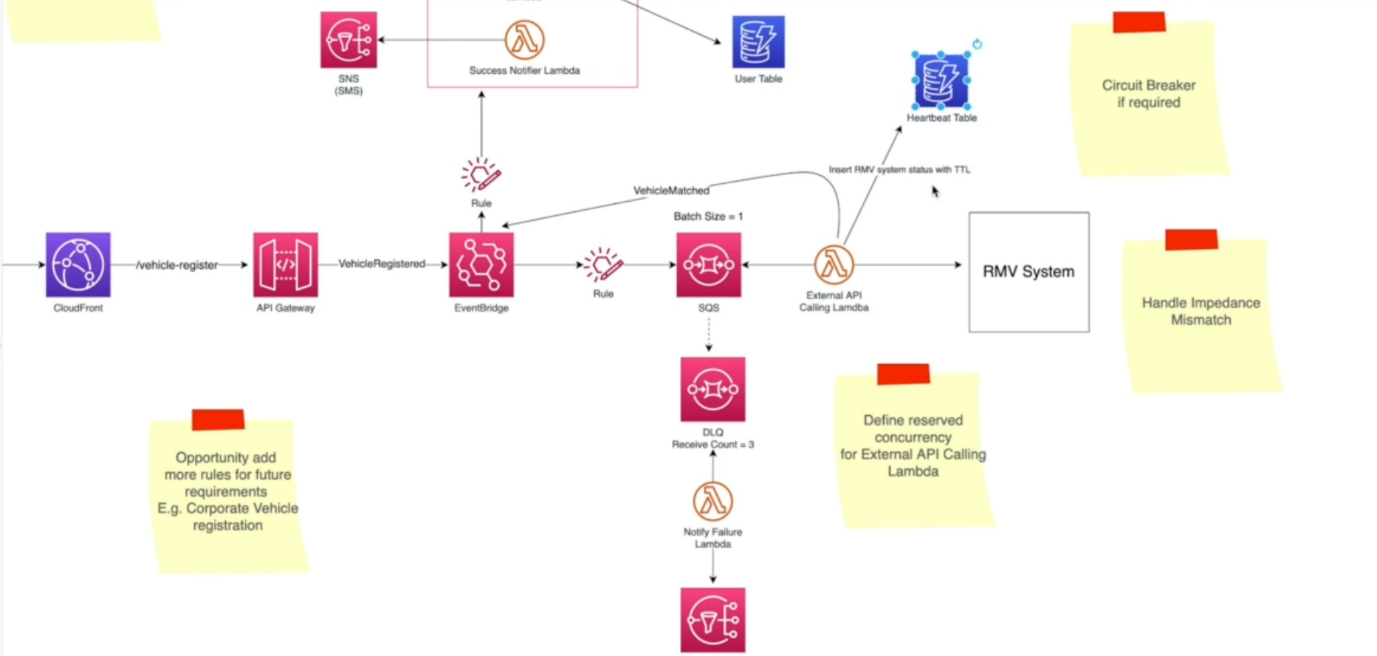
**12. Optimizing with Circuit Breaker Pattern**



--- let us discuss those patterns as well as the user login architecture and then we are done with the system design. Now previously we discussed when a user wants to register their vehicle. Those requests will come through an API gateway and that will lend it to an event bridge. Event bus and then we have defined different rules.

Essentially, these are the filters and if these filters match with this event bridge, we will call upon the target system. In this case event, it will pass the event to an SQS. So, SQS is a durable que and we have attached a serverless function.

Now, this serverless function will consume the events. So, the main purpose of this function is to call upon an external system and verify the details that the users have provided. Now, previously, we also discussed the importance of setting up a DLQ (dead letter queue),

in case the downstream systems, the external systems are offline and this lambda functions cannot reach out to this external system so that the message will become back to this DLQ.

if it receives more than three times in DLQ, we set the receiver count to three. You can set this to any value you like, pass it down to a deal queue and then notify the users through a SNS via SMS messages. Now this works without an issue, but how can we further optimize this failure process?

Now imagine this downstream service (RMV) is offline, so all the messages that are being processed by the Lambda functions will try to call this offline system and the lambda will try not one time, but multiple times, only to find out that the systems are offline and then it will send back the message to SQS and then it will go down to the DLQ.

Now imagine the downstream system is offline because of some reason. So, while it is offline, if our system is also sending too many requests, that is not going to help this downstream system. So, what we can do instead is to use patterns like circuit breakers.

So, circuit breaker, as the name implies, will break the circuit. So, here is how it happens. So, Lambda will try to connect to the downstream service, and it will try a couple of times and it find out the system is offline and it is not responding. So, then what it will do is it will return the message back to SQS and it will record that the system is offline, maybe in a database table also. So, we will create another database table.

In this case, I will call this maybe heartbeat table. So, this lambda function will update the system status in that hardware table. So, it will update a record that RMV system is offline now and when we are adding this record, we can associate a total. So here we will insert RMV system status with TTL (time to leave).

let us say we set the detail to 5 minutes. So once this TTL value is expired, this entry will be automatically removed from the table. So, during this total time, if lambdas start processing another message before we try to reach out to the RMV system, it will check this heartbeat table.

So, if there is any record available related to this RMV system or this third-party system, then it knows that this system is offline. So, what it does is it will short circuit this request, it will not call this request. after this TTL time expired, let us say the 5 minutes expired, then the lambda will start processing another message or the event.

It will again check the heartbeat table, but it will not find any record related to our system. So, the short circuit will no longer happen. It will make the request to RMV system and probably at that time the RMV system is back online. In that case it will be operating normally but if the system is still offline, then again, the lambda function will add another new record with a new TTL.

So, with this circuit breaker pattern, we will reduce the load to the external system when they are offline, as well as we can easily determine if the downstream systems are offline and take actions accordingly.

--- libraries (COCKATIEL)

So, if we want to implement this pattern, there are some health libraries I would like to recommend to you. So, one of the libraries is COCKATIEL. Now this is the resiliency library for Node.js, particularly if you are developing in Node.js and this library provides different helper functions or utility functions. Now one of these functions is circuit breaker and it also provides retries, exponential backups and so on and so forth.