) \times 2 = 38000000 = 38 \text{ MB}$

Database size = *Sum of all table sizes + Schema size (100 MB) + index size*

Reach goals

Example used Twitter

Total users : **500 million**

Active users : **2 million**

Average photo size : **200KB**

1 million photos every day

Total space for one day : $1 \text{ M} \times 200\text{KB} = 200\text{GB}$

Total space required for one year = $365 \times 200 \times 1 = 73 \text{ TB}$

1. Social

Friday, September 24, 2021

10:19 AM

1.1 Twitter

Tuesday, July 27, 2021 1:35 PM

Core functionalities:

- User is share content (Text and Images) - Tweet
- User Profile Management
 - Profile details
 - User will be following other users
- Feed : According to each user.
 - Based on users is being followed
 - Latest on top
- Hashtag Search
 - All public tweets

Persistence Layer:

- User Profile Data
 - Partitioning based on user_id (Mongo-DB)
- Tweets of Users
 - Text
 - Images
 - Properties
 - User ID
 - Content
 - Hashtags
 - Time Of Posting
 - Sharding By UserID
 - Cassandra
- Save users followers
 - User Profile DB separate schema (Partitioned Based on user ID)
 - UserID -> Followed Users
 - Followers -> UserID (friends ID)
- Hashtag based indexes of Tweets
- Image Server
 - AWS S3

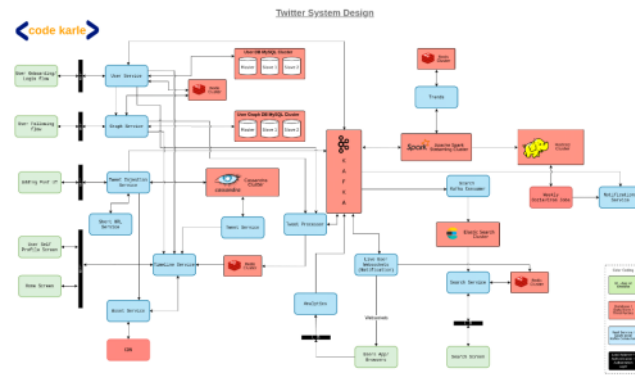
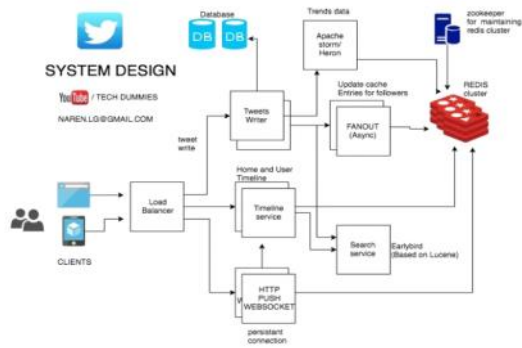
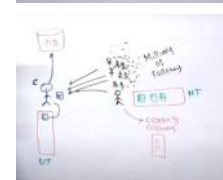
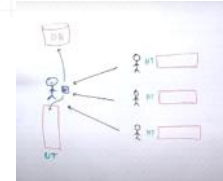
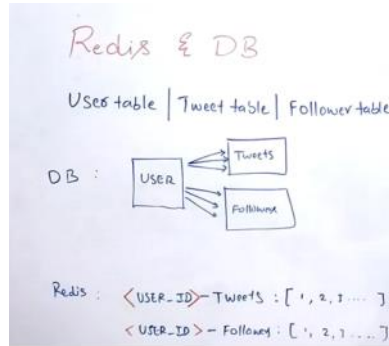
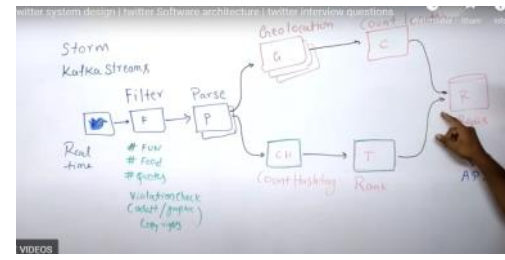
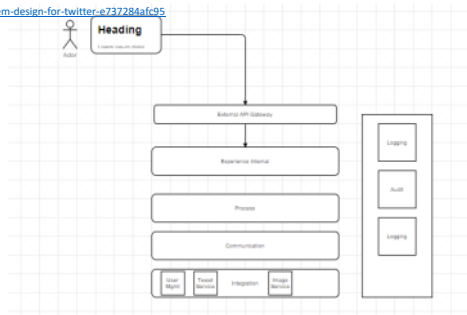
1. Tweet
2. Retweet
3. Timeline
4. Hashtag

Design:

1. Pre compute Timeline
 - a. Active Users
 - b. Threshold Users
2. Hybrid

Read heavy
Eventual consistency
Storage

<https://medium.com/@narengowda/system-design-for-twitter-e737284af95>



Capacity Planning :

Capacity Estimation: Back-Of-The-Envelope Calculation for System Design | Twitter Example

Total number of users = 600 million
Daily active users = 200 million
25% of the users tweet, each 2 tweets per day
 $200 * 25\% * 2 = 100$ million new tweets per day
 $200 \text{ million} * 100 = 20\text{B}$ views per day

$$600 \times 10^6 \text{ Users}$$

$$200 \times 10^6 \text{ Active users}$$

$$25\% \text{ Active users tweet}$$

$$200 \times 10^6 \times 25\% \times 2 = 100 \text{ million write}$$

$$200 \text{ million read} = 100 \text{ tweets}$$

$$200 \text{ million} \times 100 = 20 \times 10^9 \text{ read}$$

100 characters per tweet on an average

2 bytes per character

200 bytes + 50 bytes = 250 bytes

Tweet Text Data = 25GB/day

$$100 * 10^6 * 250$$

$$1000 * 1000 * 1000$$

$$100 \times 10^6 \times 250$$

$$= 25 \text{ GB}$$

100 characters per tweet on an average

2 bytes per character

200 bytes + 50 bytes = 250 bytes

Tweet Text Data = 25GB/day

1/20 will have an image associated with it: 200KB

1/100 will have a video associated with it: 2MB

$$\frac{100 * 10^6 * 200}{20 * 1000 * 1000 * 1000} + \frac{100 * 10^6 * 2}{100 * 1000 * 1000}$$

Image + Video = (1 + 2) = 3 TB /day

Total final storage: 25 GB + 3 TB ~ 3 TB/day

Incoming to server = 3 TB/day ~ 23 MB/sec

Outgoing text tweet data = 20B * 250bytes ~ 60 MB/sec

$$3 \times 10^9 \times 250 = 750 \times 10^9 \text{ bytes}$$

$$= 750 \text{ GB}$$

Total final storage: 25 GB + 3 TB ~ = 3 TB/day

Incoming to server = 3 TB/day ~ = 23 MB/sec

Outgoing text tweet data = 20B * 250bytes ~ = 60 MB/sec

	$20 * 10^3 * 10^6 * 250$

	$86400 * 1000 * 1000$

Incoming to server = 3 TB/day ~ = 23 MB/sec

Outgoing text tweet data = 20B * 250bytes ~ = 60 MB/sec

1/20 will have an image associated with it: 200KB

Outgoing image data ~ = 2.5 GB/sec

1/100 will have a video associated with it: 2MB

When we show 5 videos, users watch 1 (1/5)

Outgoing video data ~ = 1 GB/sec

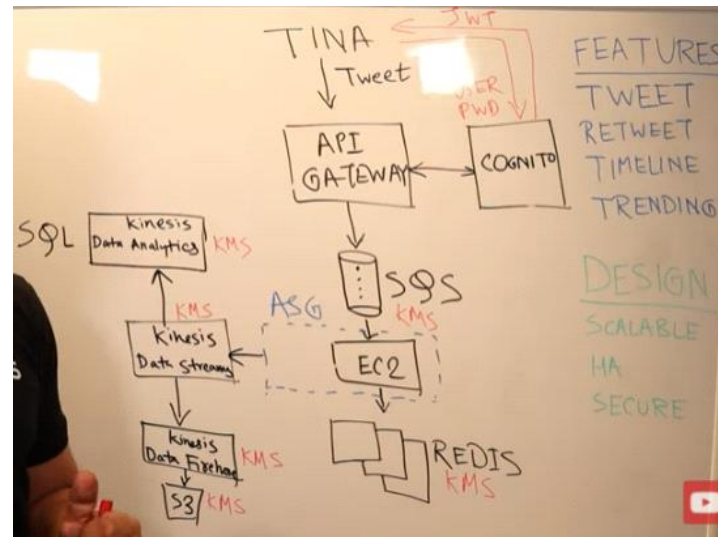
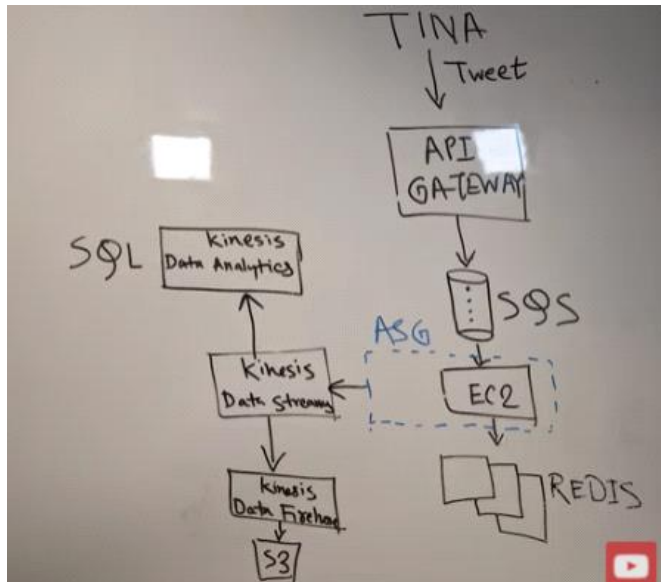
Total Outgoing = 60 MB/sec + 2.5 GB/sec + 1 GB/sec ~ = 3.5 GB/sec

$$\frac{3 \times 10^{4+2}}{86400 \times 10^3 \times 10^6} = 23 \text{ MB/sec}$$

$$60 \times 60 \times 24$$

raj

Tuesday, September 28, 2021 5:25 PM



Twitter

Thursday, September 23, 2021 9:23 AM

Functional

- Tweet
- Retweet
- Timeline
- Trending

Non Functional

- Business Continuity and Disaster Recovery
 - o Fault Tolerance
- Cost
- Security
- Monitoring
- Scalability and Reliability
- Deployment
- Hybrid

Architecture Patterns:

- **Communication**
 - Asynchronous
 - Message Broker
 - Event Streaming
 - SAGGA Orchestrator
- **Security**
 - Authentication
 - Authorization
 - Federator
 - Reverse Proxy
- **Data Pattern**
 - Access Trends
 - CQRS
 - Relevance Tags
 - Repository
 - Unit of work

Data Storage and Network bandwidth:

Total Users: $500 * 10^6$

Active Users: $200 * 10^6$

Write Message : $200 * 10^6 * 25/100 * 2 = 100 * 10^6$

Per Day = $200 * 100 * 10^6 = 20 * 10^9$

Storage: (bits)

Text = 250 b => $100 * 10^6 * 250 = 25 * 10^9 = 25 \text{ GB}$

Image 200 KB => $100 * 10^6 * 200 * 10^3 / 20 = 1 * 10^{12} = 1 \text{ TB}$

Video 2 MB

Network

Incoming: $20 * 10^{12} / 86400 = 34 \text{ MB /s}$

Outgoing : $20 * 10^9 * 250 / 86400 = 60 \text{ Mb/s}$

White board

Wednesday, September 22, 2021

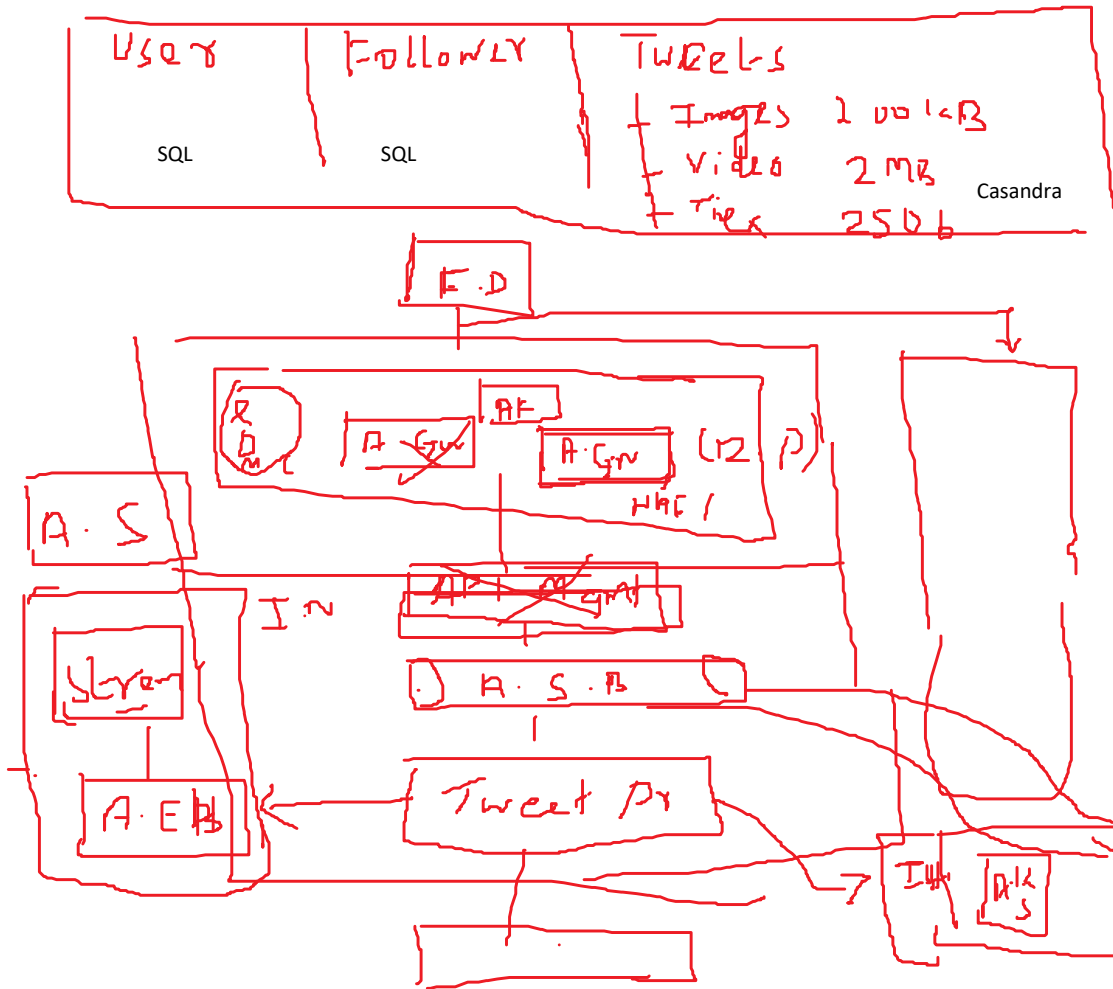
10:36 PM

Functional

- Tweet
- Retweet
- Timeline
- Trending

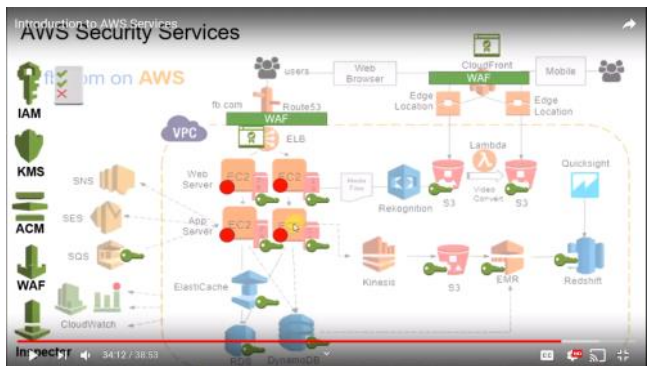
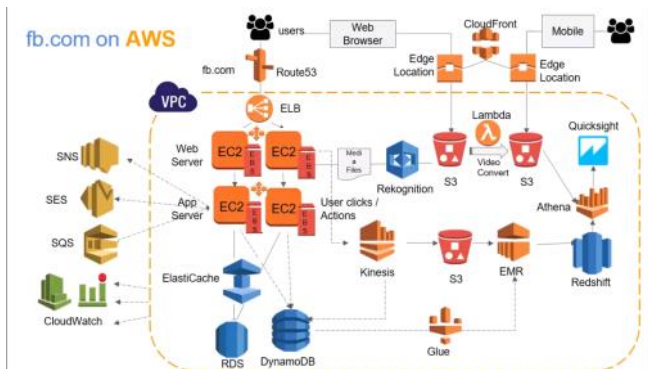
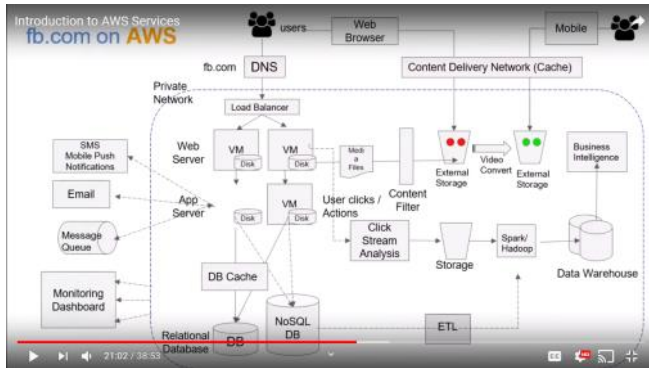
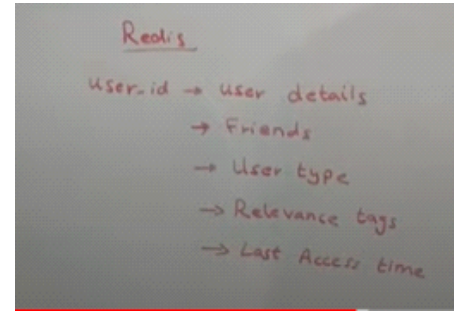
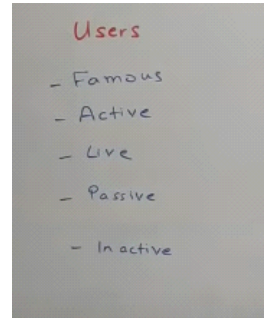
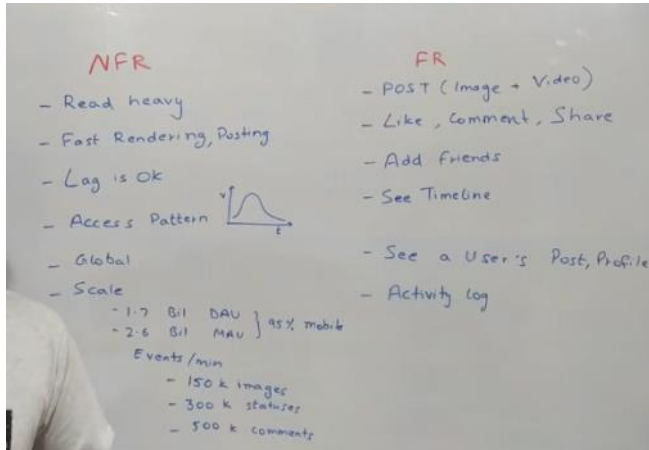
NI

- BC & FT
- Cost
- Security
- Monitor
- Scalable
- Design



1. 2. Facebook

Friday, February 7, 2020 9:25 PM





2. E-commerce

Friday, January 24, 2020 10:45 AM

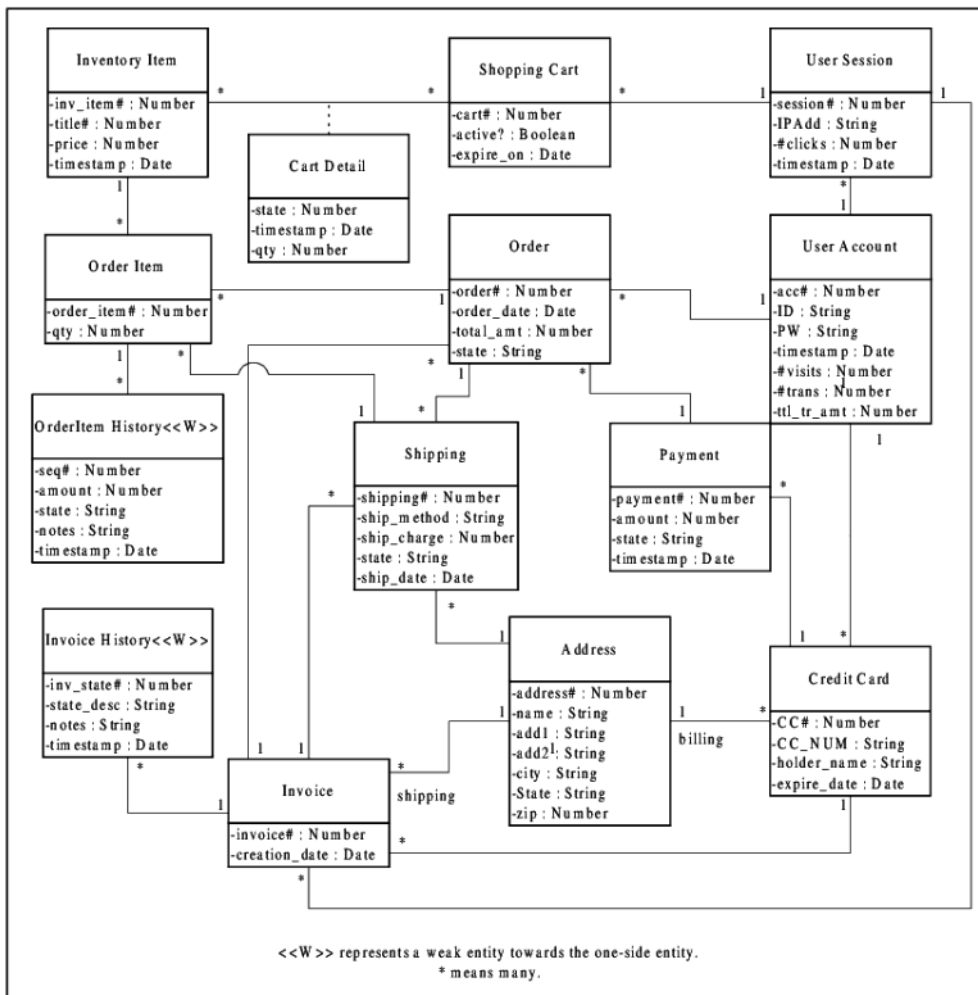
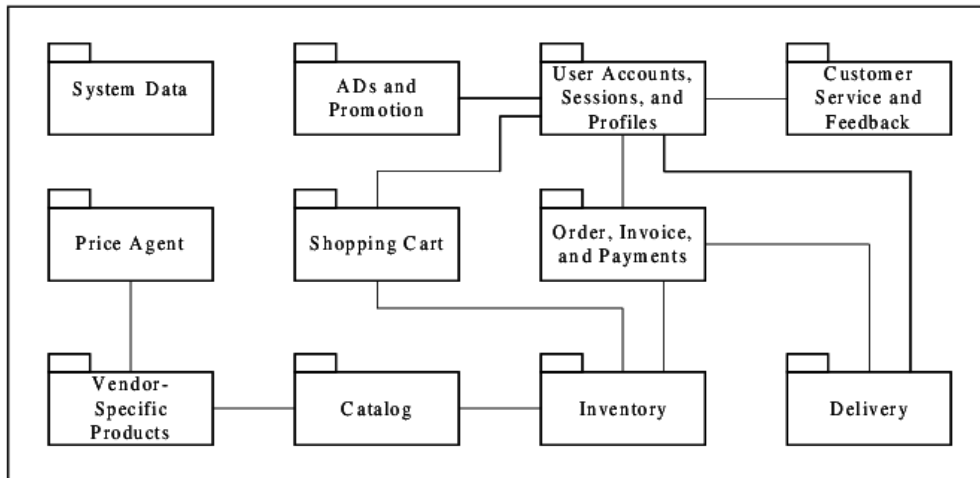
1. Customer
2. Place Order
3. Fulfillment Order
4. Validate Order
5. Shipping Order

Customer orders,
billing,
payment,
inventory,
shipping.

Insurance Business Domain
Claim
Billing
Invoice
Accounting
Corporate HR

https://www.researchgate.net/figure/A-simplified-database-schema-for-e-commerce-transaction-processing_fig2_2359510

https://www.researchgate.net/figure/A-simplified-database-schema-for-e-commerce-transaction-processing_fig2_2359510



API ✓

SOA

API

Get Write

Statofcl

objadi

SLI, SLO, SLA

SLA = 5,000R/s

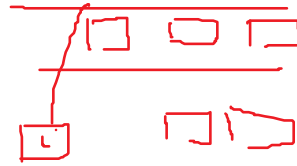
Latency

Request/Reply:-

U

Lab

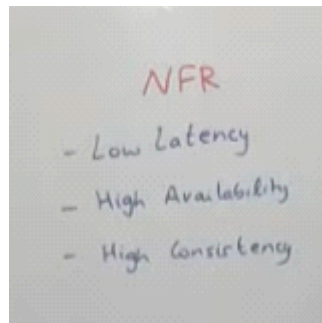
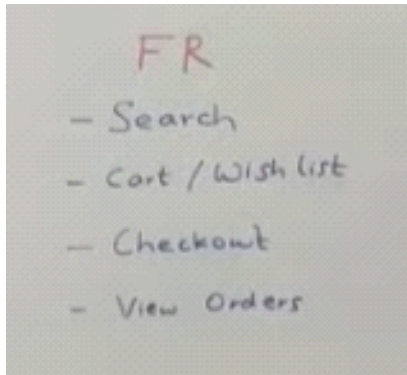
5



Functional

Friday, September 24, 2021

12:15 PM



Thursday, September 23, 2021 2:03 PM

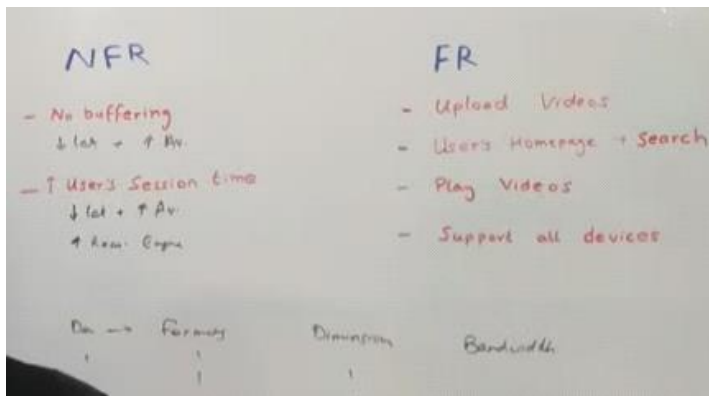


3. On-demand Video

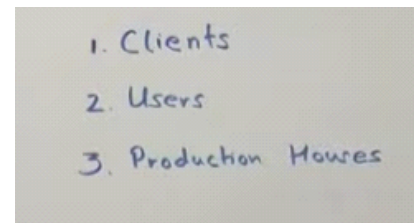
Friday, September 24, 2021 10:21 AM

3.3 Netflix 1

Friday, September 24, 2021 11:11 AM



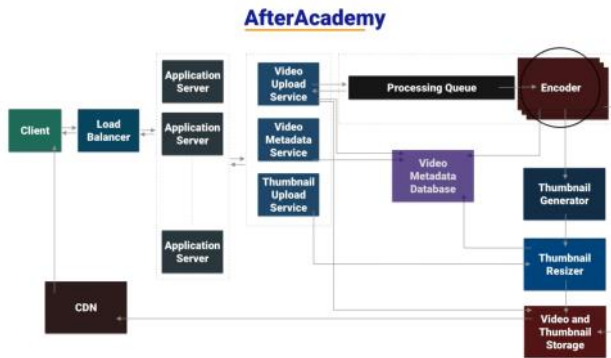
1. Devices
2. Format
3. Dimension
4. Bandwidth



3.1 YouTube

<https://www.youtube.com/watch?v=XkFL5sAmp-g&t=1s>

Friday, July 30, 2021 7:01 PM



Storage System

- Video Storage: HDFS
- Thumbnail Storage: S3 bucket

What More?

- Video duplication: Block-matching algorithm
- API for deleting a video
- Notification to the client

Database Design

- Video Metadata
- MySQL

VideoMetadata
videoID
title
description
thumbnailPath
videoRawPath
videoPath
userId

API Design

uploadVideo(accessToken, title, desc, video)

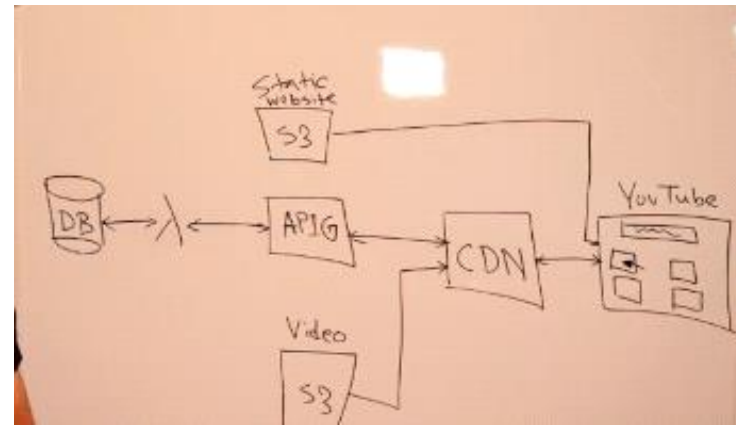
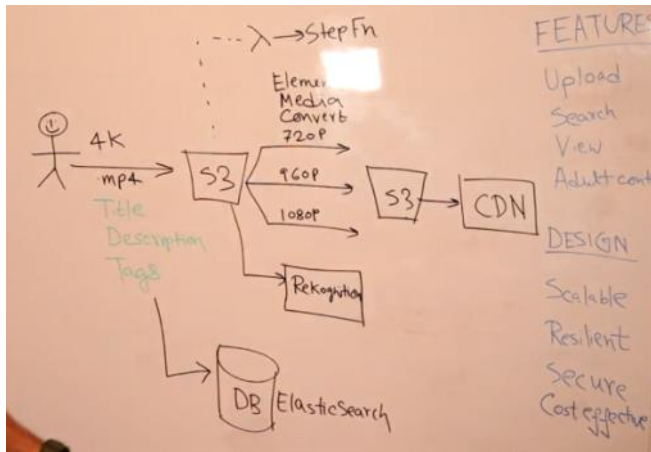
videoMetadata(accessToken, videoId, title, desc)

uploadThumbnail(accessToken, videoId, thumbnail)

3.2 Netflix - Raj

Friday, September 24, 2021 10:22 AM

Azure media service - Element media convert
Face deduction -> Rekognition



HTTP LIVE Streaming

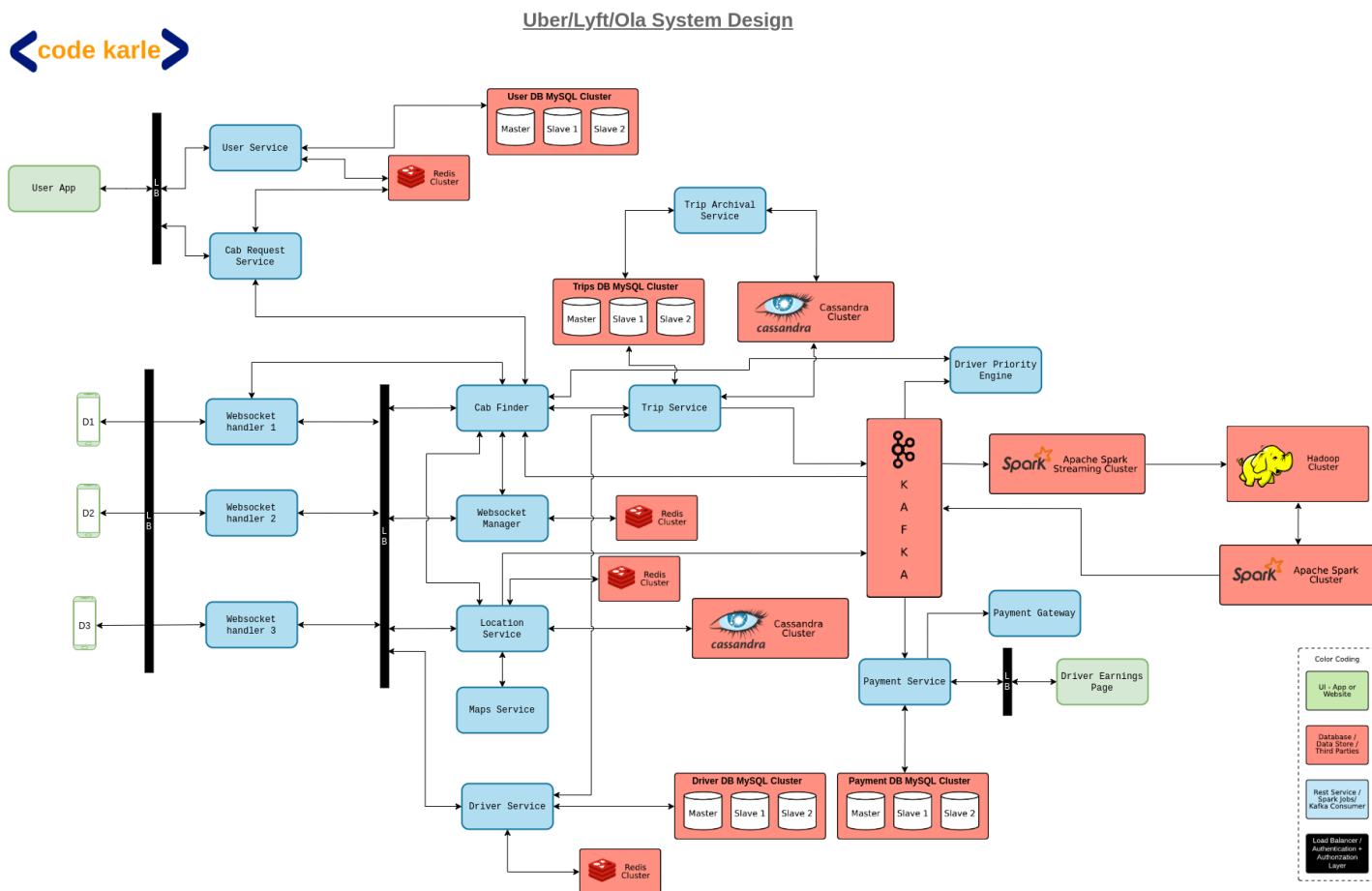
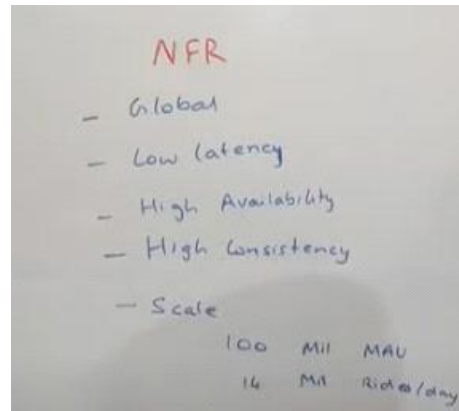
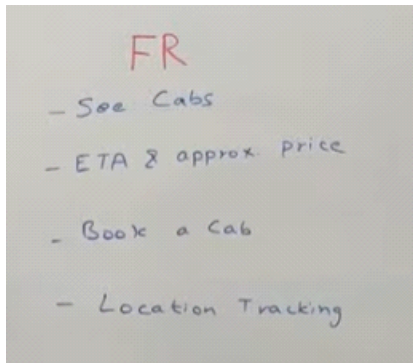
Ffmpeg

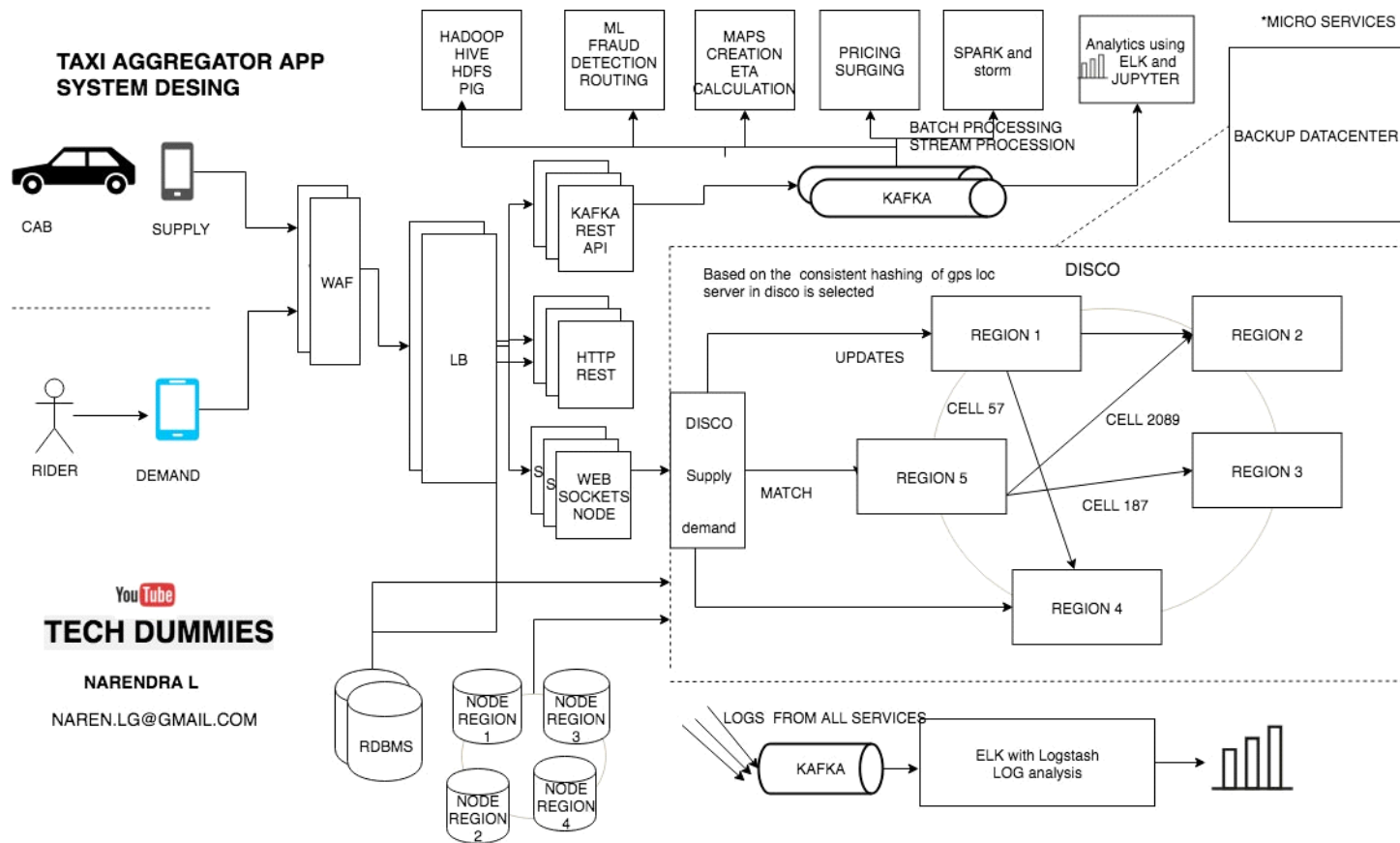
Tuesday, September 28, 2021 3:19 PM

```
ffmpeg -i test.mp4 -hls_time 10 -hls_playlist_type vod -hls_segment_filename "video_segments_%0d.ts" hls_master_for_test.m3u8 | I
```

4. Car Tracking

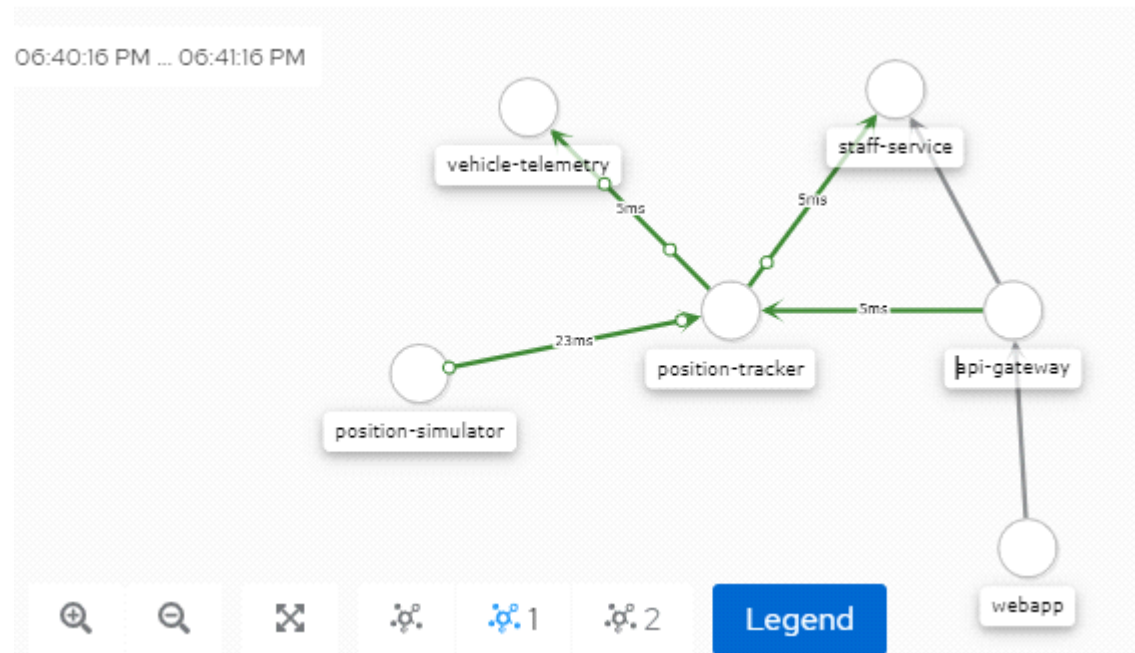
Friday, September 24, 2021 11:46 AM





4.1 Vehicle Tracker

Thursday, August 26, 2021 6:41 PM



5. Online Chat System

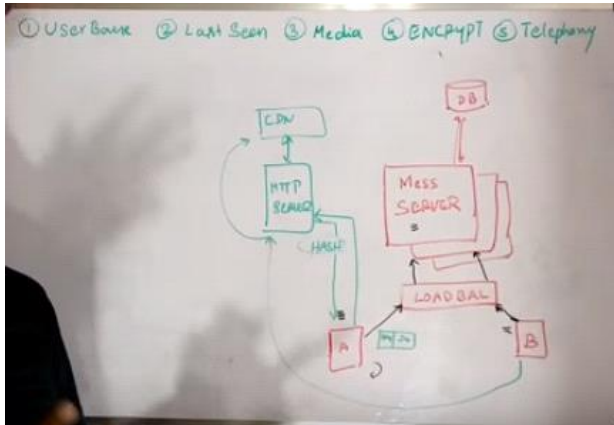
Thursday, December 12, 2019 11:06 AM

Protocol:

- TCP
- UDP
- WebScket
- BOSH
- Long Polling
- XMPP

Load Balancer Configuration

- Load
- Session

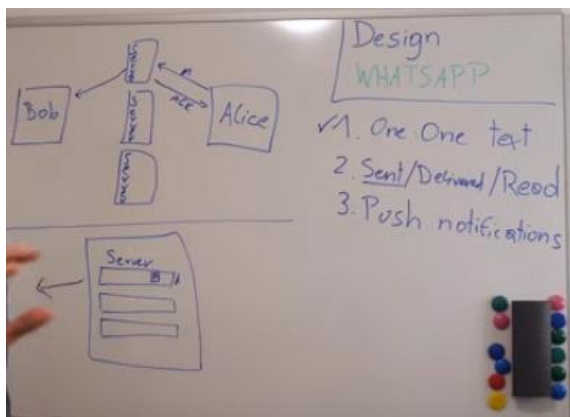


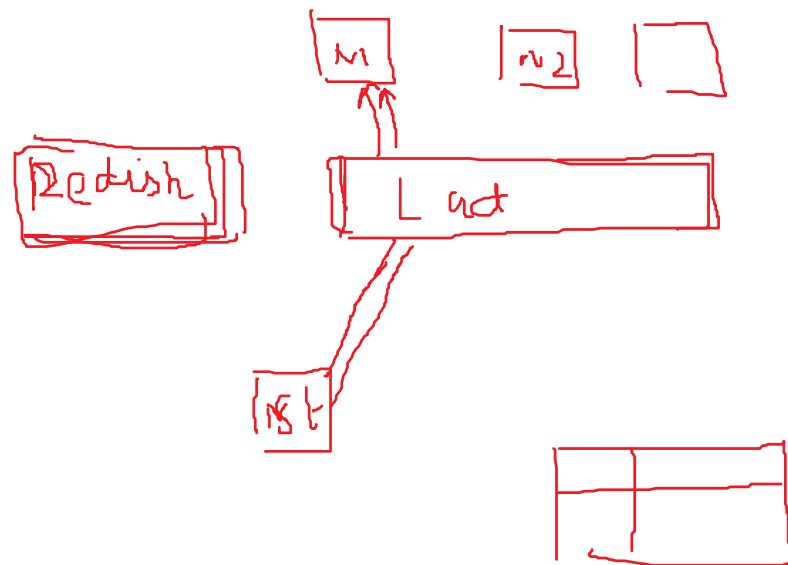
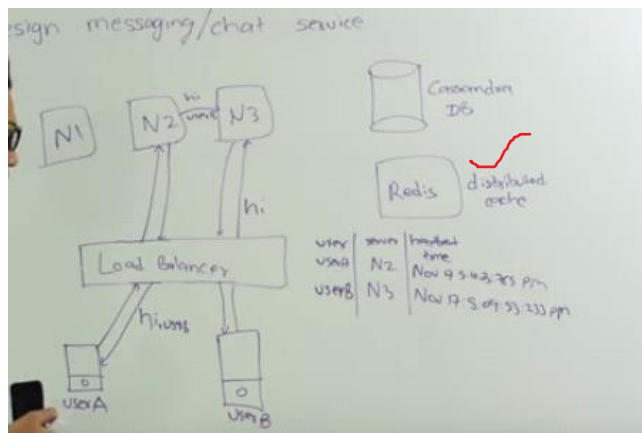
Queue Free when no message

Ephemeral
Queue health

High Availability

1. Ordering message [Session Affinity / Sticky Session]





cache

User Table

userid	username	connection-id	expirykey
user1	user2		
userA	userB	5	BABE57076
userA	userC	6	CA925289

Conversation Table

conversation-id	time	text	sender	recd
6	ts	hello	userA	

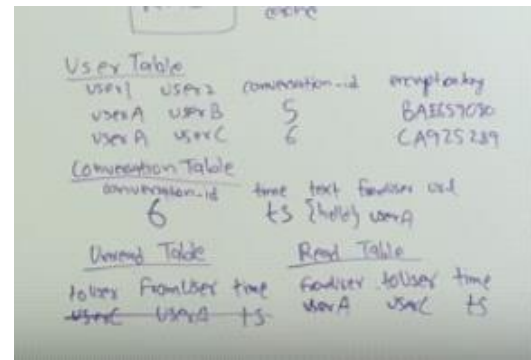
Write Table

userid	username	time
userA	userA	ts

Read Table

userid	username	time
userA	userC	ts

- ### System Design Introduction
- Features
 - Define APIs
 - Availability
 - Latency Performance
 - Scalability
 - Durability
 - Cons Diagram
 - Security & Privacy
 - Cost effective

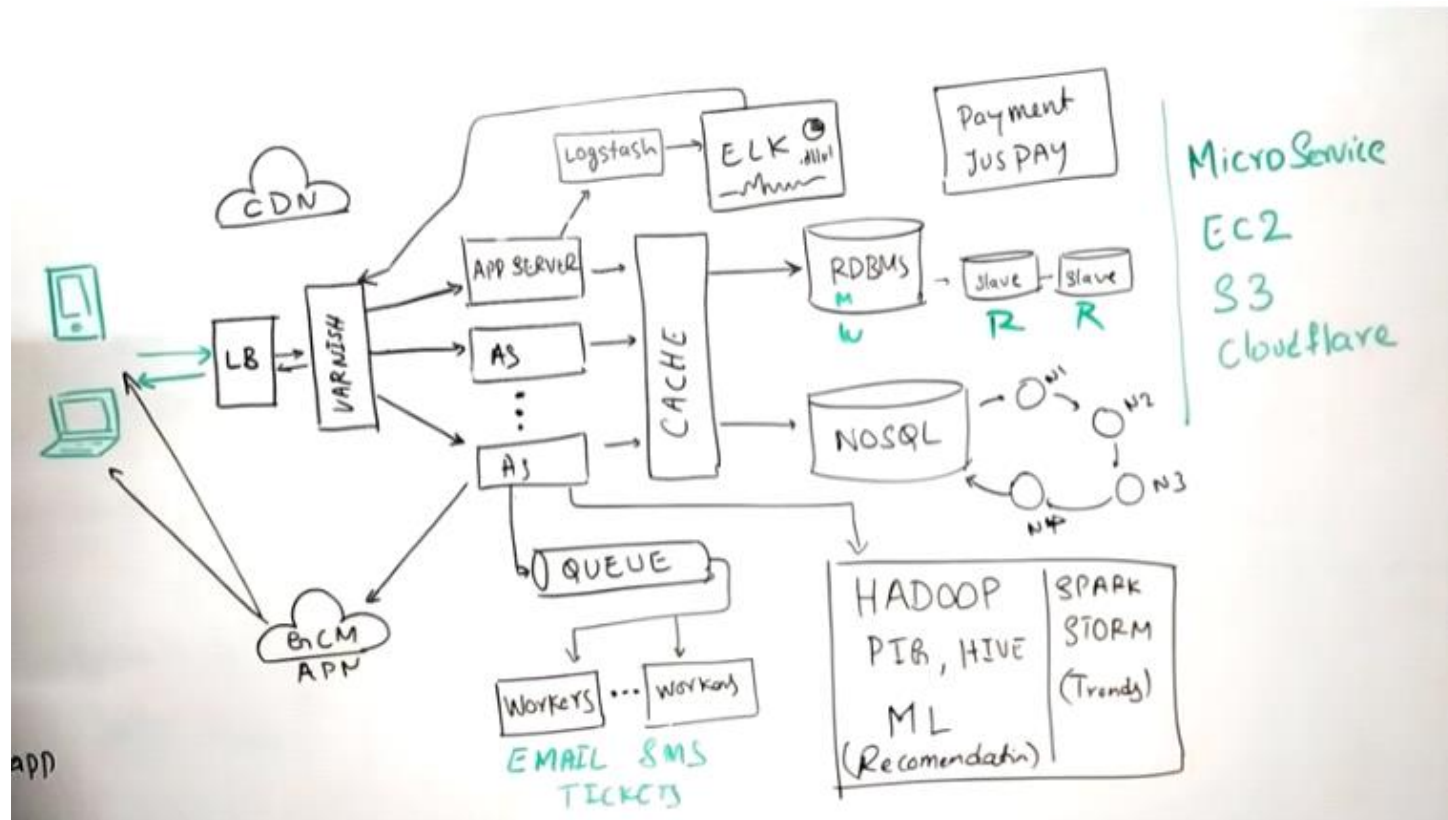


6. Online booking

Friday, September 24, 2021 2:18 PM

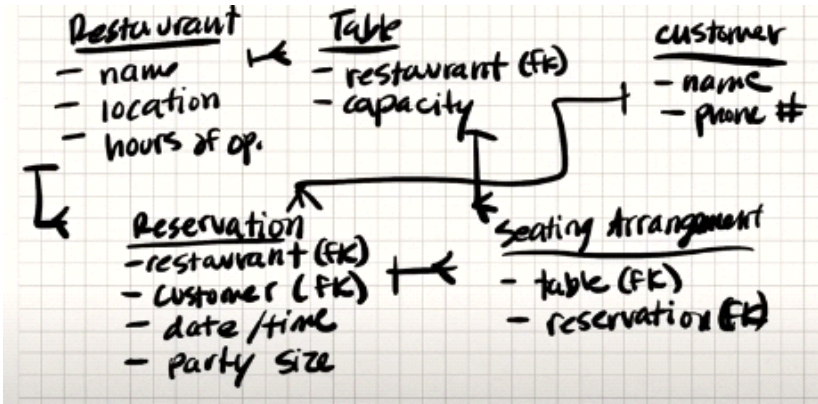
Movie

Friday, September 24, 2021 5:04 PM



Restaurant

Friday, September 24, 2021 4:51 PM



```

public interface ReservationService {
    List<Availability> getAvailability(Location l,
                                     Instant dt,
                                     int partySize);

    Reservation makeReservation(Reservation r,
                                Instant dt,
                                Customer c,
                                int partySize);
    throw Exception;

    Availability
    - Restaurant
    - List<Instant>

    void cancelReservation(Reservation r);
}
    
```

Hotel Booking

Friday, September 24, 2021

3:36 PM

Design a system for Hotel booking / 6yo / Teivagol mmt...

① Reservation on room/rooms for a certain period ^{booking} Amazon.
↳ should happen.

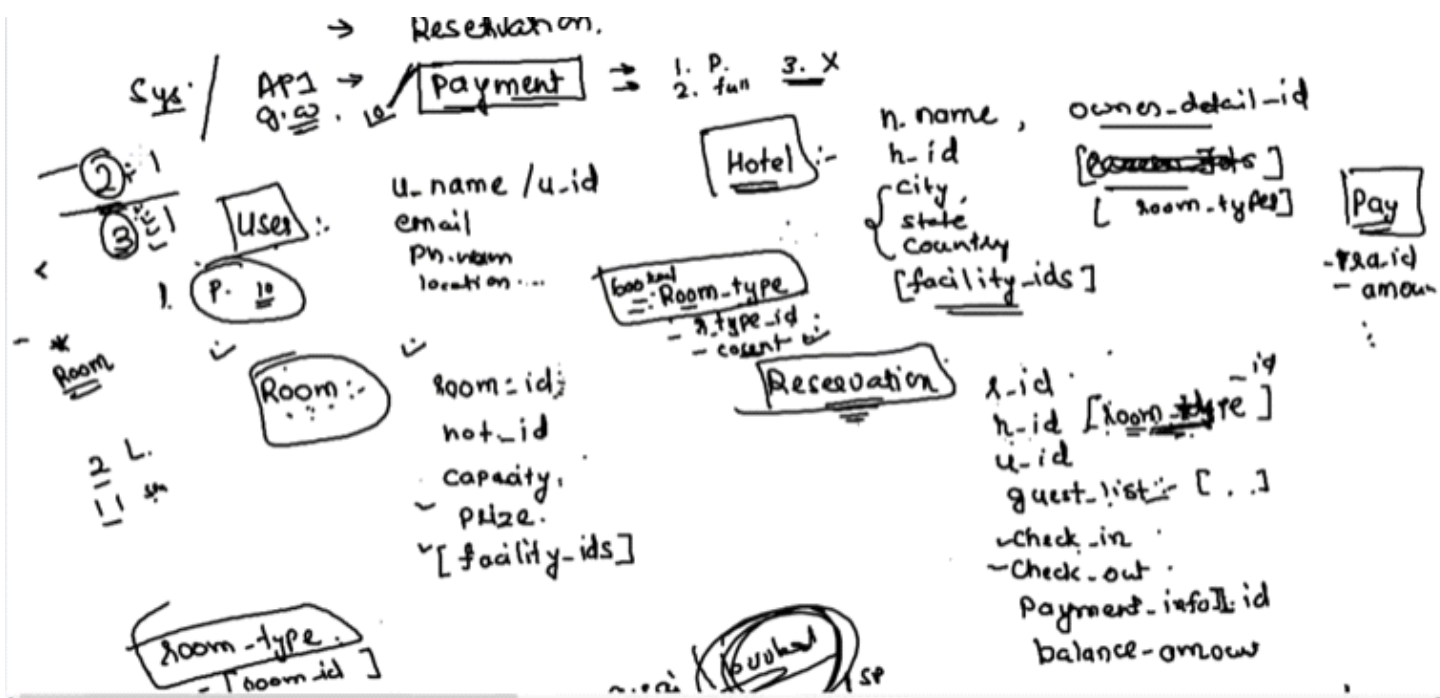
→ guests / users :- db ...

→ Hotel / hotel :-

→ Room.

→ Reservation.

Sys / API → Payment → 1. P. 2. full 3. X





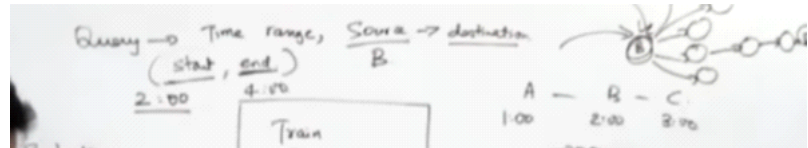
IRCTS

Friday, September 24, 2021 2:42 PM

- ① Schedule Trains.
- ② Book a single seat in a given train.
↳ multiple
- ③ Train → Route → (Stop, time)
Seating Arrangement
- ④ 3 month - advance booking.
- ⑤ Parallel seat booking.

Schedule

TrainID	Source	Destination	EDT	EAT
1	A	B	1:00	2:00
1	(B)	C	2:10	3:00
2	B	D		



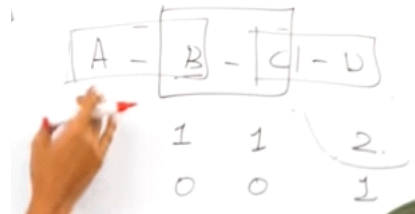
Ticket

Train	User	Seat
1	123	-
2		
...		
60		

ROUTE

TrainID	Destination	Ref	No. of Seats
1	(A)		2
1	B		1
1	C		1
1	D		1

Lock



7. Google Map

Friday, September 24, 2021 2:20 PM

8. Notification Service

Friday, September 24, 2021 2:20 PM

8. Pinterest

Tuesday, July 27, 2021 9:49 AM

```
Requirements
1 Requirements
2 - Private board
3 - Share board to be able to see
4 - only Links
5 - No limit on links
6
7
8 Non functional requirements
9 - Auth management, security
10 -
11
12 Functional Requirement
13 - Add new board
14 - List all my boards
15 - View a board via id
16 - permission
17 - no clone, comment, like
18 - timeline
19 - Eventual consistency
20 -
21 - Share
22 - Individually add or remove user
23
24 - List all boards shared with me
25 - sort shared time
26
```

```
Scale
- 500M MAU
- Distributed across globe
- Equally load across the day
- 24M daily active users
- 1M users per hour
- 1M users creating board
- .9M users would see the boards sharing
- Read heavy service

- .1 M users per hour
- .1M
- 100K/60 new board per min
- 1.6k boards per min
- per board 5 links
- 1.6k * 5 links save per min
- 900K/60 view boards per min
- 15k boards per min
```

```
DB design
- Boards
  - id (UUID)
  - owner_id (UUID)
  - name (100 char)
  - description (1000 char)
  - created_at (datetime)
  - updated_at (datetime)

- Links
  - id (UUID)
  - original_url (char 1000)
  - hash_url (char 50)
  - metadata (json)
  - title
  - og:detail (json)

- BoardLinks
  - board_id
  - link_id

  unique - board_id + link_id

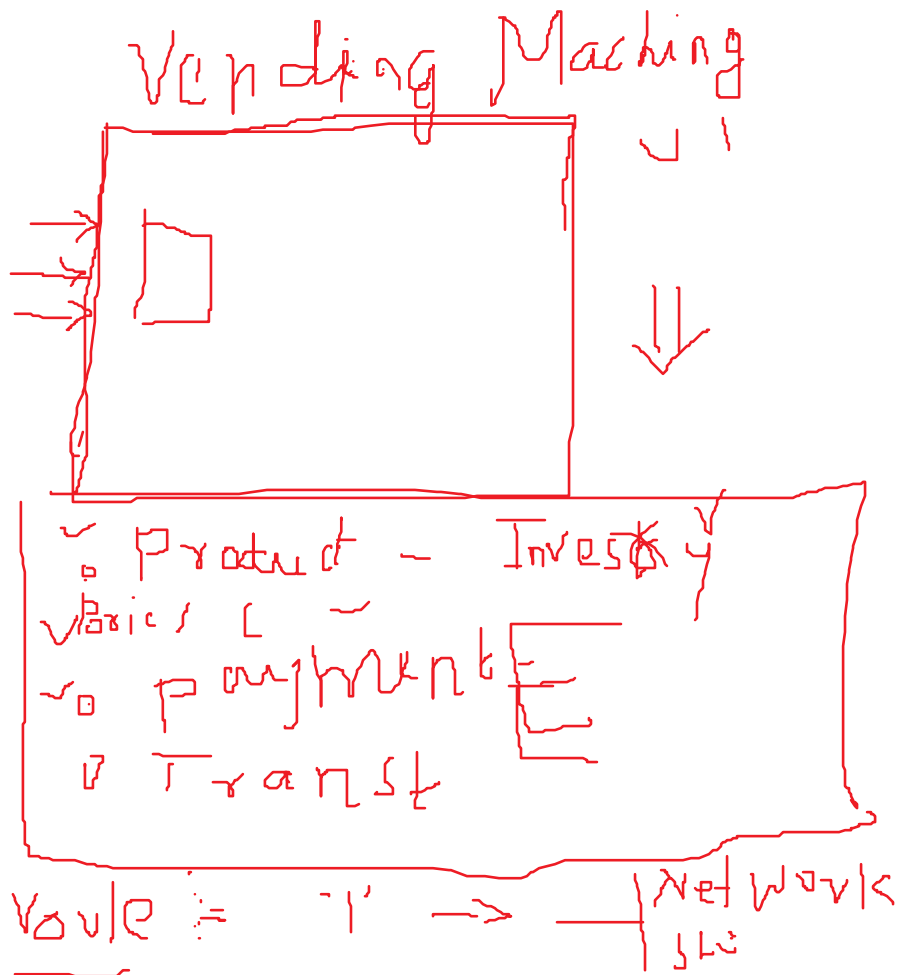
- Share
  - board_id
  - shared_with_user_id
  - created_at
```

```
Dynamodb
Board
- board_id (sort)
- owner_id (PK)
- name
- desc
- link_id_list (UUID)
- shared_with (SI)

Link
- link_id (PK)
- metadata
- url
- hash (SI)
```

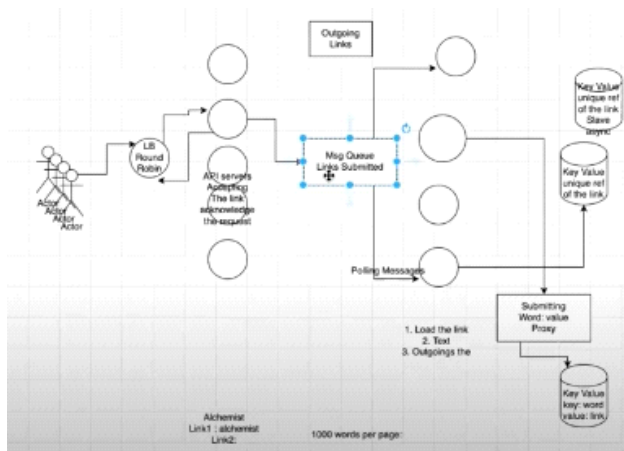
9. Vending machine

Wednesday, August 18, 2021 11:32 AM



10. Web Crawler

Monday, July 26, 2021 10:26 PM



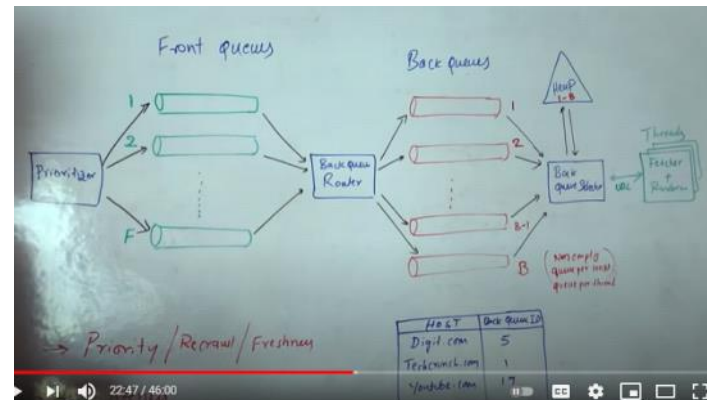
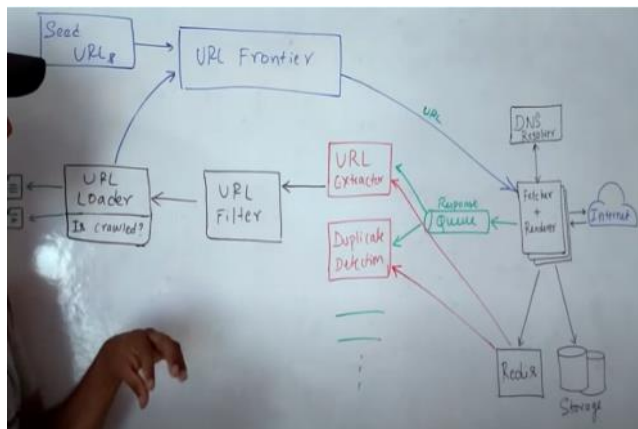
Web Crawlers

Actors
Admin
Public/Users

Functional
UI/Interface → Submitting of the URL Links which we crawl with higher priority
URLs → Outgoing links
Store / Persistence of the crawled (Search → Store data) // Search must be easy on store data

Non Functional:
Users:
URLs/No of pages: 10 million links / month submitted (UI)
Outgoing Links - 5 per URL average
Crawling once done is sufficient (PW)
Durable
High Available
Few Hours (crawl) → 2 phases (Loading Link / outgoing Links)

Assumption
Submission - acknowledge of accepting to be crawled
Email to the client/User - data is crawled
Client → Server/Processing → Data Persistence



11. Video Communication [ZOOM]

Friday, September 24, 2021 9:59 PM

Video Call

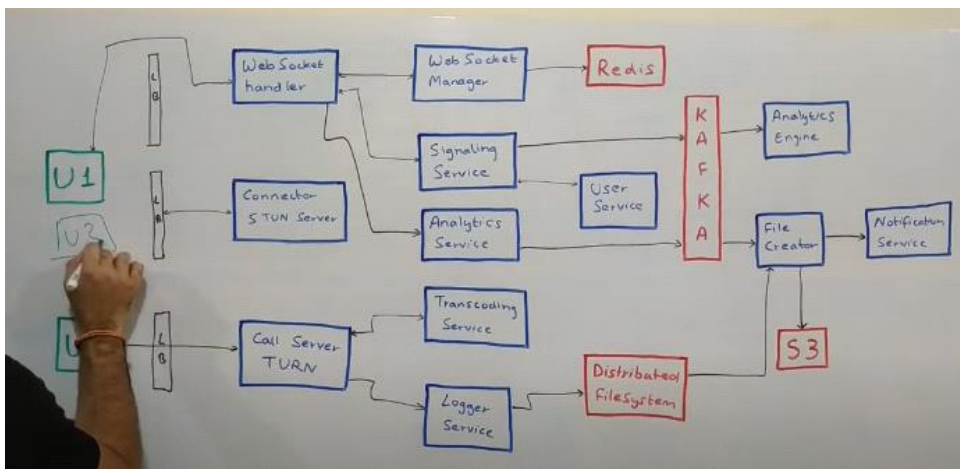
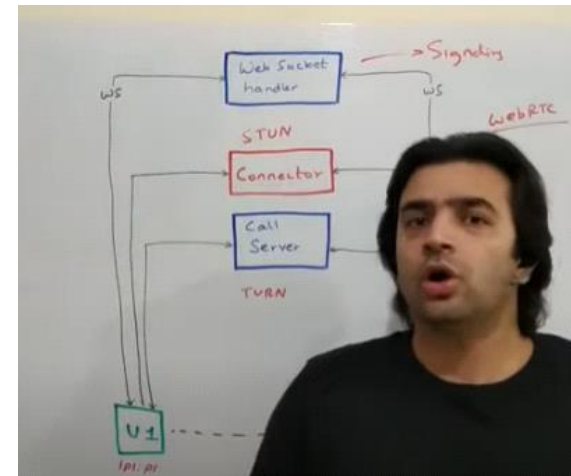
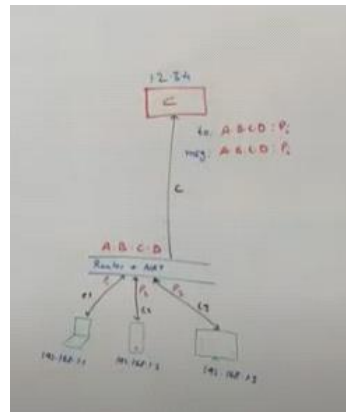
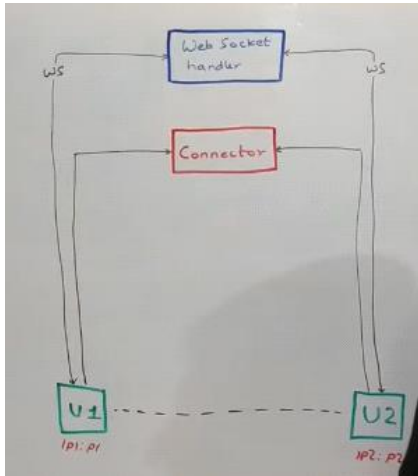
Tuesday, September 28, 2021 12:14 PM

FR

- 1 on 1 calls
- Group Calls
- Audio / Video / Screen Share
- Record

NFR

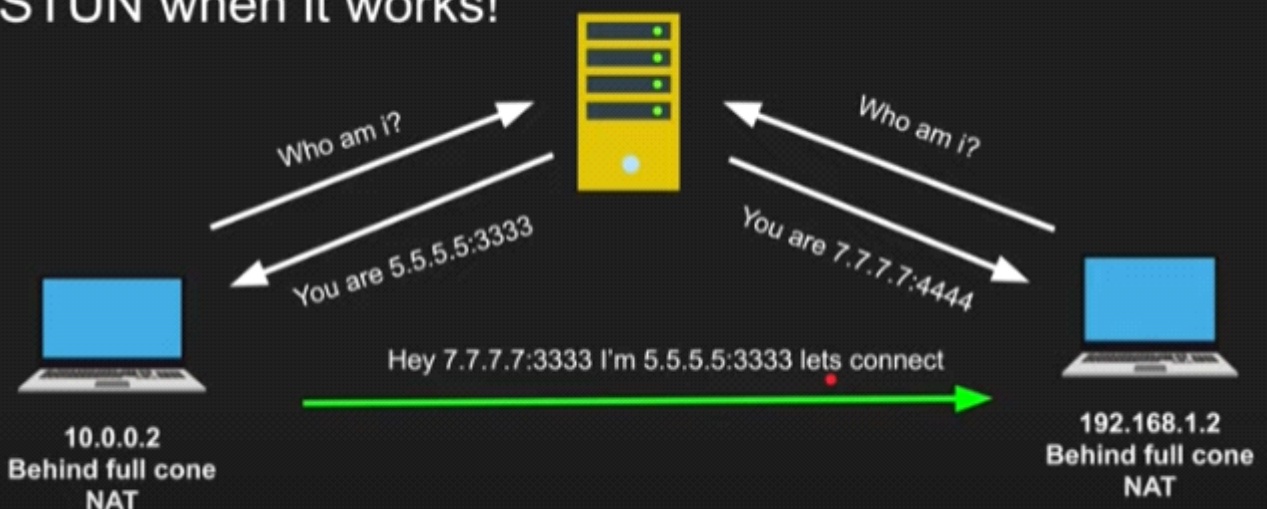
- Super Fast
- High Availability
- Data loss is ok



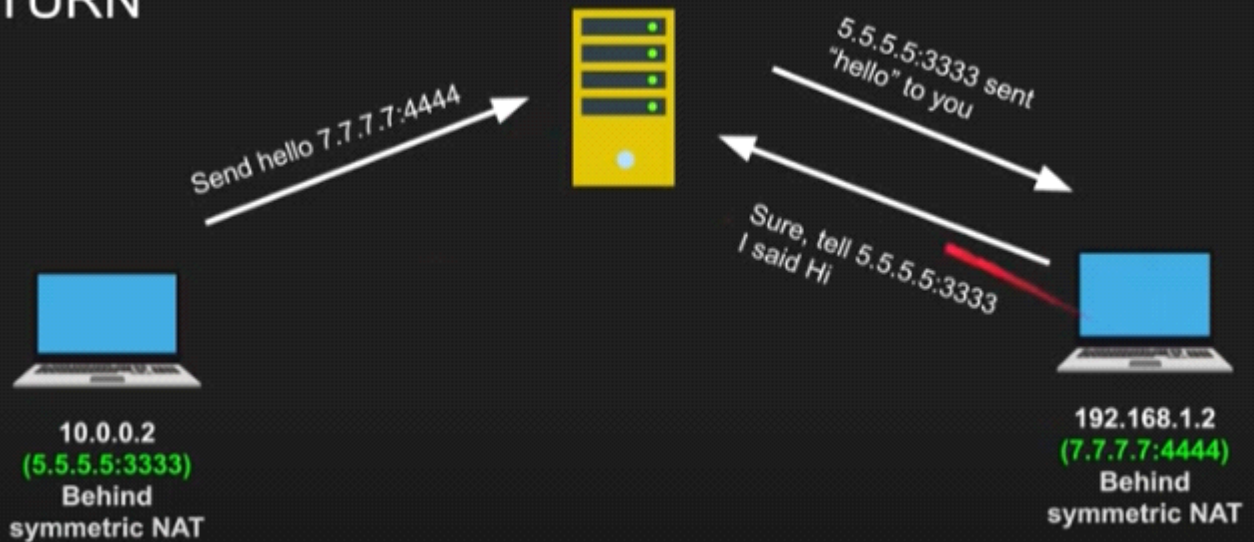
WebRTC Overview

- A wants to connect to B
- A finds out all possible ways the public can connect to it
- B finds out all possible ways the public can connect to it
- A and B signal this session information via other means
 - WhatsApp, QR, Tweet, WebSockets, HTTP Fetch..
- A connects to B via the most optimal path
- A & B also exchanges their supported media and security

STUN when it works!



TURN



ICE

- Interactive Connectivity Establishment
- ICE collects all available candidates (local IP addresses, reflexive addresses – STUN ones and relayed addresses – TURN ones)
- Called ice candidates
- All the collected addresses are then sent to the remote peer via SDP

SDP

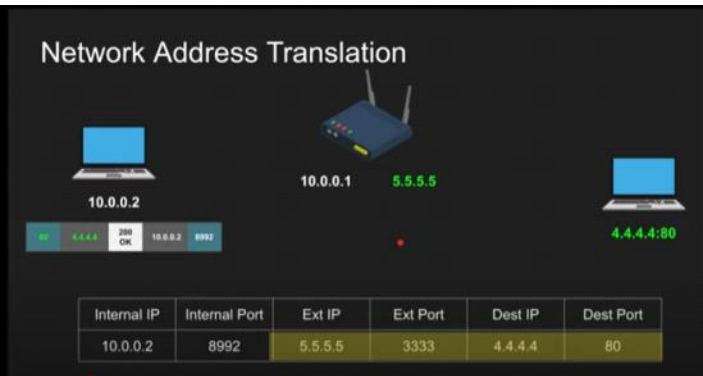
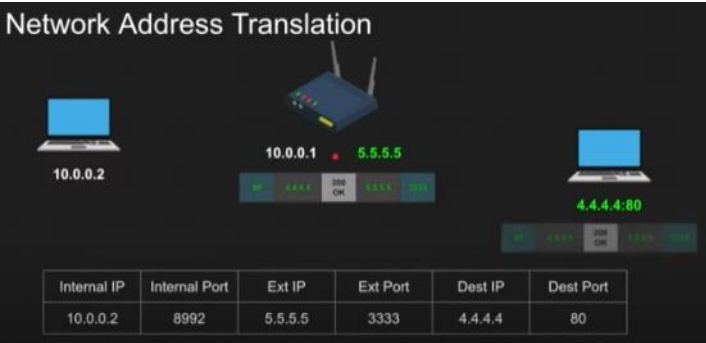
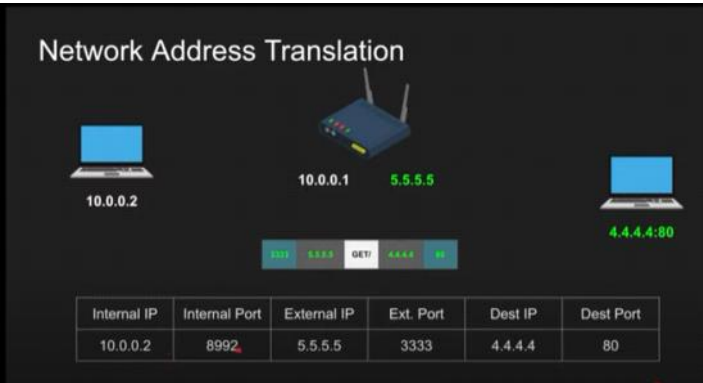
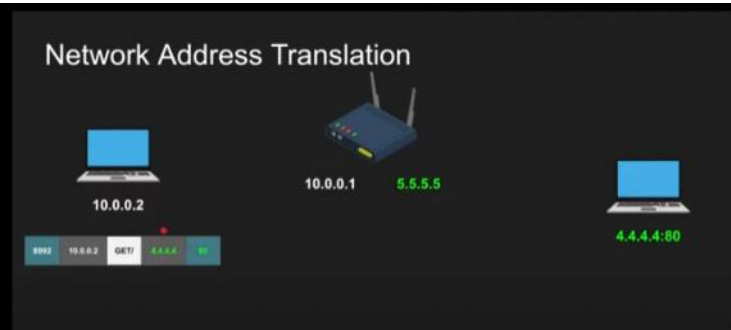
- Session Description Protocol
- A format that describes ice candidates, networking options, media options, security options and other stuff
- Not really a protocol its a format
- Most important concept in WebRTC
- The goal is to take the SDP generated by a user and send it "somehow" to the other party.

WebRTC Demystified

1. A wants to connect to B
2. A creates an "offer", it finds all ICE candidates, security options, audio/video options and generates SDP, the offer is basically the SDP
3. A signals the offer somehow to B (whatsapp)
4. B creates the "answer" after setting A's offer
5. B signals the "answer" to A
6. Connection is created

NAT

Tuesday, September 28, 2021 2:33 PM



ARP -> Address resolution Protocol

