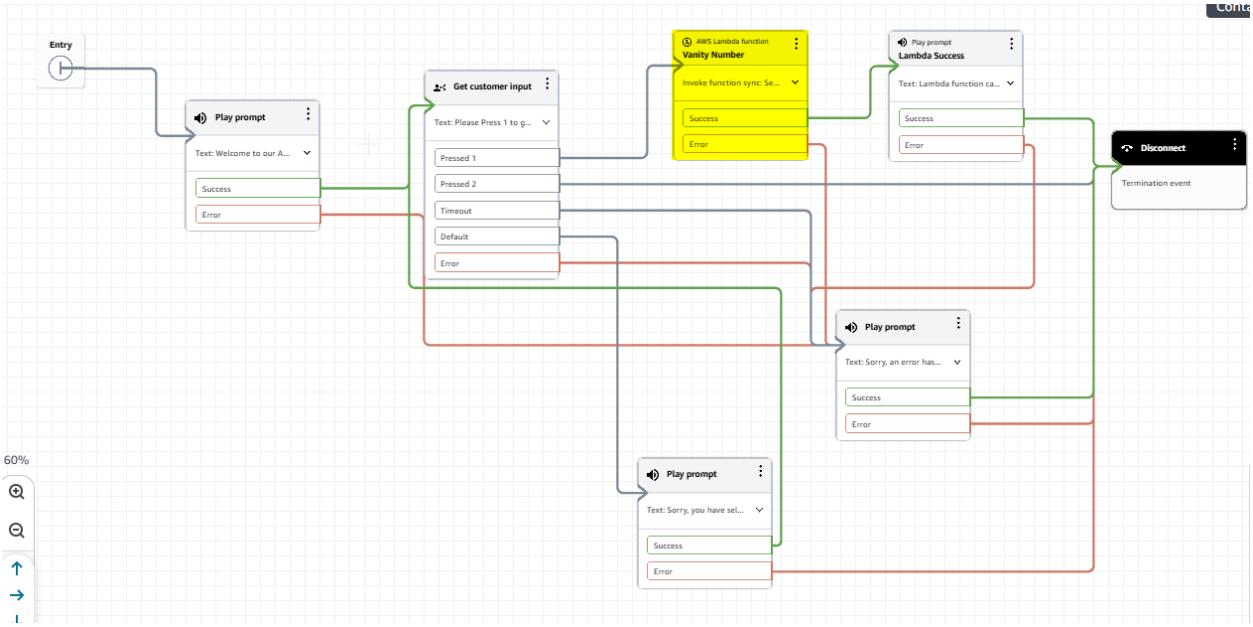
Code Repository: https://github.com/ramzilla007/senior-dev-proj.git

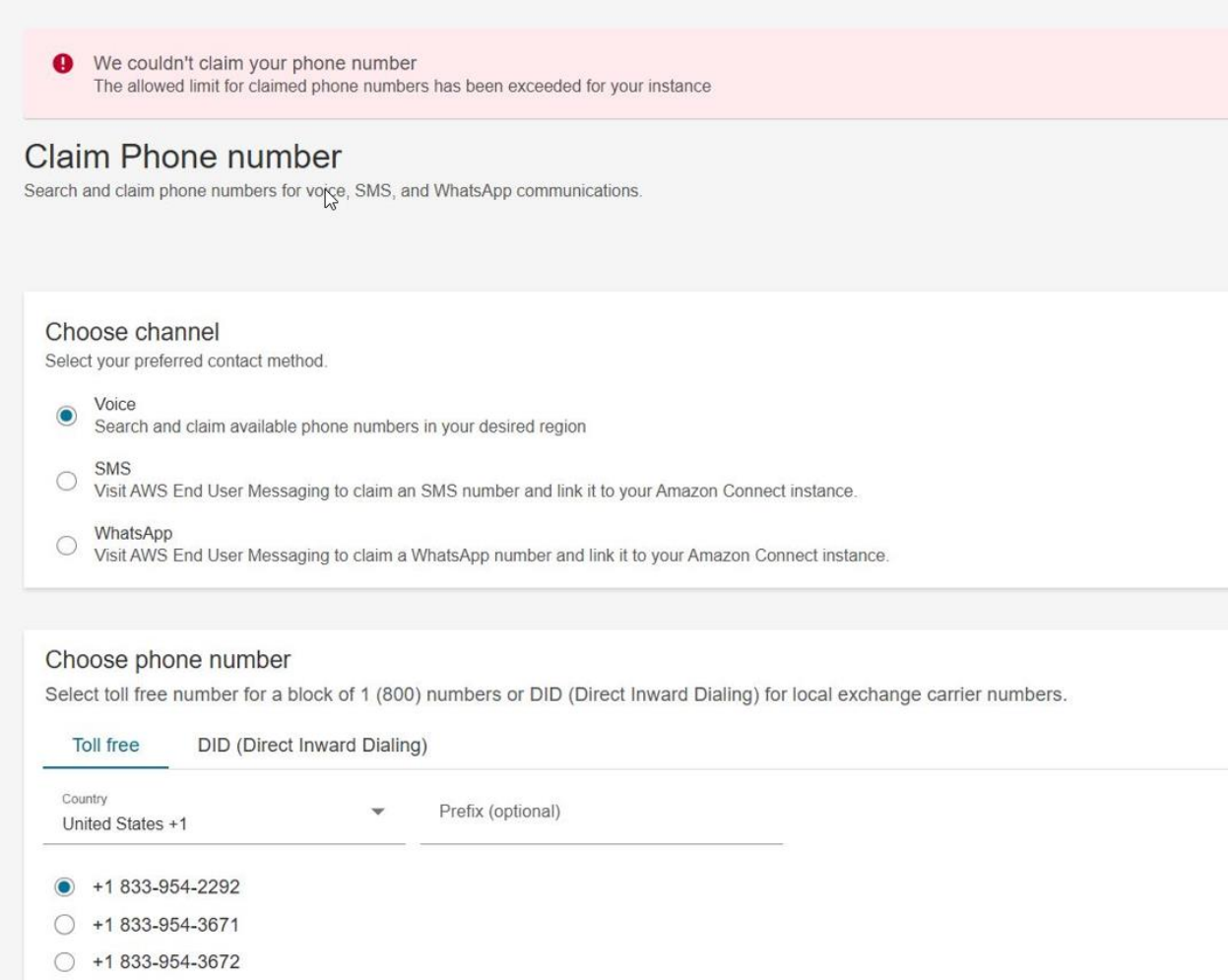
RECORD YOUR REASONS FOR IMPLEMENTING THE SOLUTION THE WAY YOU DID, STRUGGLES YOU FACED, AND PROBLEMS YOU OVERCAME.

Based on the requirements provided, I used AWS CDK to create a CloudFormation stack that creates a DynamoDB table and deploys a Lambda function. This Lambda function takes in an Amazon Connect Contact Flow JSON, reads in the caller's number, generates 5 vanity numbers, write those 5 numbers to the DynamoDB table, and outputs the top 3 vanity numbers for the contact workflow to read back to the caller.

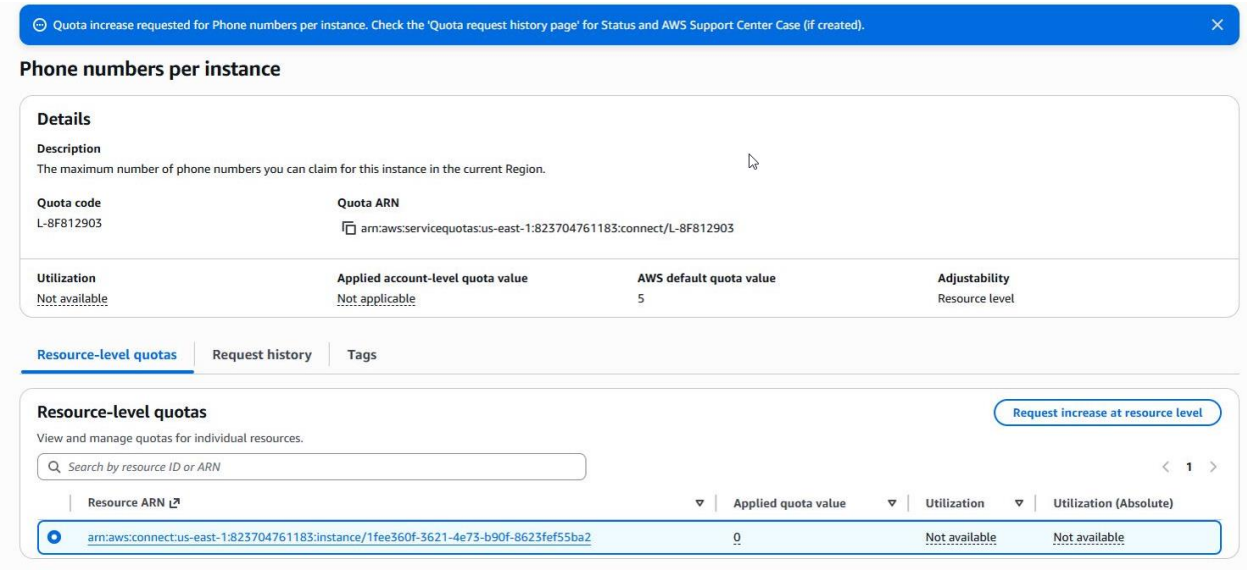
I have also attempted to use the CDK to create the Amazon Connect Contact Flow but after spending several hours trying to manipulate the code and use different Contact Flow JSON objects, I was unsuccessful in deploying the contact flow, I kept getting deployment exceptions for malformed JSON. To work around those issues, I have opted to Manually create the flow. I have exported the flow so it can be imported back in for testing:



Another issue I have faced, and could not work around, was claiming a number from Connect. I tried different regions to claim a number, but I would always get the error below even though I have not claimed any numbers before:



I went over to my Service Quotas and under Connect > Phone numbers per instance and noticed I had a “5” default value but none applied at my account level. I opened a case with AWS to increase my quota but still have not recieved a response.



With no access to a valid number, I used Lambda's console to test my function passing it a "Connect Contact Flow" as a template and changing the phone number under:

Event JSON

```
{
  "Name": "ContactFlowEvent",
  "Details": {
    "ContactData": {
      "Attributes": {},
      "Channel": "VOICE",
      "ContactId": "5ca32fbd-8f92-46af-92a5-6b0f970f0efe",
      "CustomerEndpoint": {
        "Address": "+11234567890",
        "Type": "TELEPHONE_NUMBER"
      },
      "InitialContactId": "5ca32fbd-8f92-46af-92a5-6b0f970f0efe",
      "InitiationMethod": "API",
      "InstanceARN": "arn:aws:connect:us-east-1:123456789012:instance/9308c2a1
```

WHAT SHORTCUTS DID YOU TAKE THAT WOULD BE A BAD PRACTICE IN PRODUCTION?

Shortcuts that should be fixed before production:

1. Did not add as much error checking and logging in my lambda function
2. I used a simple way to generate vanity numbers where I could have used a search tool to come up with more meaningful numbers.

3. Did not add a lot of error checking on the contact flow
4. In my DynamoDB table, I did not setup sort keys and did not consider capacities (RCU/WCU)

WHAT WOULD YOU HAVE DONE WITH MORE TIME?

If I had more time, I would have built a web app to display entries from the DynamoDB for the generated vanity numbers. I would have also liked to get a connect phone number working to be sure the caller hears back those generated vanity numbers.

WHAT OTHER CONSIDERATIONS WOULD YOU MAKE BEFORE MAKING OUR TOY APP INTO SOMETHING THAT WOULD BE READY FOR HIGH VOLUMES OF TRAFFIC, POTENTIAL ATTACKS FROM BAD FOLKS, ETC.

If we want to make this app more secure and ready for high volume traffic, we can:

1. Use separate IAM role for the Lambda function.
2. Encrypt Lambda environment variables at rest.
3. Add DynamoDB at-rest encryption with customer-managed KMS keys
4. Encrypt in transit for HTTPS/TLS for all API calls (for the web app).
5. VPC endpoints for DynamoDB using Gateway VPC endpoints.
6. Enable scaling for Lambda, DynamoDB, and Contact flow.
7. Ensure that Lambda that Connect invokes has a restricted role and only the ability to read/write specific DynamoDB items.
8. Add CloudWatch logs for all services involved.

ARCHITECTURE DIAGRAM