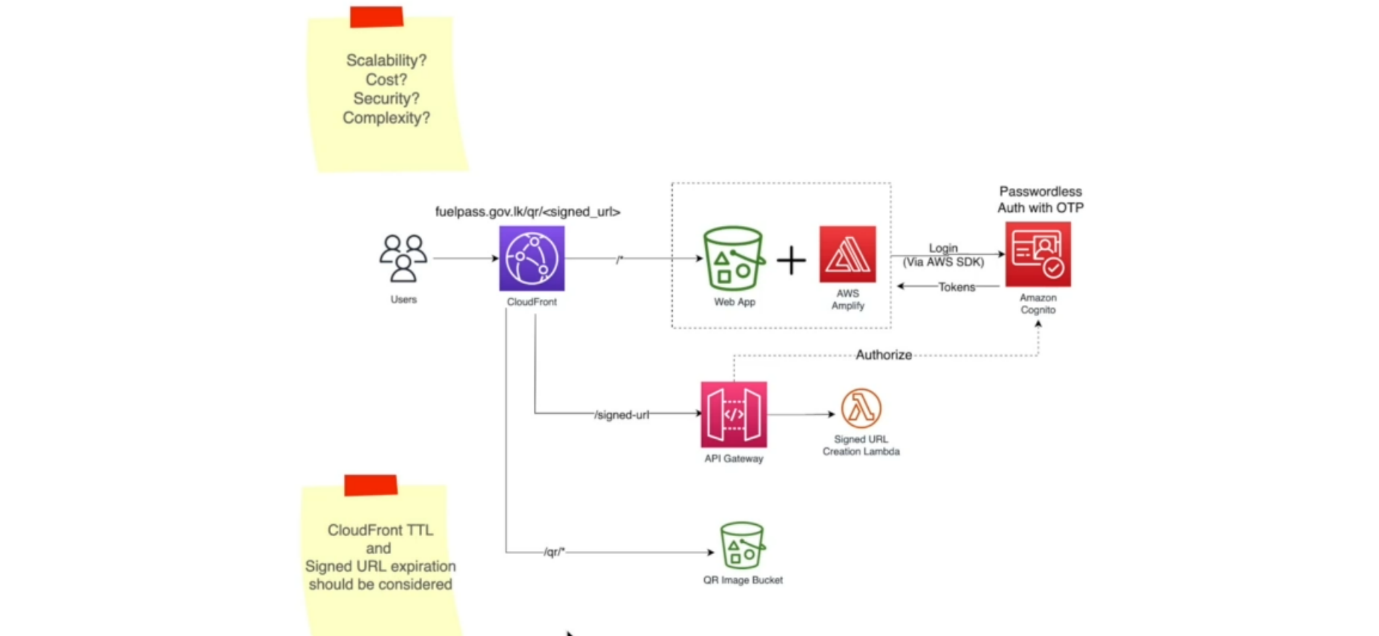
**14. Providing Temporary Access to Protected Resources**



* So now let us move on to our user logging architecture. So now that the user is registered and the vehicle details are also registered, users expecting a QR code and at the end of the user registration process, user will wait until he will receive an SMS with the QR code. So that will happen asynchronously and eventually he will receive an SMS. Okay. Now your QR code is ready.
* So, at that time he will log into the system and the login process also happening through OTP. So, we are going to use the same password less auth with incognito and the incognito will return the tokens and these tokens will be further used to authorize the request to our backend APIs.
* Now in this case, user is requesting for his QR code. So, the QR code is now stored in S3 bucket. Now, previously in the vehicle registration process, we created a QR code in this S3 bucket. Now, this S3 bucket is a private S3 bucket. It means it is not public. We cannot make it public, if the bucket is public then anyone can view anyone else's QR codes and that leads to different chaos.
* So, we will keep the bucket private and once the user logs in, we will expose another endpoint in the API gateway to get a private URL of the user's QR image. Now these short-lived private URLs are typically called as signed URLs or pre signed URLs. So, after users successfully logged in, they will call into this **/signed-url**, get endpoint.
* the API gateway will authorize their request and check whether tokens are valid. If so, we call upon this lambda function which is going to generate a private or assigned URL to the QR image in this QR image bucket.
* After that, it will return that URL to the user and then user can access the QR image using this signed
* URL and he'll again use the cloud front to call upon the URL that was given to him and the cloudfront will have a different path that will forward that request to the private QR image bucket, and this S3 bucket will return that QR image and then it'll get cached on the cloud front.
* remember we already discussed about cloudfront caching and the user can now see it. So, the second time they want to see the QR code, they can just open the app and the cloud front will return the image URL immediately without calling this S3 bucket and that will of course happen if the cache is not expired. The TTL (time to leave)
* Now when you are using signed URL with cloud front, that is one of the concerns I want you to think about and that is about TTL value in the cloud front. So, when you are generating a signed URL from this lambda function, you can specify how long this signed url is going to be valid. So, they are you again define a title value. Let us say it is 5 minutes. So, after 5 minutes this signed URL will be invalid.
* if you have a larger TTL value in the cloud front, even if you expect that the image will not be returned to the users after 5 minutes, the cloud front will still serve it because it has it on its cache. So, in order to prevent that, you had to make sure that the cloudfront TTL and the signed URL either they must be the same or else the cloud front TTL should be lower than the signed URL deal. That is one of the concerns.