Design a Caller Identification and Spam Application like Truecaller

Step-1:- Requirement Gathering

(a) Functional Requirements :-

① User Authentication :- Users can register and login.

② Calling Feature :- Users can call through Truecaller app.

③ Caller Identification :- If a caller is registered on Truecaller, then details of unknown call details are provided.

④ Spam detection & Blocking :- Can Detect fraud and spam calls and can block it automatically.

⑤ Number Search :- User can check the owner of a phone number if owner is a user on Truecaller i.e owner phone number is registered on this app.

⑥ Emergency Number Display :- Automatically show Police, Ambulance, fire brigade etc. numbers.

⑦ Call Log Maintenance :-

(a) Maintains the call record i.e received/sent history of call logs.

(b) Also, details of Phone and timing of Phone call (i.e when phone ring and for how many minutes user talked with each other and when did phone call stopped)

⑧ Contact Syncing :-
(a) Phone contacts of User will be synced to the database of Truecaller.
(b) If contact number is registered on Truecall-er then updated name and profile will appear on Truecaller App.

⑨ Profile Update :- Users can update their profile details on Truecaller app.

⑩ Notification Updates :- Users should receive notifica-tions if someone checks their profile (if privacy allows).

⑪ Call Recording feature :- Users can record in-coming & outgoing calls. Also, User can choose storage location (Cloud/local). Call recording must follow privacy laws.

⑫ VoIP Calling (Internet-Based Calling) :- Truecaller to Truecaller free calls using WIFI or mobile data.

(b) Non-Functional Requirements :-

① Scalability :- Our system should be highly scalable.

② Reliability :- Our System should be highly reliable.

③ Availability :- Our System should be highly available.

④ Secure :- Our system should be highly secure.

⑤ Latency Sensitive :- Our system should be latency sensitive. Latency sensitive means the latency time should be very low.

⑥ Maintainability & Updgradability :- Our system should be highly maintainable.

Step-2 :- Prioritisation

① Emergency Number Database Setup :-

Reason: (Fixed Schema + Small static Data) and independent of any service design.

Purpose: Emergency numbers like Police, Ambul-ance, Fire Brigade etc. should be stored and fetched.

② User Account service :-

Reason: It manages user profiles and contact syncing

Purpose: Stores user data and enable contact syncing.

③ Contact Syncing Service :-

Reason: Dependent on User Account Service and contact syncing is an immediate step after registration of user.

Purpose: Sync User's phonebook contacts with Truecaller's database.

④ User Authentication Service :-

Reason: It setups Secure Login System.

Purpose: It handles Password-based login, token generation, multi-factor authentication etc.

⑤ User Authorization Service :-

Reason: Defines each user access controls.

Purpose: Manages Normal vs Premium User access, profile privacy settings, spam reporting permissions, etc.

⑥ Spam Detection & Blocking Service :-

Reason: Filtering before connection is ensured before call so that calling feature service process valid calls.

Purpose: Identify and block spam numbers auto-matically.

⑦ Calling Feature Service :-

Reason: Dependencies are User Account service and User Authentication Service.

Purpose:- User-to-user call establishment are managed. Key functions are:-

(a) Call Initiation (Dialing)

(b) Call Connection & Routing

(c) Call Termination (End call)

(d) Call Logs Update (Received/Sent, Timestamp, Device Details)

(e) Technologies Consideration :-

→ VoIP (Voice Over IP)

→ PSTN (Public Switched Telephone Network) (switches or route standard mobile calls).

⑧ Call Recording Service :-

Reason: Dependencies over Calling feature service and Call recording service now can act as independent service after the calling feature service. And many service can be designed in parallel with this service.

Purpose: Users can record incoming and outgoing calls. Storage preferences like local device and cloud backups is also provided

⑨ Call Log Service :-

Reason: Dependencies over services like Calling Feature service. Parallel Design with Call Recording service since no dependency over call recording service.

Purpose: Maintains call log history like Call type, Timestamp, Device details, Duration etc.

8. Call Identification Service :-

Reason: Dependent over services including calling feature services and dependent on that service on which calling feature service relies.

Purpose: Identifies unknown callers, fetches caller name (if registered in database), provide spam detection, call labelling like Buisness, Personal, Spam, Real Time number lookup.

Designed in parallel with call recording service and call log service, since this service is not dependent on these services.

9. VoIP Calling Service

Reason:- Dependent on Call Identification service. Since information of call must be available when call comes which is provided by call Identification Service.

Purpose:- Internet-based calling (Truecaller to Truecaller VoIP calls)

10. Search Page Service :-

Reason:- On Searching, if an important information to be displayed by that searched number, we must provide it and this is possible only if VoIP calling service is designed so there is dependency over VoIP calling service.

Purpose:- User can search for unknown numbers

11. Profile Checking Service :-

Reason:- Dependencies over search page services as when user check his profile, we need to show him that xc number of people searched him and viewed his profile so

Purpose:- Users can check who viewed their profile.

(12) Notification Service & Analytics Service

Reason:- Dependent on all services

Purpose:- Notify users when someone checks their profiles, missed calls etc.

Step-3 :- Infrastructure Estimation

(a) Storage Requirements:-

Let, Total Users on Truecaller = 1 Billion

Daily Active User = 80% of Total Users

$= 0.8 \times 1$ Billion

$= 0.8$ Billion

$\Rightarrow$ DAU

Now, New Users added upto next year

$= 0.25$ Billion

(Rate of growth assumed to be 25%)

Total Users upto next year = 1.25 Billion

Storage Requirements of User will be decided by following factors:-

(i) Calls Logs Storage

(ii) Spam Reports & Blocking Data

(iii) Search History & Analytics

(iv) User Interaction & Notification Logs

(v) AI-based Spam detection Metadata.

(vi) User Profile Data

Contact Detail Stored in Truecaller for user after contact syncing

(a) Contact Name:- 50 Bytes

(b) Phone Number:- 20 Bytes

(c) Email (optional):- 50 Bytes

(d) Profile Photo (Base 64):- 5KB

(e) Sync Metadata (Timestamps, Hashes, etc.) :- 250KB

Let, 10 KB will be required to store each contact detail
Let, On an average each user has 750 contacts in it's phone number.
Then, Contact details data for contact syncing for each user = 10 KB x 750
$= 10 \times 10^3 \times 750$ Bytes
$= 7.5 \times 10^6$ Bytes
$= 7.5$ MB
$\approx$ 10 MB (Buffer 2.5 MB taken)

Overall Data for Contact Syncing
$= 10\text{MB} \times 1.25$ Billion
$= 10 \times 10^6 \times 10^9 \times 1.25$ Bytes
$= 12.5 \times 10^{15}$ Bytes
$= 12.5$ PB

Cassandra DB is used for Contact syncing storage now, compression to save space = 2.5 PB
if we want lossless decomposition / compression Algorithm for compression used:- LZ4.

For call logs:-
Let each Call Log take 250 Bytes of Data
We assumed that maximum 4000 call log entry can be stored in User's Truecaller app.
Single User Call log Estimate for data
$= 4000 \times 250$ Bytes
$= 1000000$ Bytes
$= 10^6$ Bytes
$= 1$ MB
For All Users, Data Estimate = 1 MB x 1.25 Billion
$= 10^6 \times 1.25 \times 10^9$ Bytes

Data Estimate = 1.25×10^{15} Bytes

= 1.25PB

DB used:- Cassandra

LZ4 Compression Algo is used, it provides 30-50% compression.

So, Data saved = 0.3×1.25 PB

= 0.375PB

So, Data required Storage = 1.25 - 0.375

= 1.250
- 0.375

= 0.875 PB

For Spam Reports & Blocking Data

Let Each user does 100 spam reports and 50 blocks

Let Each Spam & Block report take 100 bytes of data.

Single User Spam Reports & Blocking Data 100B next year = (100+50) x100 Bytes

= 150 x 100 Bytes

= 15 KB

Now, Total Data for Spam Reports & Blocking data upto next year = 1.25 B x 15 KB

= $1.25 \times 10^9 \times 15 \times 10^6$ Bytes

= $1.25 \times 15 \times 10^{15}$ Bytes

= 18.75 PB

Compression Algorithm used :- LZ4, Database:- Neo4J

Assumed Worst-case Compression (30% - 33%),

We can save 6.75PB of data.

Hence, Spam Reports & Blocking Data = 12 PB

For Search History & Analytics

Let, 1 search Query take 400 Bytes of data.

Also, Each user on an average do 10 queries
Then, Search Query DB for single user require data of 10×400 Bytes
$= 4KB$ per day

Total Data required for single user upto next year $= 365 \times 4\ KB$
$= 1460\ KB$
$= 1.46\ MB$

Now, Total Data upto Next year for Search Queries & Analytics $= 1.25$ Billion $\times 1.46 MB$
$= 1.46 \times 1.25 \times 10^9 \times 10^6$ Bytes
$= 1.825 \times 10^{15}$ Bytes
$= 1.825\ PB$

Database Used :- Elastic Search

Compression Algorithm :- LZ4

Data Reduced to 1.5PB (Worst Case Compression assumed)

Note :- Here we saved $(1.825 - 1.5)PB$ of data i.e $0.325PB$ of data is saved.

For User Interaction & Notification Logs :-

Daily Interactions per user (Assumed) = 50 per day
Notification logs per User (Assumed) = 10 per day
Now, Daily Interaction Data = 200 Bytes (Assumed)
Size per Notification log = 150 Bytes (Assumed)

Data Estimate upto next year :-

Daily Interactions data upto next year
$= 365 \times 50 \times 200$ Bytes
$= 365 \times 10\ KB$
$= 3650\ KB$
$= 3.65 MB$

Notification log data upto next year
$= 365 \times 10 \times 150$ Bytes
$= 547500$ Bytes
$= 0.5475 MB$

Total data upto next year for User Interaction and Notification logs = $1.25 \times 10^9 \times (3.65 + 0.5475) \times 10^6$ Bytes

$= 1.25 \times 10^9 \times (3.65 + 0.55) \times 10^6$ Bytes

$= 4.2 \times 1.25 \times 10^{15}$ Bytes

$= 5.25$ PB

Database used :- Cassandra
Compression used :- LZ4
On Worst Case Assume only 1.25 PB of data, it can remove.
So, data required for User interaction & Notification logs = 4 PB

AI-based spam detection Metadata :-

Spam detection logs event (each) data = 500 Bytes
Let each user does 5 action in total.
Then for each user spam detection log event data
$= 5 \times 500$ Bytes
$= 2.5$ KB per day

Now, Model Training data size per event be 1KB
Let number of spam message be 500M per day
Then, Per day spam message data for Model Training $= 500\text{M} \times 1\text{KB}$
$= 500 \times 10^6 \times 10^3$ Bytes
$= 0.5$ TB

Now, Meta Data storage size per event be 300 bytes
Each user do 10 logs which is responsible for this storage
Then, Metadata storage per user per day
$=$ 3KB (10 x 300 Bytes)

Total storage required
$= ((3 + 2.5) \times 365 \times 1.25 \times 10^9 \times 10^3 + 0.5 \times 365 \times 10^{12})$ Bytes
$= 2.6$ PB $+ 0.1825$ PB
$= 2.7825$ PB $\simeq 2.8$ PB

Database Chosen:- Cassandra + PostgreSQL + TimescaleDB + Elastic search

Compression used:- LZ4

Now, LZ4 is worst performance assumed to be 20% i.e 20% data is saved after compression

Then, data storage required = 2.8 - 0.2 × 2.8
= 2.8 - 0.56
= 2.24 PB

Cassandra for Spam detection logs.

PostgreSQL + Timescale DB for ML Model Predictions

Metadata storage used Elastic search

(Reasons) Fast search and Retrieval

Efficient Time-Series storage

Fast writes, scalable

User Profile :-

Avg size data per user = 1.4 KB

Total storage = 1.25 B × 1.4 KB
= $1.25 \times 10^5 \times 10^3 \times 1.4$ Bytes
= 1.75 TB

Image will be stored in File Based System

Avg size = 50 KB per user

Total Requirement storage upto next year = data for Distributed File
= $50 \times 10^3 \times 1.25 \times 10^5$
= 62.5×10^{12} Bytes
= 62.5 TB

Compression Algorithm:- LZ4 (Again 20% Reduction assumed)

Amazon S3 distributed file storage selected.

Image Links & Metadata stored in PostgreSQL

Average size of image Link = 500 Bytes

Then Total Image Link data = $1.25 \times 10^9 \times 500$
= 0.625 TB

After Compression :-

File Data = 53TB & Image Link data = 0.53TB

User Profile Data 1.4KB Breakdown :-

Basic Info → 300 Bytes → Cassandra / PostgreSQL

Preferences & Settings → 200 Bytes → PostgreSQL

Activity History → 500 Bytes → Elastic Search

Spam Reports & Blocks → 400 Bytes → Cassandra

LZ4 Compression used (worst case 20%)

So, User Profile data = $\frac{1.75}{1.2} \approx 1.46$ PB

Now,

Overall Data Estimation of Truecaller

$= 1.46\,PB + 0.53\,TB + 0.053\,PB + 2.24\,PB + 4\,PB + 1.5\,PB + 12\,PB + 0.875\,PB + 12.5\,PB$

$\simeq 35\,PB$

(B.) Traffic Estimation :-

(i) Daily Active User Estimation :-

80% of users are assumed to active on truecaller

So, Daily Active User = 0.8 Billion (Already Estimated)

All users are not active at the same time. Requests will be distributed over 24 hours and also the requests rate will be distributed over 24 hours and also the request rate is not uniform it will be high at certain hours and low at certain hours.

(ii) QPS Request per second (RPS) :-

Assumed 10% of the users are active at the same time

Peak concurrent User = 10% of DAU

$= 0.1 \times 0.8$ Billion

$= 0.08$ Billion

Assumed Each user on an average give 10 requests per Minute

Peak Load = 10×0.08 Billion /min

$= 0.8$ Billion/min

$= \frac{0.8}{60}$ Billion/sec

$= \frac{800}{60} \times 10^6$ Requests per second

$\Rightarrow$ Peak Load = 13.33 M Requests per second

Now, Average Load = $\frac{0.8 \text{ Billion} \times 10}{24 \times 60 \times 60}$

$= \frac{8000}{24 \times 3600} \times 10^6$ Requests per second

$= 92.5$ K Requests per second

(b) Read vs Write Heavy

Read Heavy Application as 80% Reads and 20% Writes

(ii) Compute Power Estimation:-

(a) BackEnd servers Estimation (Application Layer)

Assumed 500 Requests are handled by each server.

For peak Load, Required Number of servers

$= \frac{13.33 \text{ M}}{500}$

$= \frac{13.33 \times 10^3 \times 10^3}{500}$

$= 26.66 \times 1000$

$= 26660$ servers

$\simeq 27000$ servers.

(b) Database Scaling (Read & Write) Request Handling

Read Requests = $0.9 \times 13.33\text{M} \simeq 12\text{M}$

Due to heavy Read system, we need database replica

Assumed 1 DB Instance ~ 50K reads handle

Required Read Replicas Servers

$$= \frac{12000\,M}{50K}$$

$$= \frac{12000\,M}{50K}$$

$$= 240$$

$$\approx 240 \text{ servers}$$

Hence, Read Replica Servers = 240

For writes, strong consistency is needed

hence we use horizontal scaling

1 strong Write DB instance ~ 10K Write handle easily

Required Write Instances $= \frac{0.2 \times 13.33\,M}{10K}$

$$= \frac{2.666 \times 10^3\,K}{10K}$$

$$= \frac{2666}{10}$$

$$= 266.6 \approx 267$$

So, Required DB Write instances = 267 servers.

(c) Caching Layer (Redis / Memcached)

To reduce DB load, we serve 80% read requests by cache alone

Cache hit Ratio = 80%

Total cacheable requests = $0.8 \times 13.33M \times 0.9$

$$\simeq 9.6M \text{ requests}$$

Each Redis/Memcached Node can handle 100K RPS assumed

So, Total Cache Nodes Required = 9.6M / 100K

Total cache Nodes = $\frac{9.6\,M}{100K}$

$= \frac{9.6 \times 10^3\ M}{100K}$

$= \frac{9600}{100}$

⇒ Total Cache Node = 96

So, Total 96 Redis Nodes are required

(iii) Throughput Calculation :-

Already computed in DAU estimation i.e Peak Load.

(iv) Bandwidth Estimation :-

Assume Average Request size = 5KB

Total Bandwidth Estimation = $13.33 \times 5KB/s \times 10^6$

$= 67$ GB/sec

If used compressed algorithm / techniques, we can reduce it by 50% i.e approx 34 GB/sec.

CDN & caching will reduce the load on main servers and we can achieve Bandwidth estimation upto 7 GB/second.

(v) Cache Estimation :-

70% of total data is used as the storage of Cache to increase system performance.

Caching is used in :-

(a) User Profile Photo :-

Overall Cache Storage required = 0.7×62.5 TB

$= 43.75$ TB

CDN caching is used here

(b) Search History & Analytics :-

Overall Cache Storage Required = 0.7×1.5 PB

$= 1.05$ PB

(c) Spam Reports & Blocking :-

Overall Cache Storage Estimation = 0.7 x 12 PB
= 8.4 PB

(d) AI Spam Detection Metadata :-

Overall Cache Storage Required = 0.7 x 2.8 PB
= 1.96 PB

Total Cache Data Required = 1.96 PB + 8.4 PB + 1.05 PB + 43.75 TB
= 11.45375
≈ 12 PB

(vi) Replication Factor :-

For Normal DB :-

Replication factor chosen :- 3

For caching :-

Replication factor chosen :- 2

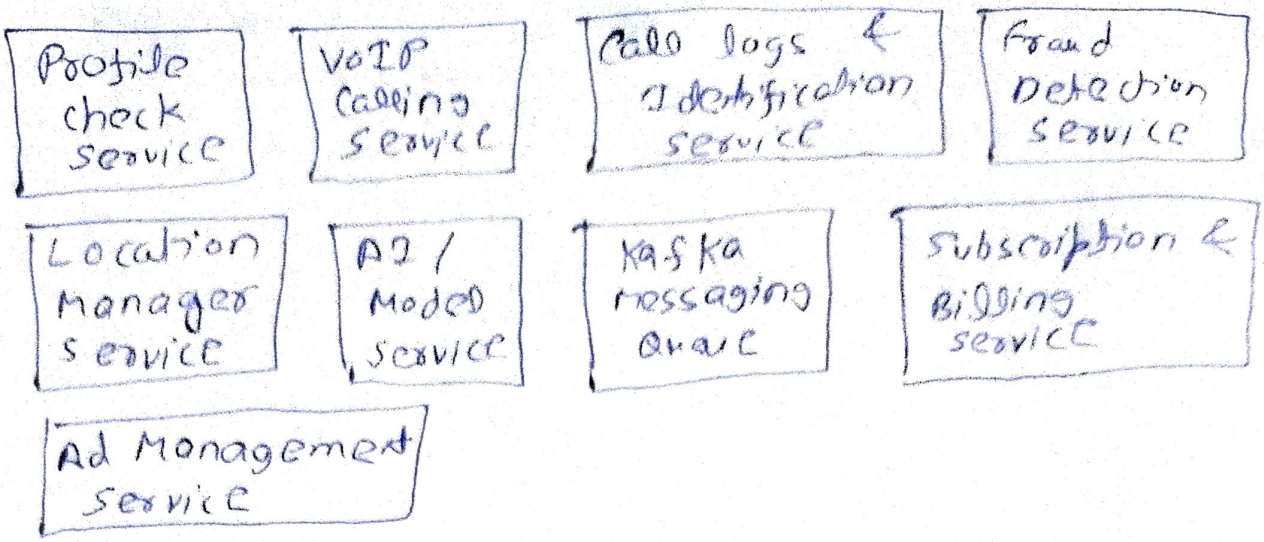
This boosts high availability of the system

Overall Storage required = 2 x 12 PB + 3 x 35 PB
= 24 PB + 105 PB
= 129 PB
≈ 130 PB

Step-5 :- Component Identification :-

Services that comes in my mind are given :-

- User Account service
- User Authorization Service
- User Authentication Service
- Load Balancer
- Zookeeper
- Request Handling service
- Search Page Service
- Notification service
- Customer support service
- Contact Management Service
- Call Recording Service
- Spam Detection & Blocking service



Role of Each Service:-

User Account Service:- Manage User profile Data, subscription details etc.

User Authorization service:- Give access to the users like free user / premium User. Free users are not given the information of the person who viewed his profile. Also premium users are not shown ads so Ad Management service is not given right to provide ads to premium users.

User Authentication Service:- Authenticate Users :- valid or invalid, responsible for login / register.

Load Balancer:- Distribute load in order to provide high availability.

Zookeeper:- keep health check-up of server and replace it with other servers if server connot handle load properly, used in service where there is high load.

Request Handling Service :- Responsible to handle all requests of users after authentication is successful.

Profile Check Service:- inform users about who checked his/her profile and give access / deny access to check a premium user profile based on the fact that checker is free user or premium user.

Contact Management Service :- Responsible for contact syncing, favourite contact, blocked contacts management etc. Users can edit name, add phone number i.e edit contact is also allowed in this service, add contact, remove contacts, all these feature are provided.

Spam detection & Blocking service :- Inform users about spam on phone call and help them to not become prey of spam by Automatically blocking the number. This service has the access to block a phone call by blocking that number on call and alert the user about that phone number.

VoIP calling service :- Truecaller to Truecaller call is provided over Internet / WIFI instead of Normal call.

Ad Management Service :- Provide sponsors a platform to advertise their product in form of ads when a free user is using app, or going to call someone.

Subscription & Billing Service :- Provide Premium features to user like unlimited calls, unlimited profile searches, ad-free experience & many more.

KAFKA Messaging Queue :- Used where Asynchronously we need to provide tasks to services in parallel.

Call Logs & Identification Service :- Provide details of unknown call like where the person is calling from, what is his/her name if he/she is registered on Truecaller and maintain call log. Example in our system maximum 4000 call logs can be stored for a user. But a person how many times called him/her and how many times they called user this data is maintained by this service.

Location Manager :- Manage Location of all User from where they are calling, this is managed by this service.

AI/Model service:— Based on data of Spam Detection & Block service, Location Manager, Request Handling service, search page service and various services, this service uses ML's LLM models that make certain services to perform very well like training fraud detection service that alerts user about fraud through Truecaller app or in form of Notification to the user via Notification service

Customer support service:— If a user is facing some problem, this service connects with MTS team of Truecaller and helps them to solve their problem.

Fraud Detection Service:— Alert users about fraud.

Notification service:— Gives information about certain things like call logs (missed calls), profile checks, recommendations etc. in form of Notifications.

Search Page service:— By this service, user can search unknown phone call number and get some details that it is fraud, spam etc.

Call Recording service:— Provide features like auto record the calls and manual record and responsible for saving the recordings

Call Recording service Estimated Data (Part of Infrastructure Estimation i.e Database Estimation)

Let Auto Recording Users = 12%.

Manual Recording Users = 8%.

Average Call Duration be 10min (Assumed)

Assumption:-

Audio Format → WAV (Uncompressed PCM Audio

Sample Rate → 44.1KHz (Standard for High-Quality Audio)

Bit Depth → 16bit (Standard for most recordings)

Channels → Mono (Single channel since call recordings are usually mono).

Size = Sample Rate × Bit depth × Channels × Duration

= 44100 × 16 × 1 × 600 Bits

Size (in Bytes) = $\frac{44100 \times 16 \times 600}{8 \times 1}$

= 52.92MB

Size (in Bytes) ≈ 53MB

Database used will be Distributed file storage i.e Amazon Simple storage service (S3)

Compression Algorithm :- OPUS

Worst Compression assumed :- 50%.

So, Size in Byte for single call recording

= 0.5 × 53MB

= 26.5MB

Also, Link must be saved in call log to fetch this call recording from S3 storage usually Link take data of about 300 Bytes

We assumed that our system will maintain call log of 4000 calls only (old call data will be erased automatically)

Now, Data Estimation for Auto Recording VS CRS :-

= (26.5003) MB × 4000 × 0.12 × 0.8 Billion

= (26.5003) × 10^6 × 4000 × 0.12 × 0.8 × 10^9

= 10176.1152 × 10^{15} Bytes

= 10.176 × 10^{18} Bytes ≈ 11 EB

Since, impractical to store 11EB of data, we can reduce number of call recording capacity we store last 10 call recordings at max
This Reduce Storage Requirement to 27.5PB

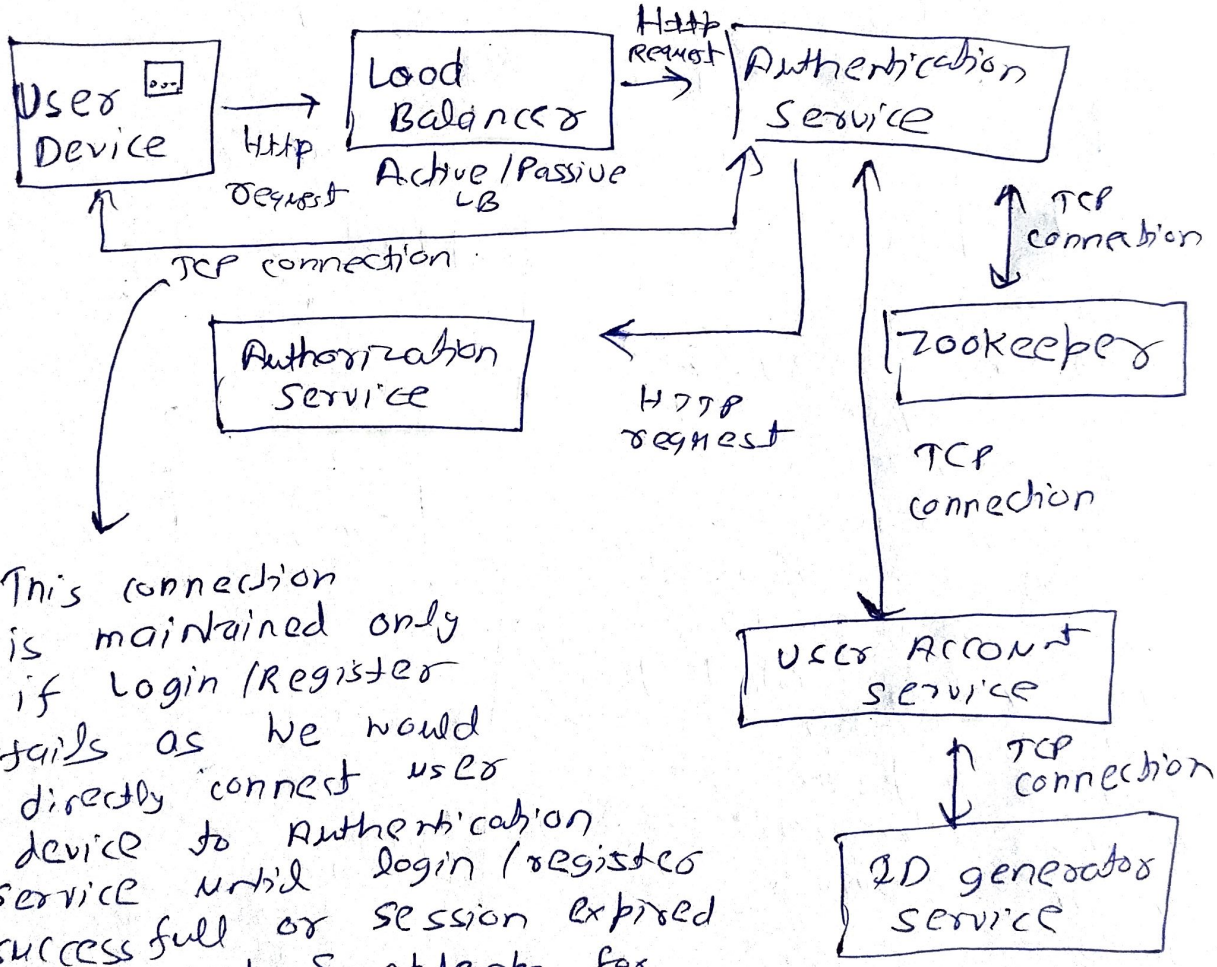
Total Storage Requirement = 130PB + 27.5PB
≈ 158 PB

For manual users Recording Let 2PB of data is needed.

Total Storage Requirement = 160PB

No Replication is done here i.e Replication factor is 1 for Call recording data

HOW service interact with each other

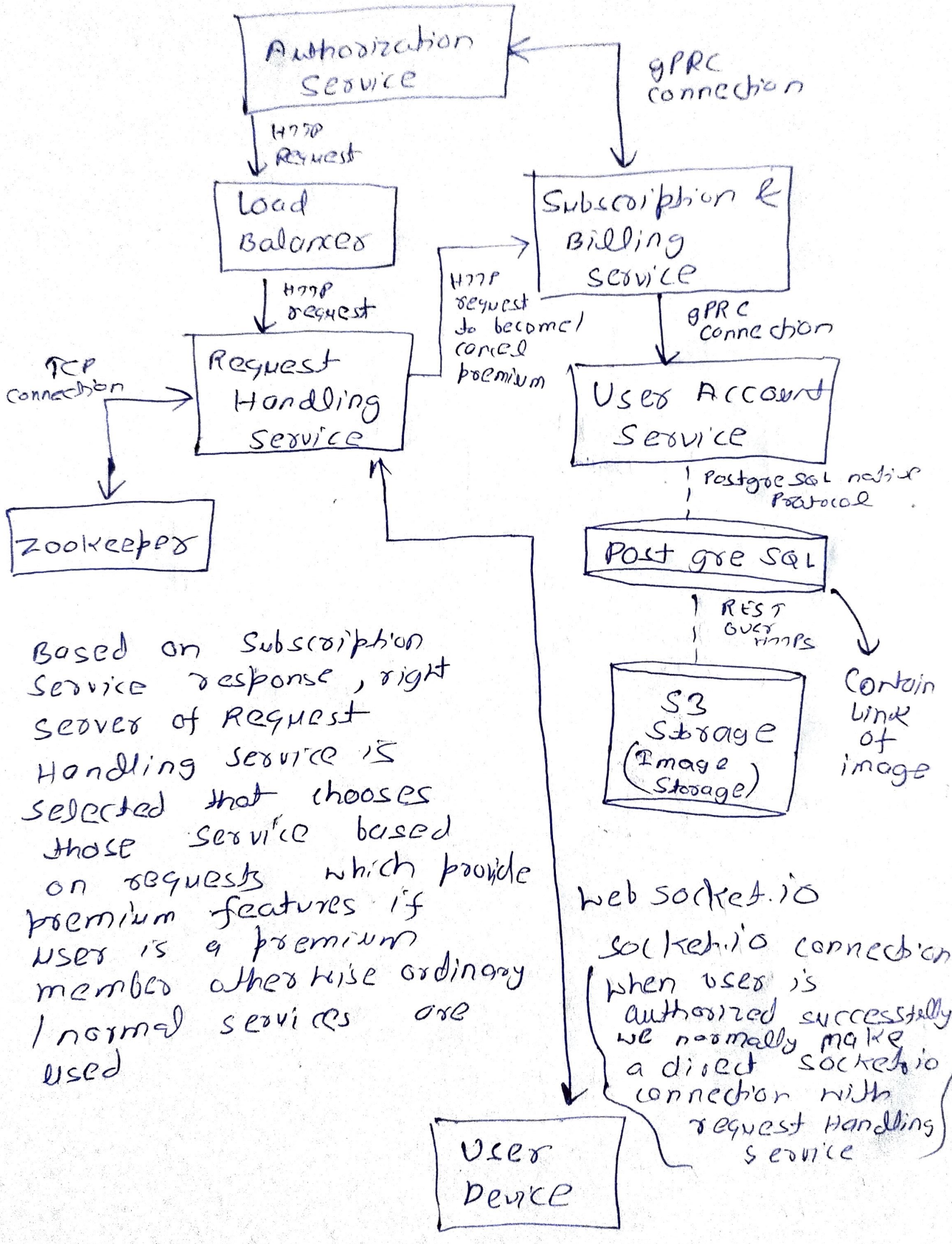


This connection is maintained only if Login/Register fails as we would directly connect user device to Authentication service until login/register successfull or session expired like 3 to 5 attempts for login/register then TCP connection will broke out and again user will connect to Truecaller via HTTP request.

Noted Points:-

All services are distributed in nature

How services interact with each other especially request Handling service



Based on Subscription service response, right server of Request Handling service is selected that chooses those service based on requests which provide premium features if user is a premium member otherwise ordinary / normal services are used

Services that maintains User call -

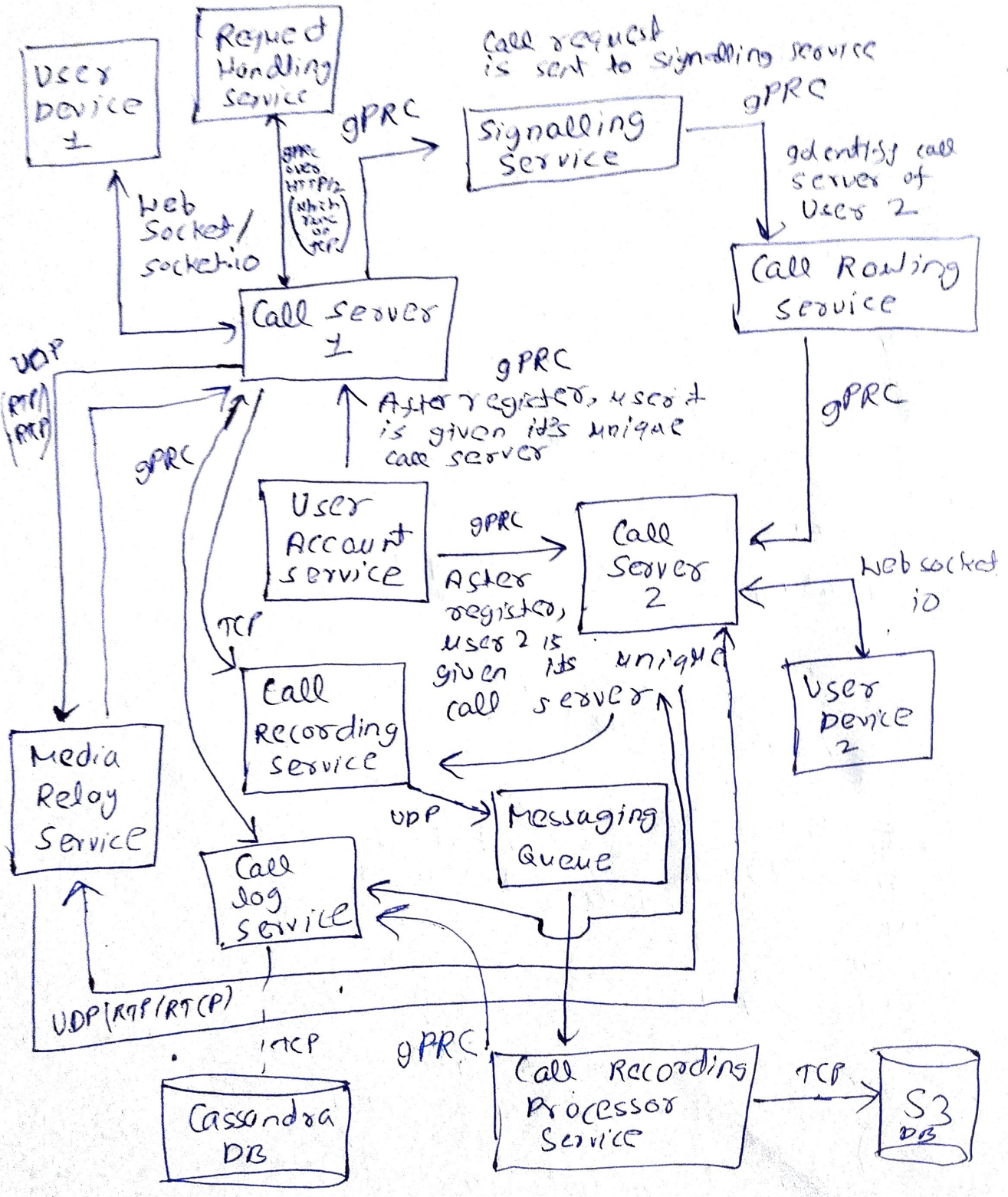
Call servers

Signalling service

Call Routing service

Media Relay Service

When a call is established we make direct web socket connection of user device to call servers.



Media Relay Service:- Ensures that User1 words are heared or sent to User2 and User2 words are heared or sent to User1. This is maintained by this service. This is done in order to provide meaningful communication, ensures that problem user voice is not repeated to him, he receives only his caller's voice.

Call Log Service:- Conneded to call server, so whenever call is established it keeps a track on call server in order to store history of call details like, Timestamp of call initiation & termination, timings of call, audio recording link (if recording is activated) However this link is provided by Call Recording processor service as it decides the location where the audio need to be stored

Signalling Service:- On call requests, call server sends request to signalling service this initiates the call by calling call routing service, this service search in which call server this caller is available and connect to that call server

Call Routing service:- Responsible to find correct call server and if call server is alive (VoIP calling requires internet so check online status of call servers) give/send request to call server 2 and go out of work. If status is offline it sends the request and request is pushed to messaging queue which is taken by notification service and inform the user device about the missed call.

Call Recording Processor service:- It convert the audio into required file since it receives packets and it need to combine those packets, apply some compression and prepare an audio file, send to distributed file system like S3 and decides location where it needs to be stored and provide location to

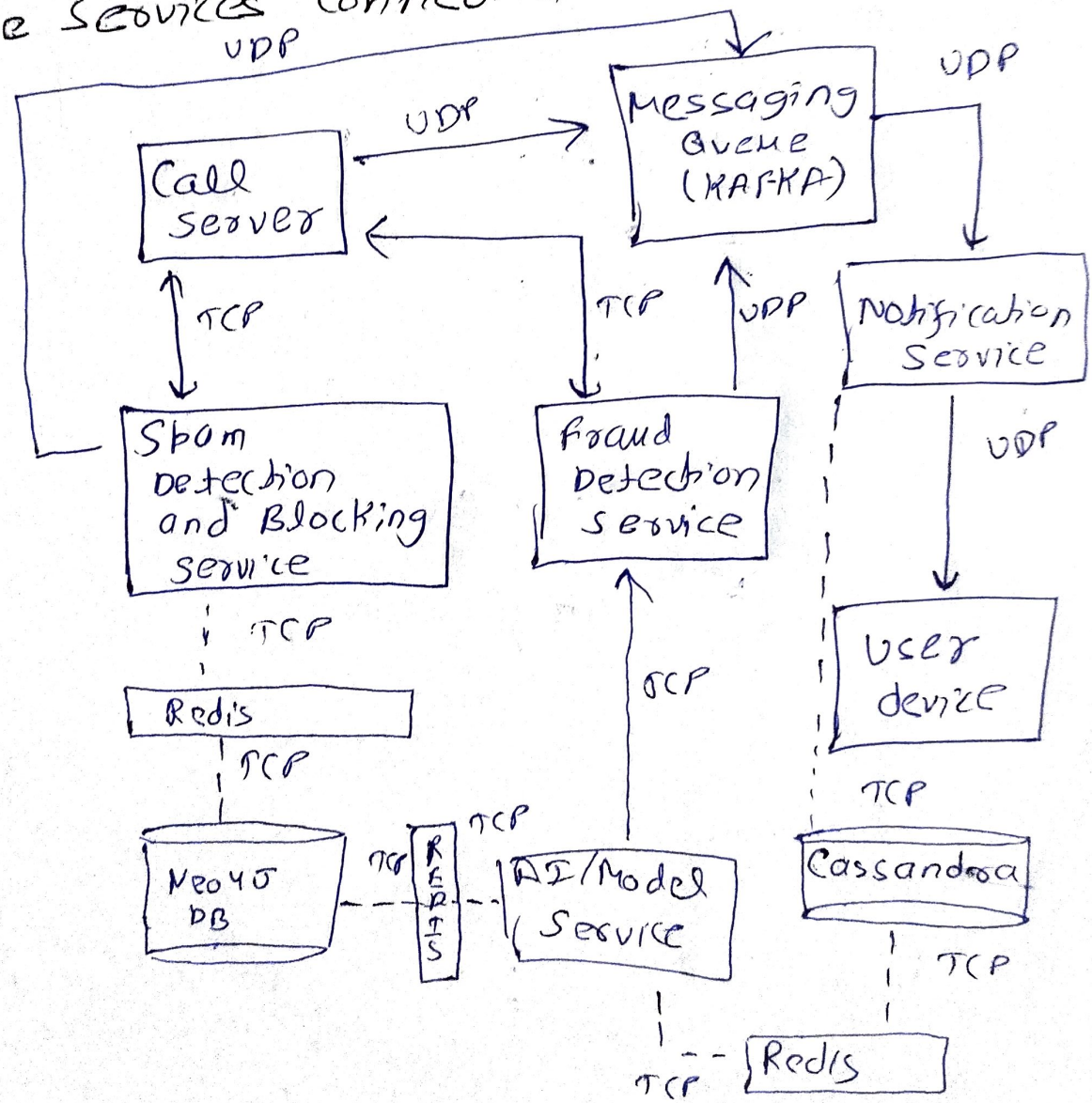
Call log service in form of link.

Important Highlights

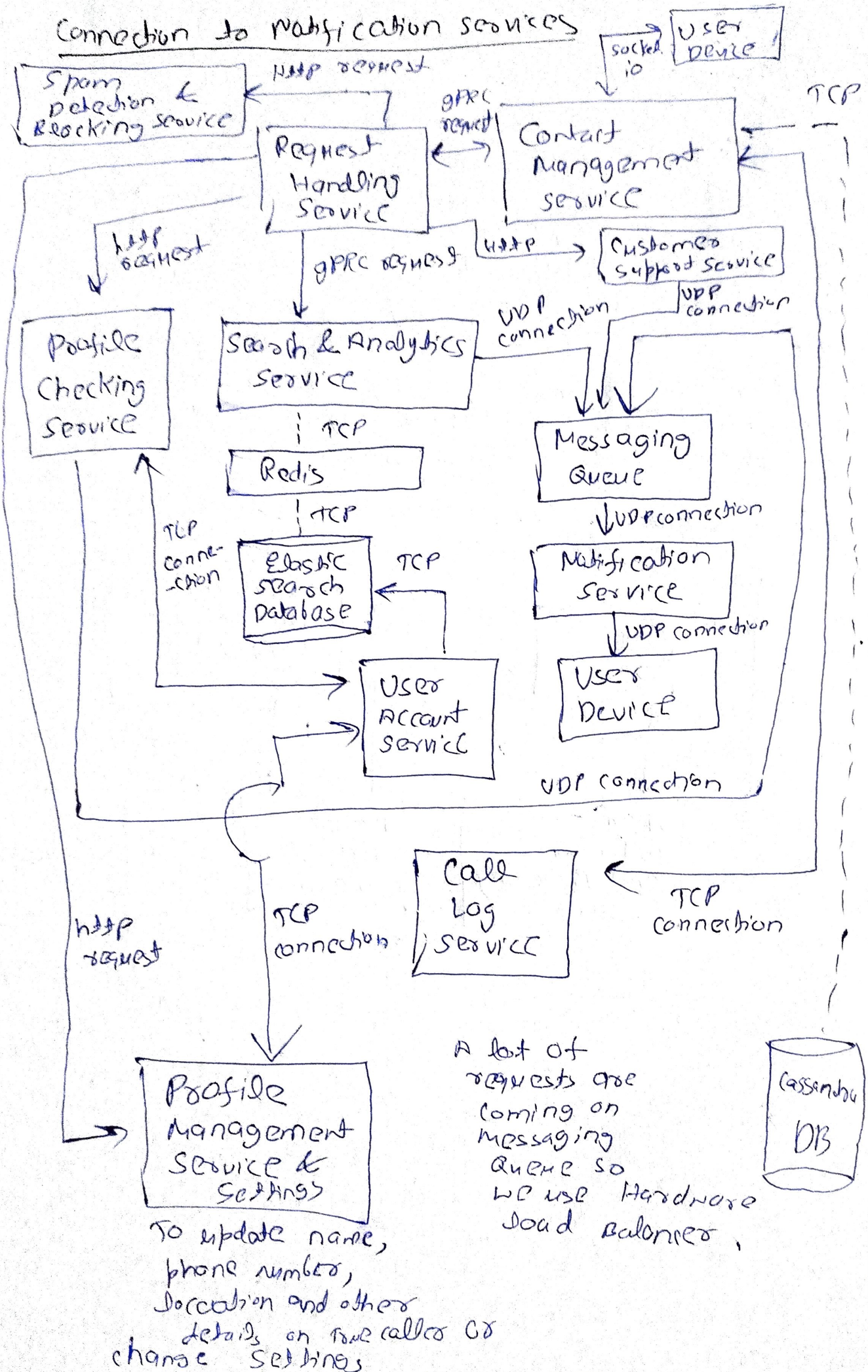
Sticky Load Balancing is used in the connection of request Handling service to call servers. Since, call servers are fixed for each users at the time of registration call servers are assigned to them. This algorithm is applied in software Load Balancing.

Note:- On Diagrams, if load Balancer is shown explicitly this means we used hardware Load Balancing otherwise by default we are using software load Balancing.

More services connection :-



Connection to Notification services



Sequence of Load Balancing Algorithm :-

Round Robin
↓
Least connection
↓
Weighted Round Robin

↓ Request per second increase.

Stateless System are preferred :-

(a) Scalability :- Easily horizontally scalable

(b) Microservices Architecture :- Best, because service works independent of each other

(c) Load Balancing :- can send request to another system as no session dependency will be there

(d) Failure Handling :- If a server failed, request go to another server without any data loss.

Service Oriented Architecture is also used :-
(AI/ML service) & (Spam detection & Blocking Service) both used some DB i.e Neo4J or share neo4J DB with each other. Also User Account service and search and Analytics Service share DBS i.e Elastic search as well as Notification service & AI/Model service share same DB i.e Cassandra

Microservices is used a lot as compare to SOA :-
Most of the services (approx 90%) follow microservices Architecture in this design.

Deciding Distributed Database (Master-Master & Master-Slave or where we can use the combination of both) Strategies :-

Master-Master :- User Account service, Subscription (Write-Heavy) and Billing service, Fraud detection Service, Call Recording service, profile Management & settings service, Contact Management service, Notification service, Call Server, Load Balancer,

Authentication service, ID Generator service, Call Routing service, Media Relay service, Request Handling service (partial made in Master-Master Architecture)

Master-slave:- Spam Detection & Blocking service, (Read-Heavy) Profile checking service, Call log service, Search & Analytics service, API/ Model service, Zookeeper, Authorization service, Signalling service, Request Handling Service (Partial made in the Master-slave architecture).

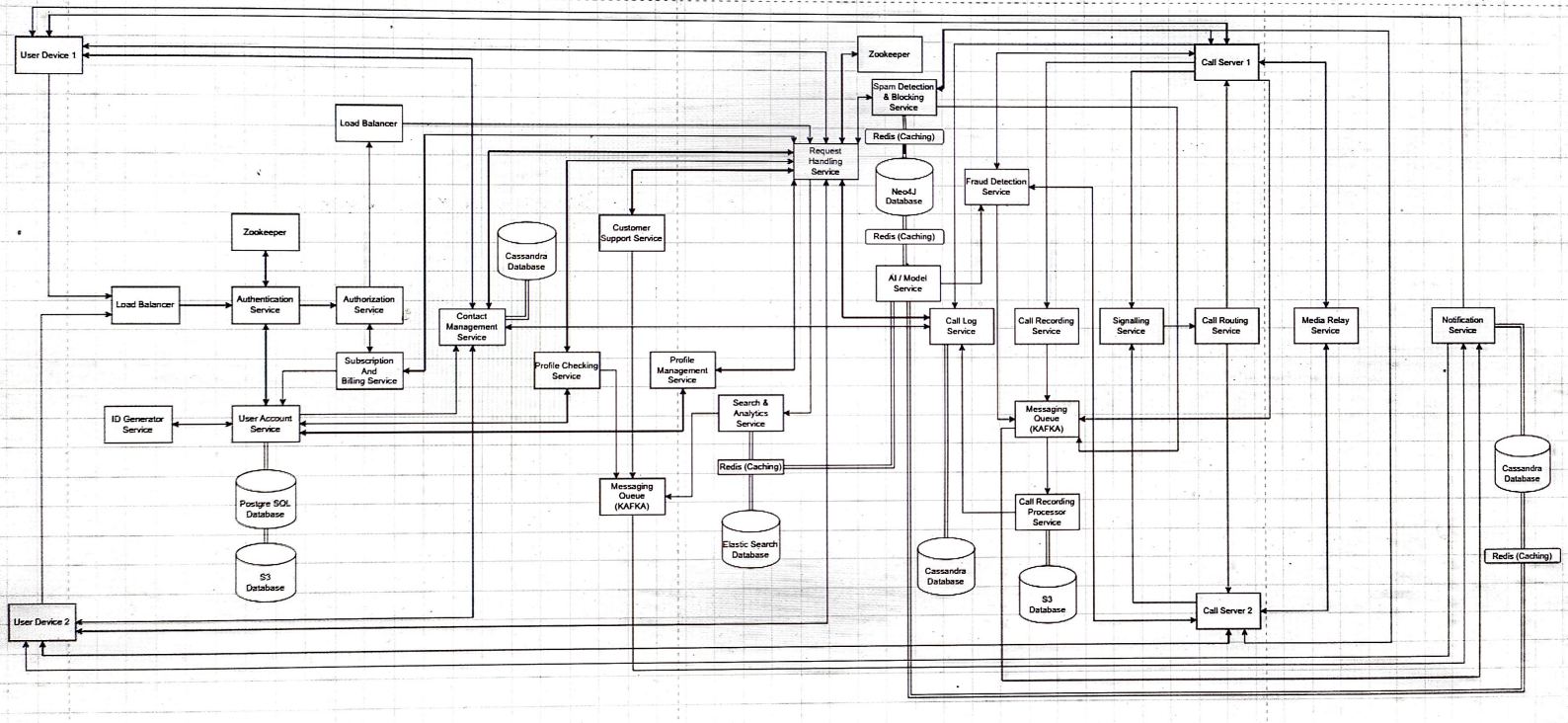
Important Point:-
Sharding is used along with range-based partitioning in databases

Communication Protocol and L4/L7 Load Balancing :-

Service 1	Service 2	Communication Protocol	Load Balancer L4/L7
Call server	Messaging Queue (KAFKA)	UDP	L4
Fraud Detection service	Messaging Queue (KAFKA)	UDP	L4
Spam detection & Blocking service	Messaging Queue (KAFKA)	UDP	L4
search & Analytics service	Messaging Queue (KAFKA)	UDP	L4
Customer support service	Messaging Queue (KAFKA)	UDP	L4
Profile checking service	Messaging Queue (KAFKA)	UDP	L4
Call Recording service	Messaging Queue (KAFKA)	UDP	L4
Call Recording Processor service	Messaging Queue (KAFKA)	UDP	L4

Service 1	Service 2	Connection	Load Balancing L4/L7
Call Server	Fraud Detection service	TCP	L4
All Services	Database / Redis	TCP	L4
Redis	Database	TCP	L4
Spam Detection & Blocking Service	Call Server	TCP	L4
Notification Service	User Device	UDP	L4
AI / Model Service	Fraud Detection service	TCP	L4
Request Handling service	Spam Detection & Blocking service	HTTP	L7
Request Handling service	Contact Management service	gPRC	L7
Request Handling service	Profile Checking service	HTTP	L7
Request Handling service	Profile Management & Settings service	HTTP	L7
Request Handling service	Search & Analytics service	gPRC	L7
Request Handling service	Zookeeper	TCP	L4
Request Handling service	Customer Support service	HTTP	L7
Request Handling service	Call Server	gPRC over HTTP/2 (which runs on TCP)	L7
Request Handling service	User Device	Socket.io (WebSocket)	L7
Request Handling service	Subscription & Billing service	HTTP	L7
Request Handling service	Load Balancer (Hardware LB)	HTTP	L7
User Device	Load Balancer (Hardware LB)	HTTP	L7
User Device	Authentication Service	TCP	L4
User Device	Contact Management service	Socket.io	L7
User Device	Call Server	Socket.io	L7

Service 1	Service 2	Connection	Load Balancing L4/L7
Profile checking service	User Account service	TCP	L4
User Account service	Profile Management service	TCP	L4
Contact Management service	Call log service	TCP	L4
User Account service	Call Server	gPRC	L7
Call server	Signalling service	gPRC	L7
Signalling service	Call Routing service	gPRC	L7
Call Routing service	Call server	gPRC	L7
Call Server	Media Relay service	UDP	L4
Call Server	Call log service	gPRC	L7
Call Server	Call Recording service	TCP	L4
Call Recording Processor service	Call log service	gPRC	L7
Load Balancer (Hardware LB)	Authentication service	HTTP	L7
Authentication service	Authorization service	HTTP	L7
Authentication service	Zookeeper	TCP	L4
Authentication service	User Account service	TCP	L4
User Account service	ID Generator Service	TCP	L4
Subscription and Billing service	User Account service	gPRC connection	L7


User Device 1
Load Balancer
Zookeeper
Load Balancer
Authentication Service
Authorization Service
Subscription And Billing Service
ID Generator Service
User Account Service
Postgre SQL Database
S3 Database
User Device 2
Cassandra Database
Contact Management Service
Customer Support Service
Profile Checking Service
Profile Management Service
Search & Analytics Service
Redis (Caching)
Elastic Search Database
Messaging Queue (KAFKA)
Zookeeper
Spam Detection & Blocking Service
Redis (Caching)
Request Handling Service
Neo4J Database
Redis (Caching)
AI / Model Service
Fraud Detection Service
Call Log Service
Cassandra Database
Call Server 1
Call Recording Service
Signalling Service
Call Routing Service
Media Relay Service
Messaging Queue (KAFKA)
Call Recording Processor Service
S3 Database
Call Server 2
Notification Service
Cassandra Database
Redis (Caching)