**Customer Care Calling Service**

**INRODUCTION**

The idea for this project stemmed from the fact that MNC are investing a lot on customer service and they are keeping the customers as first priority in this era. Amazon’s leadership principles start with the point about customers obsession. Hence, we thought about building a fully functional customer care service that uses AWS Services like,

1. Connect <https://aws.amazon.com/connect/>
2. Transcribe <https://aws.amazon.com/transcribe/>
3. Comprehend <https://aws.amazon.com/comprehend/>
4. QuickSight <https://aws.amazon.com/quicksight/>
5. RDS <https://aws.amazon.com/rds/>
6. S3 <https://aws.amazon.com/s3/>
7. DynamoDB <https://aws.amazon.com/dynamodb/>
8. LEX <https://aws.amazon.com/lex/>
9. Lambda <https://aws.amazon.com/lambda/>

**FLOW**

This is the most important service that helped us build this project from scratch was AWS **Connect**. Connect provides users with unique phone numbers that can be used for starting the interaction with the service. The experience that the user gets once they call into the number is decided by **contact flows** that help the user navigate through the various available options.

In our application we integrated **LEX** into the **contact** **flow** inputs so that the customer could also interact with the service using **voice inputs** along with the usual keypad inputs. If the customer asks a question that was part of a predefined set of FAQ’s, then the answer for the question would be fetched automatically from DynamoDB. For example, if a customer wants to know their account balance, then there is no need for an agent to intervene, the query can be answered directly by fetching data from the database. **LEX intends** were given for answering these questions.

The routing of calls for different issues was implemented using **in inbuild queues in contact flows**. Agents would be assigned to queues and depending on availability calls would be routed through to respective agents.

Connect features that were used for the project are -

1. **Check queue status** – returns the waiting time in a particular queue
2. **Get customer input** – receives inputs from customers in form of text or voice
3. **Set working queue** & **Transfer to queue** – route call to specific queue
4. **Store customer input** – input phone number of customer
5. **Set contact attributes** – attach phone number to the customer so that it can be verified at later stage
6. **Set callback number** – sets the number that need to be pushed into the callback queue
7. **Check hours of operation** – call will get connected only if its made in set time period in a day
8. **Set voice** – sets the default voice that will interact with the customer
9. **Set call recording behavior** - records the conversation between the agent and customer in S3
10. **Quick Connect** – used for transferring calls from one agent to another
11. **User management** – create admins and agents for interactions with the customer

When the conversations get stored in **S3** (A) bucket, a **Lambda** function is triggered that will **Transcribe** the audio into text and store in into another S3 (B) bucket. Now for analyzing this transcribed text, we used **Comprehend** service, which will help us implement **sentiment** **analysis** on the obtained text file. This analysis is triggered whenever a new text file gets added to bucket B and its output is stored in **RDS** in the format, (**Timestamp, Sentiment**). We decided to use RDS because it was highly scalable and our data was relational in nature. We could also integrate **RDS** with **QuickSight** for creation of dashboards. The final dashboard in QuickSight showed the number of conversations in each type of sentiment (Positive, negative and neutral).

