Rider Request Ride from A to B - as Private or Shared

Driver Creates Ride from A to B - Private / Shared - Seats and Time, Offers

Rider Creates - Get Matching Driver based on criteria

Send Rider Requests to Matched Drivers

DRIVER(s) Give Price

Rider can Negotiate(once), Accept, Decline

Rider can Negotiate(once), Accept, Decline

Give Best Optimised Pickup Order for Driver

Share Live Location and Precise Pickup time for each riders

Estimated Metrics

Everyone In Canada Takes 2 Rider as Rider Per week

2 * 7 * 4(AVG Seats per ride) = 60 Rides Per Driver

Canada Population 40 Million = 40 Million/ 60 = 600000 At a Time

Map Drivers and Riders = 600000 * 2 = 1.2 Million To Map

Mapping by Lat, Lng (Uber H4) = 1.2 *(64 bit lng, lat, UID) = 0.4 GB = 0.5 GB

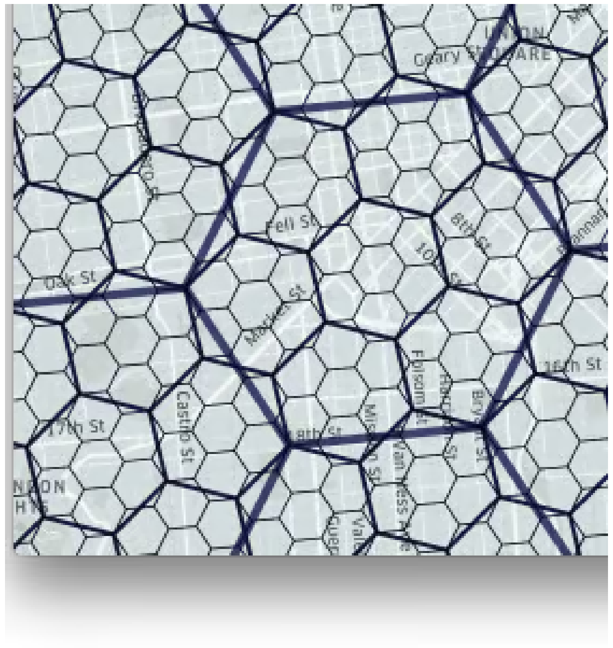
600000 * 2 = 1,200,000 / week = 1,200,000 / 604800 (7days * 24hrs * 60 Minutes * 60 seconds) = 2 Requests / second - Node.js can handle(express and fastify)= 15,000 Requests

(1 * 3 / min req) * 600000 = 1.8 Million Requests - to give price
50% Gives Price -> 1 Million Request / expires 1 Min - Partition - Socket Server

Driver Creates Ride

Driver Ride Details is stored in DB

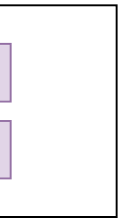
Store Ride Source (Lat, Lng), stops, Destination in Memory (Uber H3)

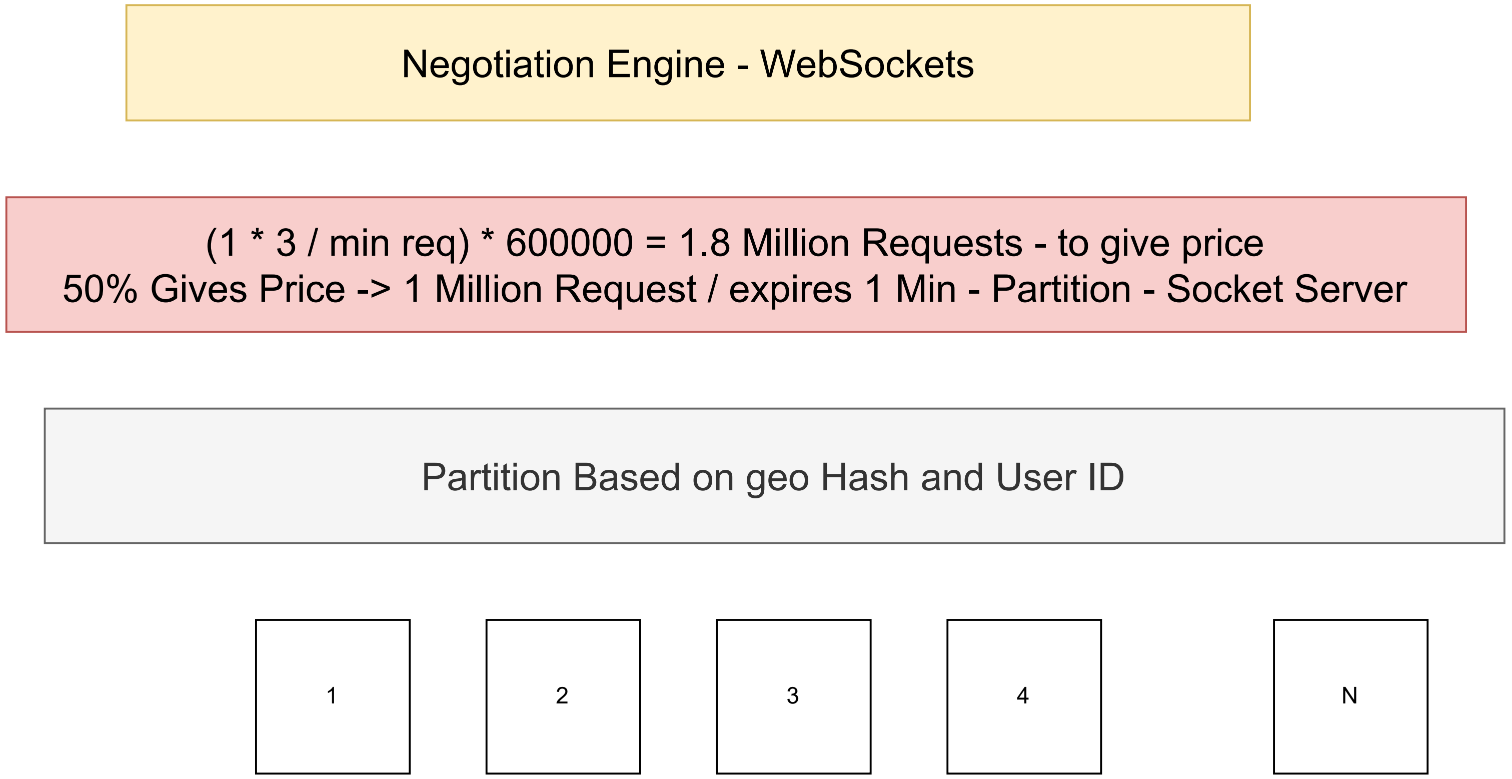


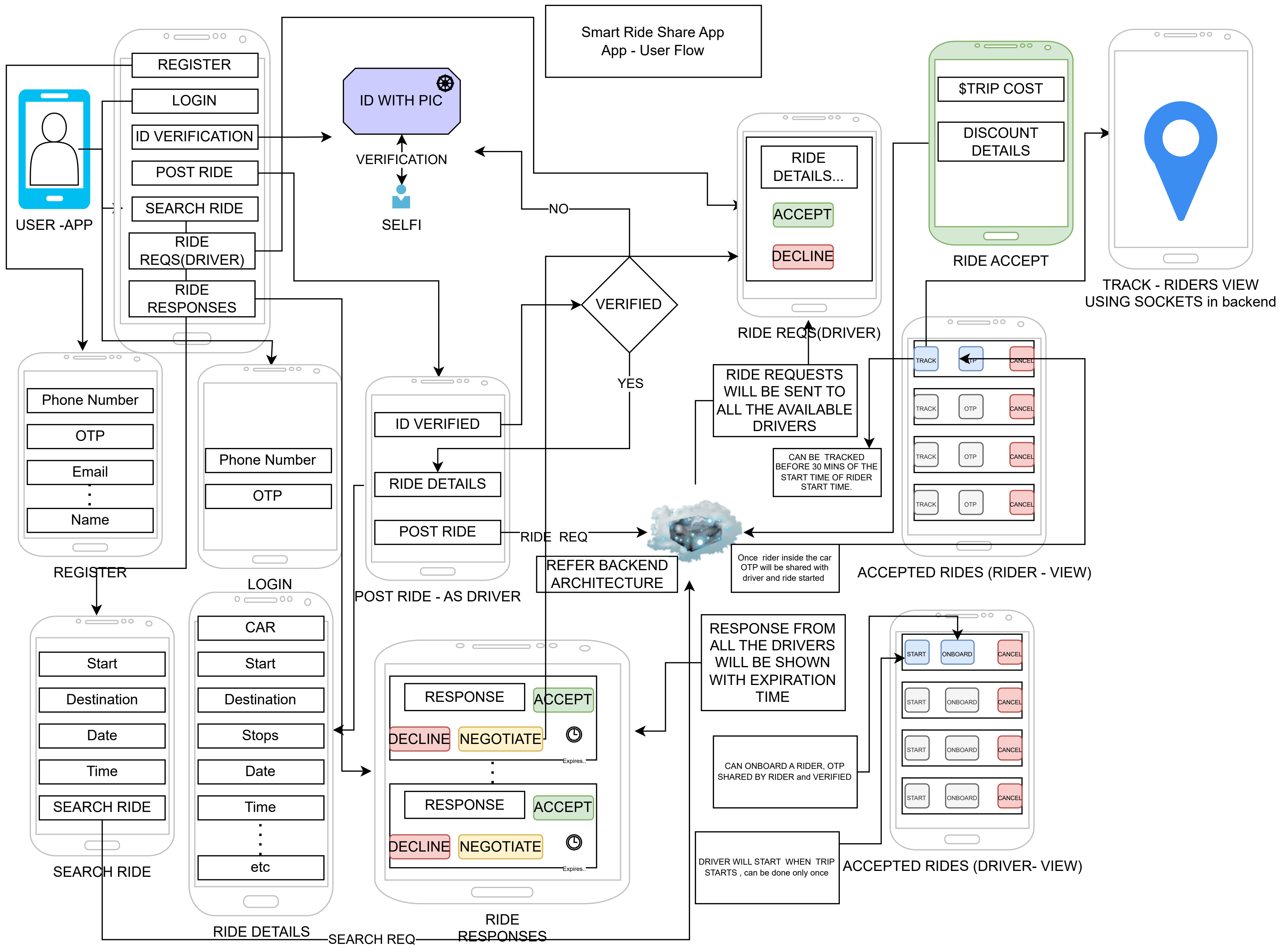
Rider Creates Ride

Rider Ride Details Stored in DB

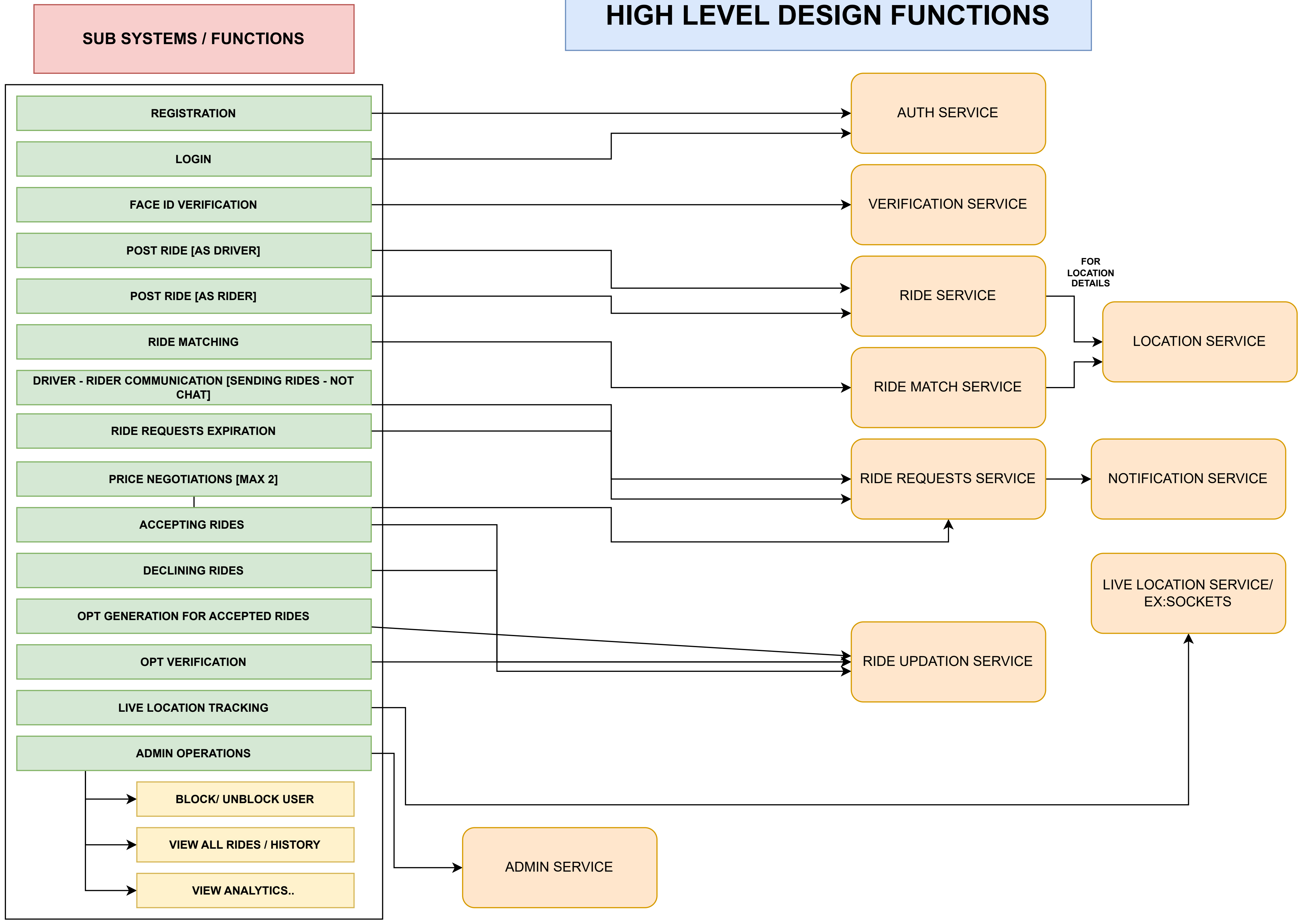
Put Ride into Slots and Do Find Nearby



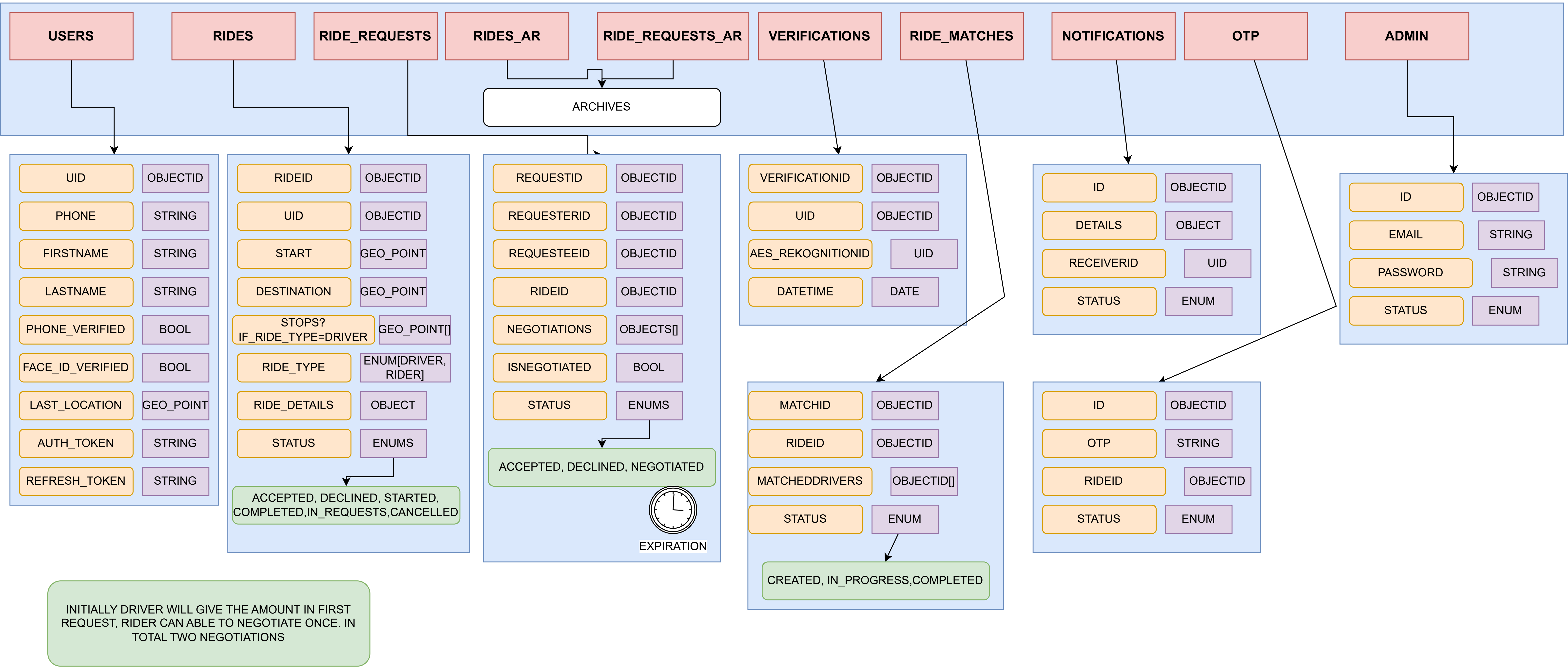

Negotiation Engine - WebSockets
(1 * 3 / min req) * 600000 = 1.8 Million Requests - to give price
50% Gives Price -> 1 Million Request / expires 1 Min - Partition - Socket Server
Partition Based on geo Hash and User ID
1
2
3
4
N


Smart Ride Share App
App - User Flow
USER -APP
REGISTER
LOGIN
ID VERIFICATION
POST RIDE
SEARCH RIDE
RIDE REQS(DRIVER)
RIDE RESPONSES
ID WITH PIC
VERIFICATION
SELFI
NO
VERIFIED
YES
Phone Number
OTP
Email
Name
REGISTER
Phone Number
OTP
LOGIN
ID VERIFIED
RIDE DETAILS
POST RIDE
POST RIDE - AS DRIVER
RIDE REQ
REFER BACKEND ARCHITECTURE
RIDE DETAILS...
ACCEPT
DECLINE
RIDE REQS(DRIVER)
RIDE REQUESTS WILL BE SENT TO ALL THE AVAILABLE DRIVERS
CAN BE TRACKED BEFORE 30 MINS OF THE START TIME OF RIDER START TIME.
$TRIP COST
DISCOUNT DETAILS
RIDE ACCEPT
TRACK - RIDERS VIEW USING SOCKETS in backend
TRACK
OTP
CANCEL
Once rider inside the car OTP will be shared with driver and ride started
ACCEPTED RIDES (RIDER - VIEW)
Start
Destination
Date
Time
SEARCH RIDE
SEARCH RIDE
CAR
Start
Destination
Stops
Date
Time
etc
RIDE DETAILS
RESPONSE
ACCEPT
DECLINE
NEGOTIATE
Expires..
RIDE RESPONSES
SEARCH REQ
RESPONSE FROM ALL THE DRIVERS WILL BE SHOWN WITH EXPIRATION TIME
CAN ONBOARD A RIDER, OTP SHARED BY RIDER and VERIFIED
DRIVER WILL START WHEN TRIP STARTS , can be done only once
START
ONBOARD
CANCEL
ACCEPTED RIDES (DRIVER- VIEW)

HIGH LEVEL DESIGN FUNCTIONS



DATABASE HIGH LEVEL DESIGN - MongoDB



UID	OBJECTID
PHONE	STRING
FIRSTNAME	STRING
LASTNAME	STRING
PHONE_VERIFIED	BOOL
FACE_ID_VERIFIED	BOOL
LAST_LOCATION	GEO_POINT
AUTH_TOKEN	STRING
REFRESH_TOKEN	STRING

RIDEID	OBJECTID
UID	OBJECTID
START	GEO_POINT
DESTINATION	GEO_POINT
STOPS? IF_RIDE_TYPE=DRIVER	GEO_POINT[]
RIDE_TYPE	ENUM[DRIVER, RIDER]
RIDE_DETAILS	OBJECT
STATUS	ENUMS

ACCEPTED, DECLINED, STARTED, COMPLETED,IN_REQUESTS,CANCELLED

REQUESTID	OBJECTID
REQUESTERID	OBJECTID
REQUESTEEID	OBJECTID
RIDEID	OBJECTID
NEGOTIATIONS	OBJECTS[]
ISNEGOTIATED	BOOL
STATUS	ENUMS

ACCEPTED, DECLINED, NEGOTIATED

VERIFICATIONID	OBJECTID
UID	OBJECTID
AES_REKOGNITIONID	UID
DATETIME	DATE

MATCHID	OBJECTID
RIDEID	OBJECTID
MATCHEDDRIVERS	OBJECTID[]
STATUS	ENUM

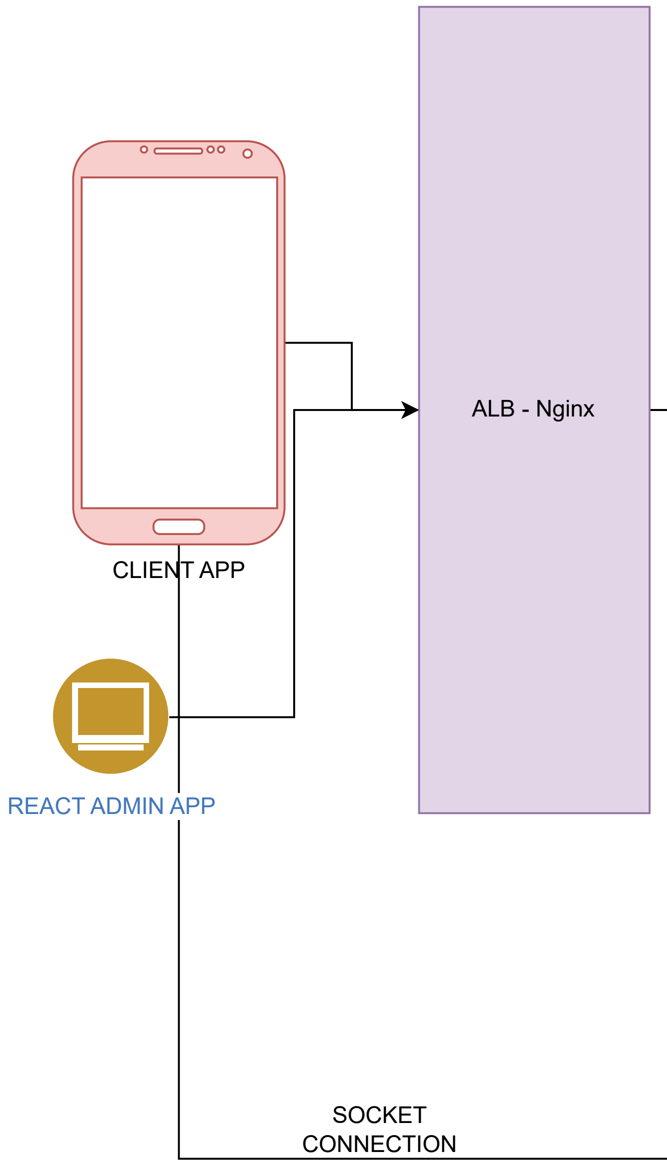
CREATED, IN_PROGRESS,COMPLETED

ID	OBJECTID
DETAILS	OBJECT
RECEIVERID	UID
STATUS	ENUM

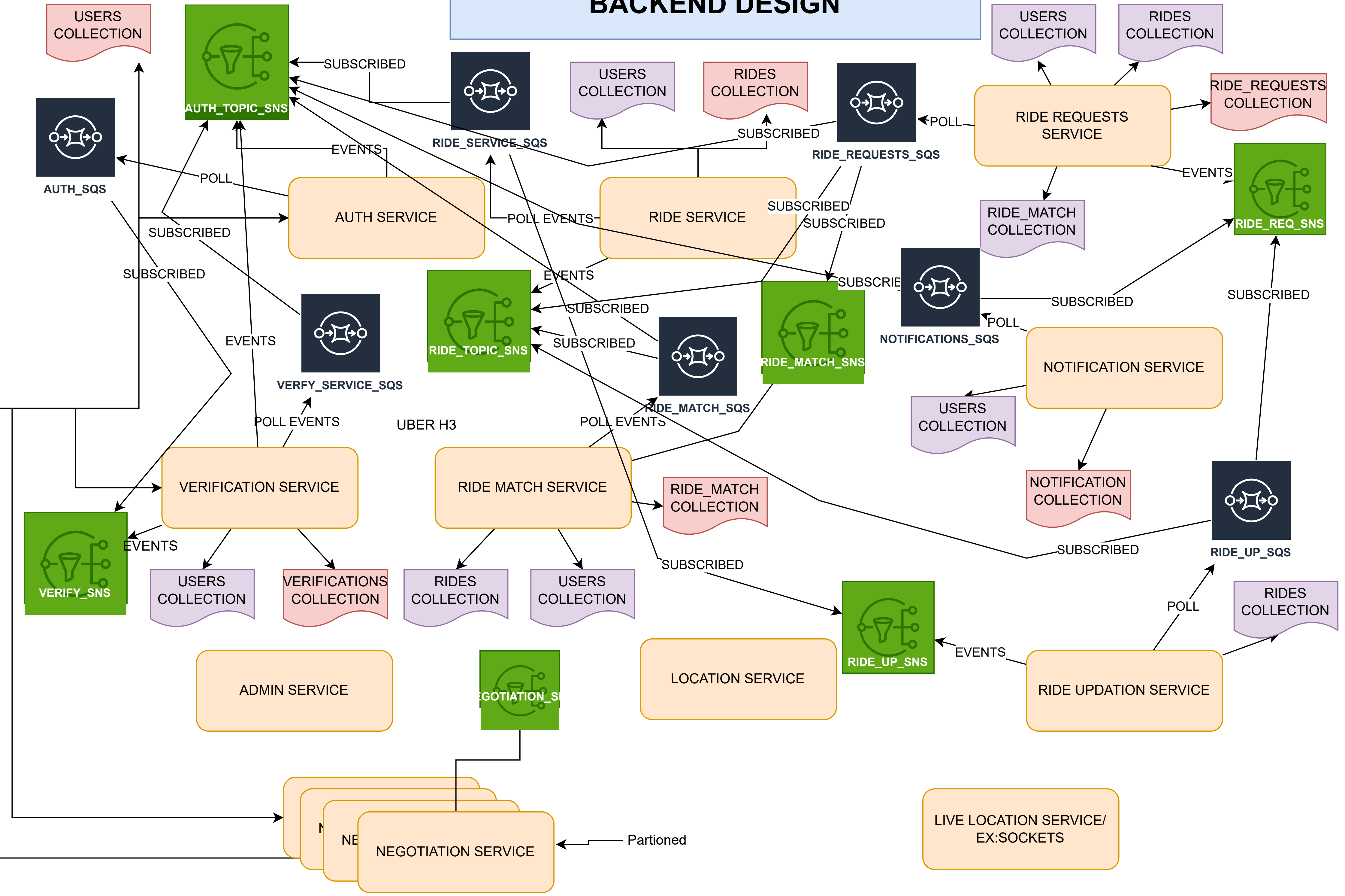
ID	OBJECTID
OTP	STRING
RIDEID	OBJECTID
STATUS	ENUM

ID	OBJECTID
EMAIL	STRING
PASSWORD	STRING
STATUS	ENUM

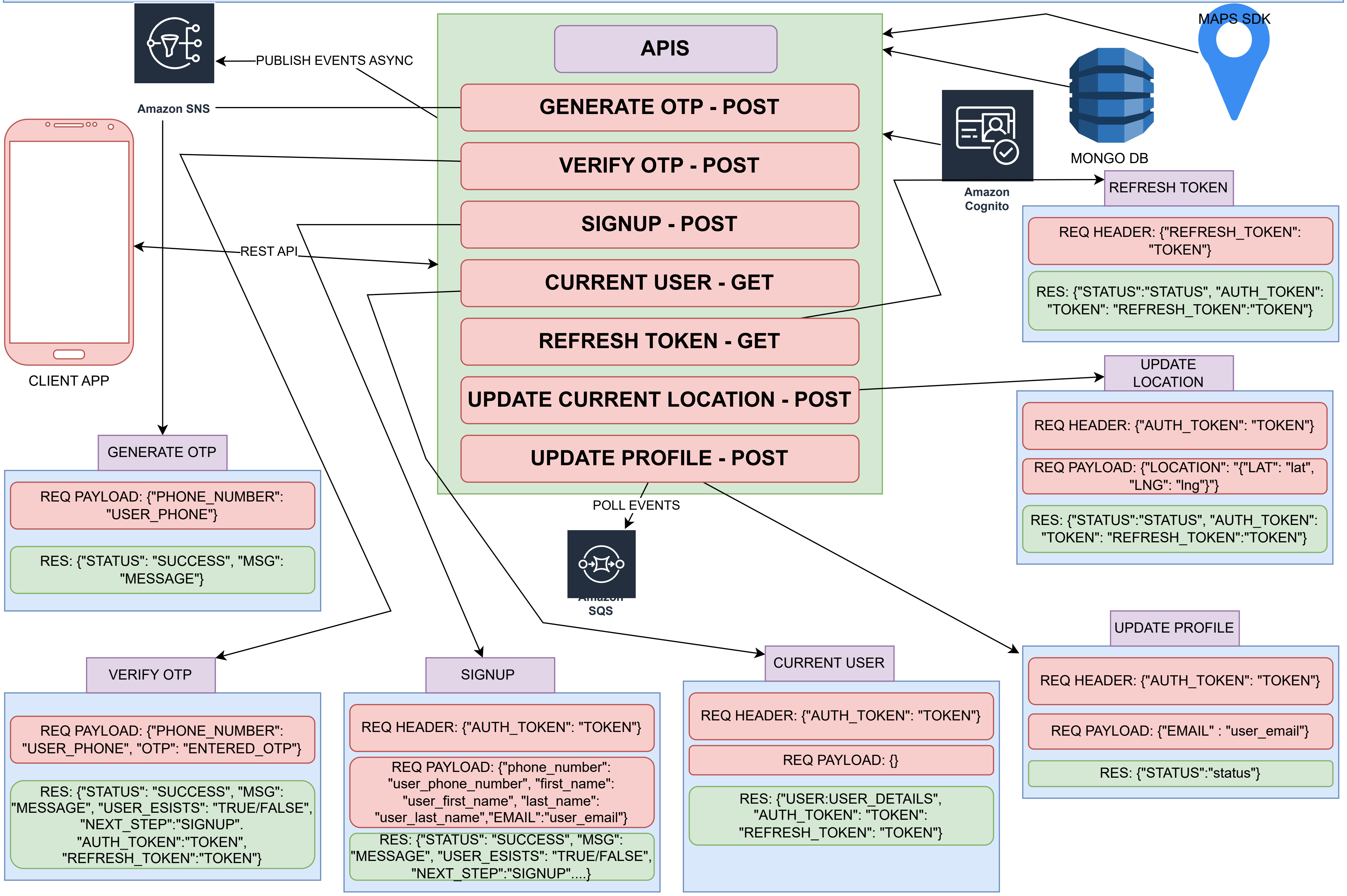
INITIALLY DRIVER WILL GIVE THE AMOUNT IN FIRST REQUEST, RIDER CAN ABLE TO NEGOTIATE ONCE. IN TOTAL TWO NEGOTIATIONS


CLIENT APP
REACT ADMIN APP
ALB - Nginx
SOCKET CONNECTION

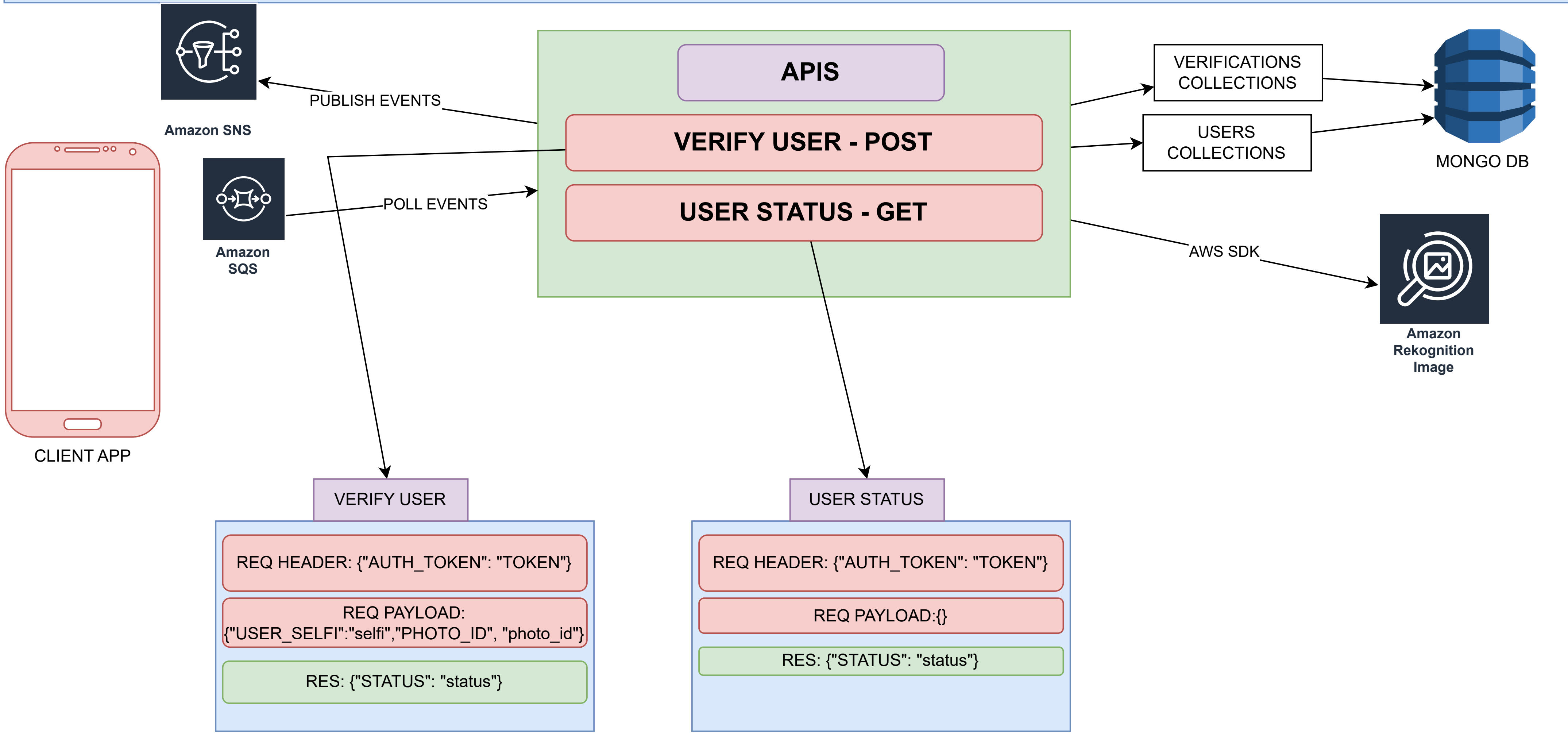
BACKEND DESIGN



AUTH SERVICE DESIGN

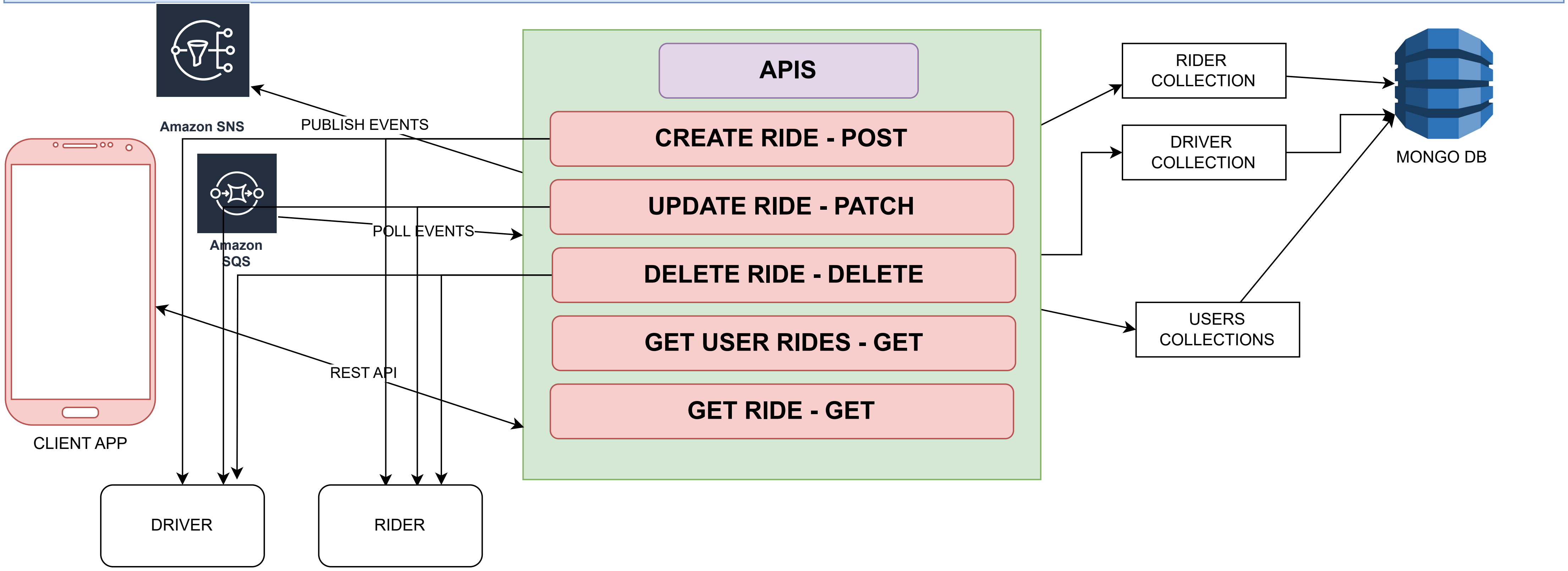

APIS
GENERATE OTP - POST
VERIFY OTP - POST
SIGNUP - POST
CURRENT USER - GET
REFRESH TOKEN - GET
UPDATE CURRENT LOCATION - POST
UPDATE PROFILE - POST
Amazon SNS
PUBLISH EVENTS ASYNC
REST API
CLIENT APP
POLL EVENTS
Amazon SQS
MAPS SDK
MONGO DB
Amazon Cognito
REFRESH TOKEN
REQ HEADER: {"REFRESH_TOKEN": "TOKEN"}
RES: {"STATUS":"STATUS", "AUTH_TOKEN": "TOKEN": "REFRESH_TOKEN":"TOKEN"}
UPDATE LOCATION
REQ HEADER: {"AUTH_TOKEN": "TOKEN"}
REQ PAYLOAD: {"LOCATION": "{"LAT": "lat", "LNG": "lng"}"}
RES: {"STATUS":"STATUS", "AUTH_TOKEN": "TOKEN": "REFRESH_TOKEN":"TOKEN"}
GENERATE OTP
REQ PAYLOAD: {"PHONE_NUMBER": "USER_PHONE"}
RES: {"STATUS": "SUCCESS", "MSG": "MESSAGE"}
UPDATE PROFILE
REQ HEADER: {"AUTH_TOKEN": "TOKEN"}
REQ PAYLOAD: {"EMAIL" : "user_email"}
RES: {"STATUS":"status"}
VERIFY OTP
REQ PAYLOAD: {"PHONE_NUMBER": "USER_PHONE", "OTP": "ENTERED_OTP"}
RES: {"STATUS": "SUCCESS", "MSG": "MESSAGE", "USER_ESISTS": "TRUE/FALSE", "NEXT_STEP":"SIGNUP". "AUTH_TOKEN":"TOKEN", "REFRESH_TOKEN":"TOKEN"}
SIGNUP
REQ HEADER: {"AUTH_TOKEN": "TOKEN"}
REQ PAYLOAD: {"phone_number": "user_phone_number", "first_name": "user_first_name", "last_name": "user_last_name","EMAIL":"user_email"}
RES: {"STATUS": "SUCCESS", "MSG": "MESSAGE", "USER_ESISTS": "TRUE/FALSE", "NEXT_STEP":"SIGNUP"....}
CURRENT USER
REQ HEADER: {"AUTH_TOKEN": "TOKEN"}
REQ PAYLOAD: {}
RES: {"USER:USER_DETAILS", "AUTH_TOKEN": "TOKEN": "REFRESH_TOKEN": "TOKEN"}

VERIFICATION SERVICE DESIGN



Amazon SNS
PUBLISH EVENTS
APIS
VERIFY USER - POST
USER STATUS - GET
VERIFICATIONS COLLECTIONS
USERS COLLECTIONS
MONGO DB
Amazon SQS
POLL EVENTS
AWS SDK
Amazon Rekognition Image
CLIENT APP
VERIFY USER
REQ HEADER: {"AUTH_TOKEN": "TOKEN"}
REQ PAYLOAD: {"USER_SELFI":"selfi","PHOTO_ID", "photo_id"}
RES: {"STATUS": "status"}
USER STATUS
REQ HEADER: {"AUTH_TOKEN": "TOKEN"}
REQ PAYLOAD:{}
RES: {"STATUS": "status"}

RIDE SERVICE DESIGN


Amazon SNS
PUBLISH EVENTS
APIS
CREATE RIDE - POST
UPDATE RIDE - PATCH
DELETE RIDE - DELETE
GET USER RIDES - GET
GET RIDE - GET
Amazon SQS
POLL EVENTS
REST API
CLIENT APP
DRIVER
RIDER
RIDER COLLECTION
DRIVER COLLECTION
USERS COLLECTIONS
MONGO DB