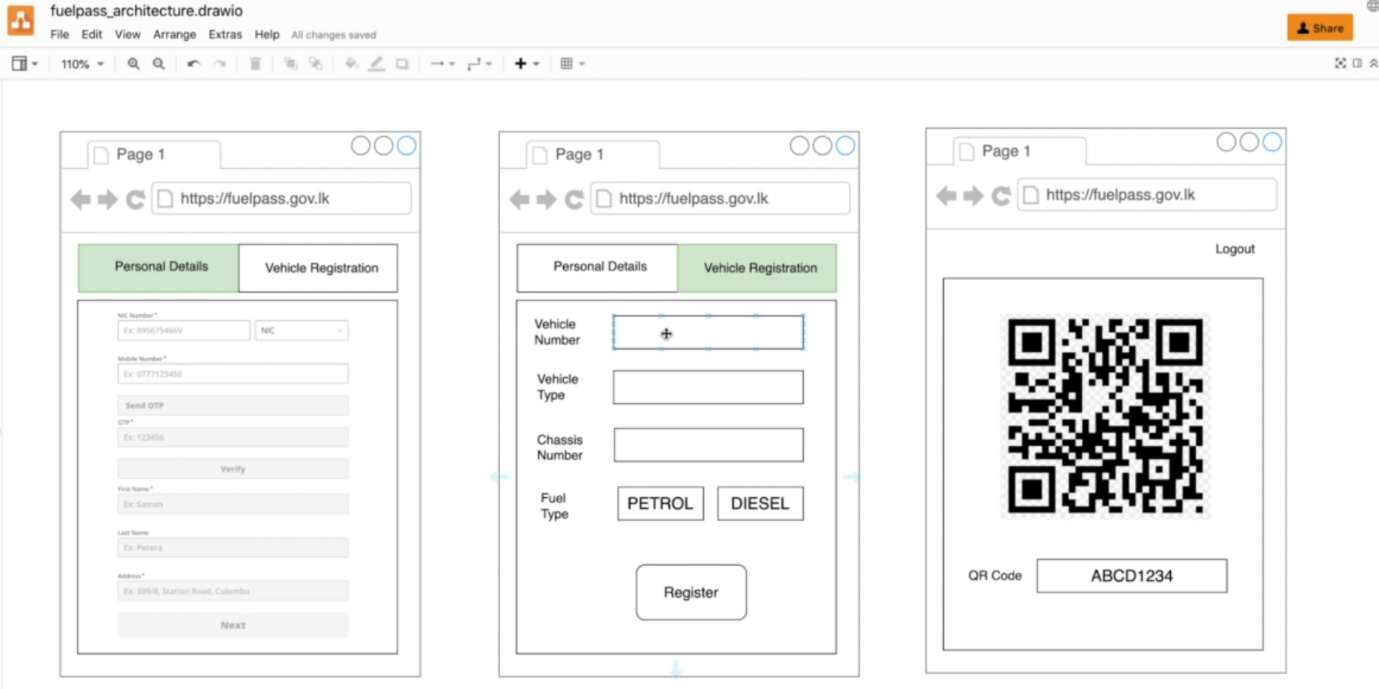
**8. Decoupling the Architecture with SQS an EventBridge**

--- So now we need to discuss about the vehicle registration part of it. That is the second step of the registration process. So once the users are successfully registered, then they must register their vehicle in the vehicle registration process. The application must connect to an external system.

This is the RMV system or the Department of Motor Traffic System to verify the information that the users have added is accurate.

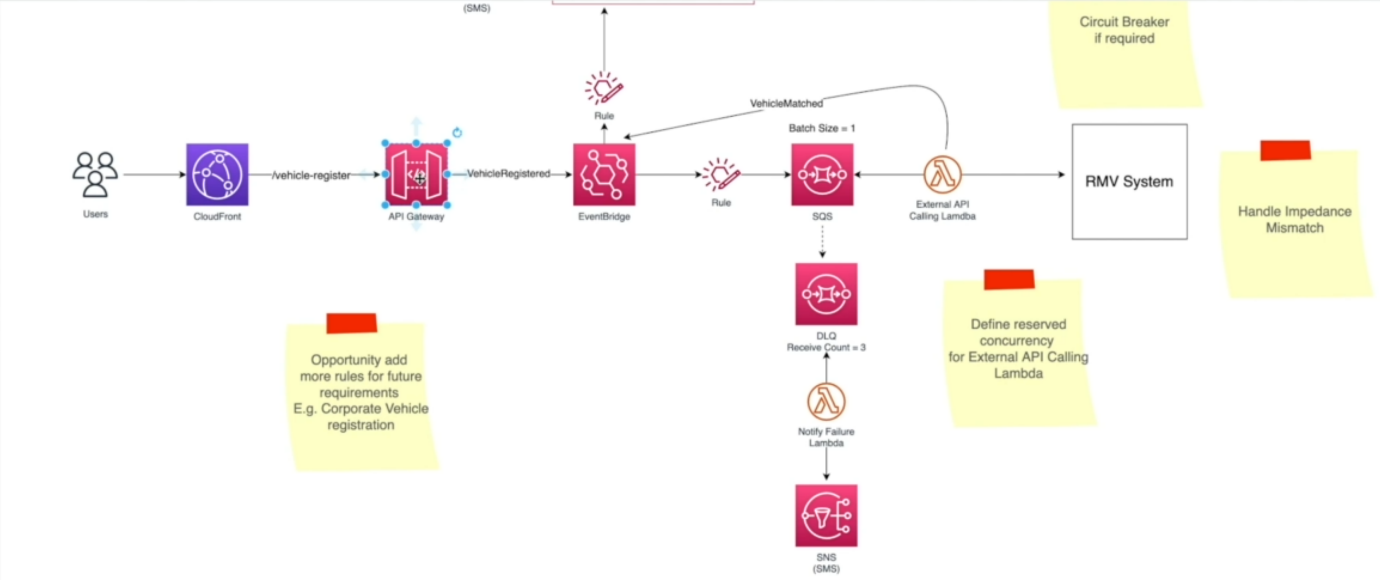
**Application UI**



So, if we go to the UI part of it, users will add their vehicle number, vehicle type, the chassis number, the fuel type, and then they click register.

So, then it will connect to the external RMV system. If the information is accurate, then the user registration will be successful.

**Architecture of vehicle registration**



* Now, by the looks of it, it has lots of connected services. I am going to discuss each one of it and Like I already mentioned, there is nothing called the perfect architecture. That all depends on the business requirements.
* let us discuss if this service architecture can justify the business requirements. Let me come quickly to the UI one more time so the user will feel this detail and click register.
* So, when they click register there will be a HTTP request and as we mentioned, we are using cloud front as the first point of contact and the cloud front will have a behaviour or the path(vehicle-register). That will be pointing that traffic to an origin and in this case, we'll be using an API gateway origin.
* Now API Gateway is the Manage Rest API service from AWS and it will receive this post request. Now this could be a HTTP post request with the details of the vehicle registration and the user details and once it get to the API gateway, the request will be authorized. Now, in the user registration flow, I already discuss how API Gateway can be connected with Amazon Incognito User Pool to verify the token of the connected user and making sure this request is authentic one.
* after it is authorized then we are using another managed service. In this case, we will be using **event bridge**. Now event bridge is the serverless bus service from AWS. So, with **event bridge**, we can create our custom event bus and then we can start publishing events to the event bus.
* Now, since this is a managed service, the built in scalability is there and it can ingest real time events from the producers. Now this acts as a pub sub system that is, there is a producer and we can have multiple subscribers that have subscribed to these events. Now this Event bridge service is pretty much like a SNS,
* the simple notification service. So, this is also act as a pub sub service. We can publish events to a SNS and then we can have multiple subscribers listening to these events. The difference between a SNS and even bridge is that even bridge has content-based filtering. That means it can look at all the attributes of an event. Event, body, event, headers, metadata and so on and so forth. So, it can access all the details of an event.
* Now, essentially, these events are JSON objects and then create filtering rules that will route that traffic to the intended targets.
* Now in this case, I'm creating an SQS. this is a manage queue service from AWS. We call it simple queue service and this is a highly scalable queue. It can ingest millions of messages and store these messages temporarily as well. So, this act as a transient storage, it can retain message up to 14 days. I think the 14 is the maximum number of days and the default is four days, I believe. So even your downstream services are offline, the SQS can ingest events or the messages from upstream services.
* So here we connected **SQS** as a target for event bridge and we created an event rule. So, event rule is basically we will say, what type of events should route to this **SQS**, So, when you are writing an event rule, we can access to the entire body of the event, including the headers and metadata. So here we will write and rule.
* If the event is vehicle registered type, then route all those events or the messages to an SQS. So, when users hit that register button, there will be a post HTTP request with all the information in the post body and that will be received by the API gateway.
* we have directly connected event bridge to API Gateway through service integration. Now usually API Gateway is connected with a lambda function, but here we don't need that lambda functions because we can directly connect to event bridge and that will remove the lambda functions altogether and we will save some cost because if we are using a lambda function it will only use AWS SDK and call SDK method to put event into event bridge. We do not really need that.
* In this case, the lambda is used as a transport mechanism, usually if you will use lambda for transporting then probably you can take it out. However, if you want to transform the request before you send it to the downstream service, then lambda is ideal, but this is just transporting so we can take it out altogether because there is a direct service integration from API gateway to event bridge and that will save you cost as well.
* So, you do not have to bear that lambda cost, execution cost as well as the latency because you don't have to bear any lambda calls start delays as well.
* So, from API Gateway, we will create an event directly in event bridge and we'll call it this is vehicle registered event and then we'll write an event rule to look at this event, registered messages and then route it to SQS.
* So, Event Bridge will directly queue those messages. So we call it and queue these messages into SQS and this is how we connected these three managed services.