

# **Customer Support Call Intelligence and Analytics**

DAMG 7245 - Spring 2022 - Team 7

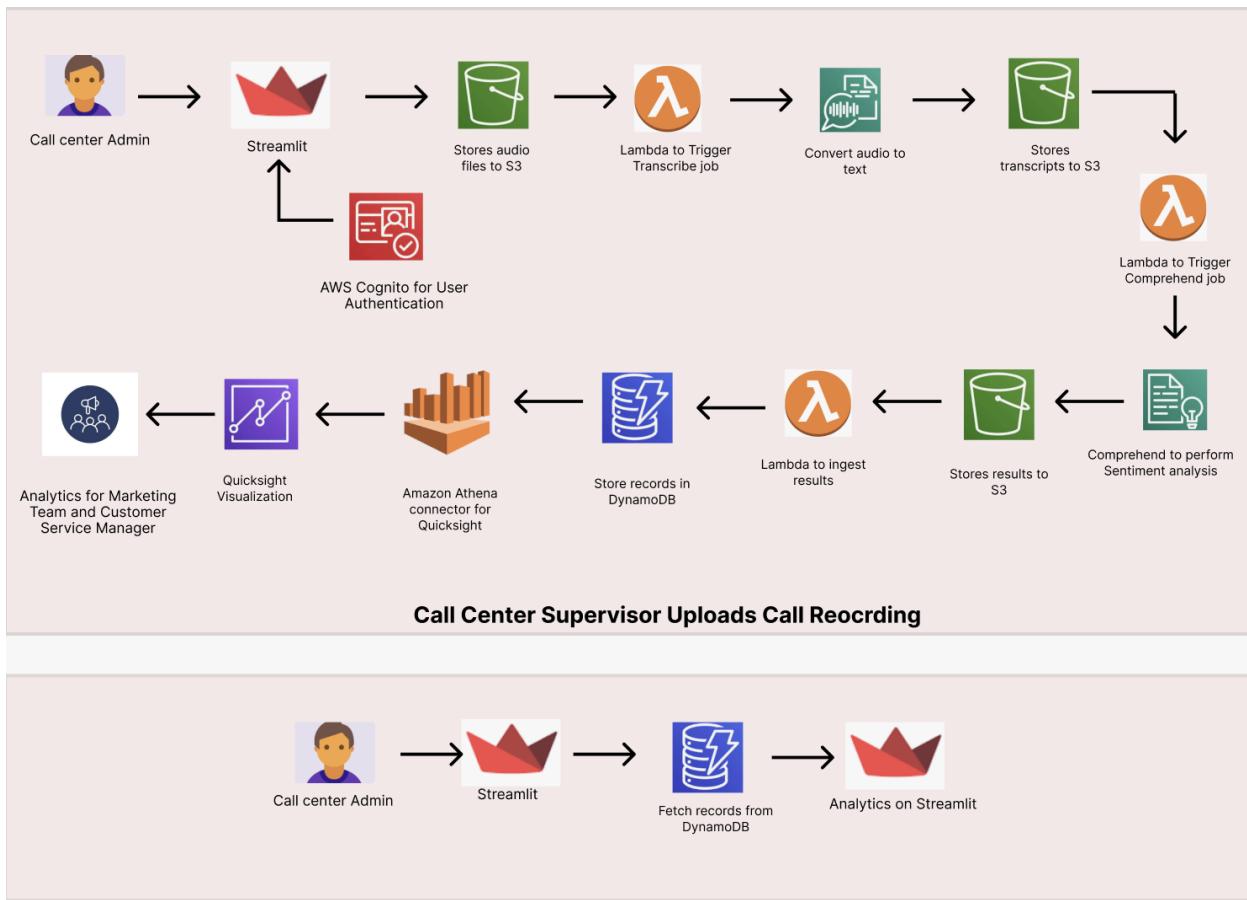
NAME	NUID
Greeshma Tatineni	002130050
Swathi Sharma	002105916
Kashyap Datta Kovvali	001098717
Bala Yeshwanth M	002191719

## **Objective/Problem Statement**

A contact center connects the business to the customers, enabling them to request support, make appointments, and much more. Each conversation with a caller is an opportunity to learn more about that customer's needs, and how well those needs were addressed during the call.

We have built a Customer Support Call Intelligence and Analytics application to analyze the sentiment of the calls received to the call center from the customers, which would help in building a valuable customer base and improving the business. Apart from this, the analysis would help the Call Center Supervisor evaluate the performance of the customer service executives or train them.

# Architecture



# Workflow

## Admin Uploads the Call Recording



An admin can sign up as a new user, and later can log into Streamlit with a designated username and password. He/She can then upload a recording of an agent speaking with a customer on the Streamlit frontend, of the format .wav or .mp3



AWS Cognito is used for user authentication while logging into Streamlit. Only authenticated users will be able to access the Dashboard on Streamlit



AWS S3 to store Audio files - After a call recording has been uploaded on Streamlit frontend, it is stored as an object in the AWS S3 bucket. For every new recording being uploaded, it is stored as a new object in the same bucket. This way, we will be storing all the audio files in one place.



AWS Lambda to Trigger Transcribe Job - When an audio file is created as an object, a lambda function is triggered in order to start the Transcribe job



AWS Transcribe - Amazon Transcribe is an automatic speech recognition service that makes it easy to add speech to text capabilities to any application. Transcribe's features enable it to ingest audio input, produce easy to read and review transcripts, improve accuracy with customization, and filter content to ensure customer privacy. The lambda function that triggered the transcribe job previously, gives us a transcription everytime a new audio file is placed in the S3 bucket. This transcribed text is then saved in another (new) bucket in .json format.



AWS Lambda to Trigger Comprehend Job - After the transcription file is generated and placed in a new bucket, another lambda function is triggered to fetch this file and perform sentiment analysis using AWS Comprehend



AWS Comprehend - Amazon Comprehend uses natural language processing (NLP) to extract insights about the content of documents. It develops insights by recognizing the sentiments and other common elements in a document. Every time a new transcribed file is generated, a lambda function is triggered to run the comprehend job. We have utilized chunking of text files as Amazon Comprehend imposes a size limit of 5000 bytes on each text file. Since the audio files' transcriptions are usually greater than 5000 bytes, chunking the text files into smaller files helps in running the comprehend job successfully. We get a sentiment, as well as a sentiment score for each of the chunks and we then take an aggregate score of the sentiment to declare an overall sentiment of the call. The result is stored as a json file in the S3 bucket.

This process will be repeated each time a new audio recording is uploaded into the S3 bucket. The file gets uploaded to S3, goes through Transcribe to get a transcription and finally goes through Comprehend to get the sentiment of the call for each agent-customer combination. Now that we have the data , we will be performing some analysis to view the results in a concise dashboard, using AWS Quicksight



AWS DynamoDb - Amazon DynamoDB is a NoSQL database that supports key-value and document data models. The result data that is in the json format is stored in DynamoDB as key:Sentiment and value:Sentiment Score pairs. This is then connected to Amazon Athena using Athena connectors



AWS Athena - Amazon Athena is an interactive query service that makes it easy to analyze data directly in Amazon S3 using standard SQL. We will be using Athena to query the DynamoDb table and get the results, for further analysis in Quicksight. Since DynamoDb cannot be integrated directly with Quicksight (because of architectural constraints imposed by AWS), we are using Athena as a connection between DynamoDb and Quicksight



AWS Quicksight - Amazon QuickSight allows everyone in the organization to understand the data through interactive dashboards. Quicksight will enable call center admins and other stakeholders to view analytics with respect to performance of the agents, number of calls attended and number of happy customers.

## Historic Call Data Analysis of an Employee



Admin Login - Admin can log into Streamlit - for a better and easier user experience to view dashboards and analytics of their subordinates



Records are fetched from DynamoDb and analytics is shown on Streamlit frontend

## Application Setup

### Streamlit Setup:

Download the available streamlit folder

Install the following packages

```
pip install streamlit pip install boto3
```

Run the application locally using following command

```
streamlit run app.py
```

Go to <http://localhost:8501> to view the application locally

### AWS Setup:

**AWS** - Create an AWS account, if you don't have one already

**AWS Cognito** - Create user pool and setup an app client. User pools are directories of federated and local user profiles. They provide authentication options for the users.

**AWS S3** - Create a bucket to store various objects on S3 like audio files, transcripts, sentiment data etc.

**AWS S3 event triggers** - Place event triggers on the landing folders to kick off the lambda functions. For example if an audio file is uploaded to S3 in a particular path, the event trigger will kick off the lambda function for the AWS Transcribe job.

**AWS Lambda** - Create 3 lambda function to accept the audio files, transcribe, comprehend, load the sentiment analysis to dynamodb and visualize on quicksight. Use the code provided in src/Lambda Functions folder.

**AWS IAM roles** - Create IAM roles for each of the lambdas to access the AWS Transcribe, AWS Comprehend, DynamoDb, CloudWatch and S3

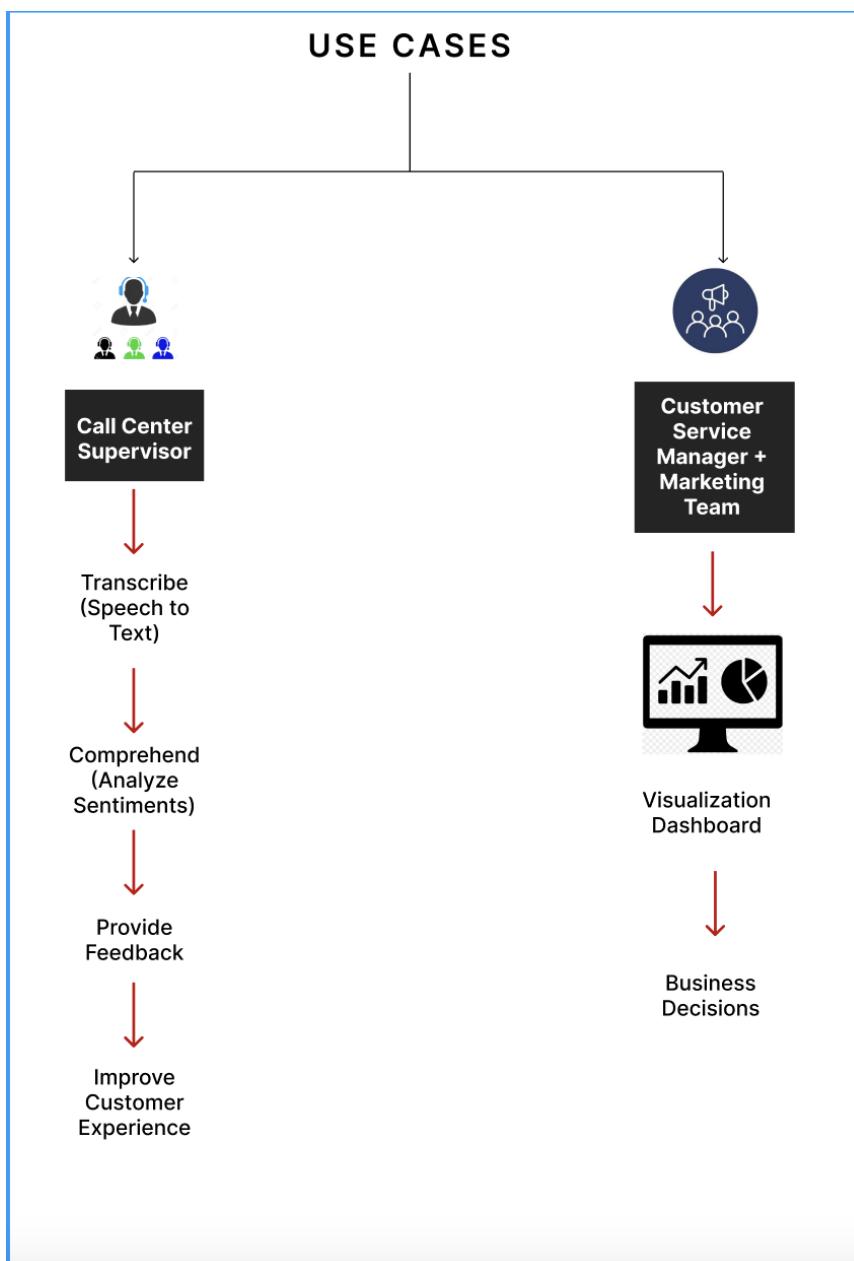
**AWS Athena** - The data in Dynamodb cannot be directly visualized on AWS QuickSight, we are using Athena connectors to be able to visualize the data on quicksights

**AWS QuickSight** - The Marketing Manager and the Call center Manager are provided with an interactive dashboard with various insights on sentiment analysis.

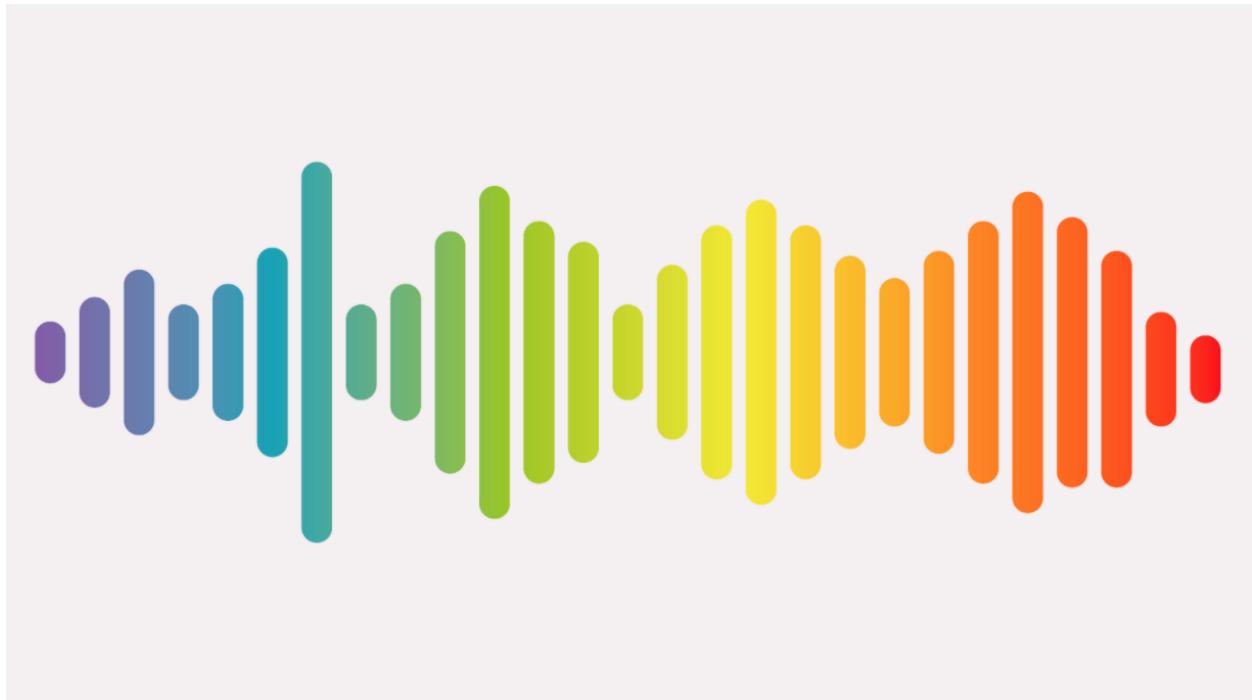
**Quicksight Access** - The marketing manager and the customer service manager will be given access to the quicksight dashboard to view the analytics. They need to follow the below steps to access the dashboard via the link:  
<https://us-east-1.quicksight.aws.amazon.com/sn/dashboards/aab311a1-27ba-40fc-80ba-532ddd1c8dd2>

1. Create a quicksight account once the email with the invitation is received
2. account name: bigdata-team7, make sure the correct account name is passed during login
3. View the interactive dashboard

# Use Cases



# Dataset

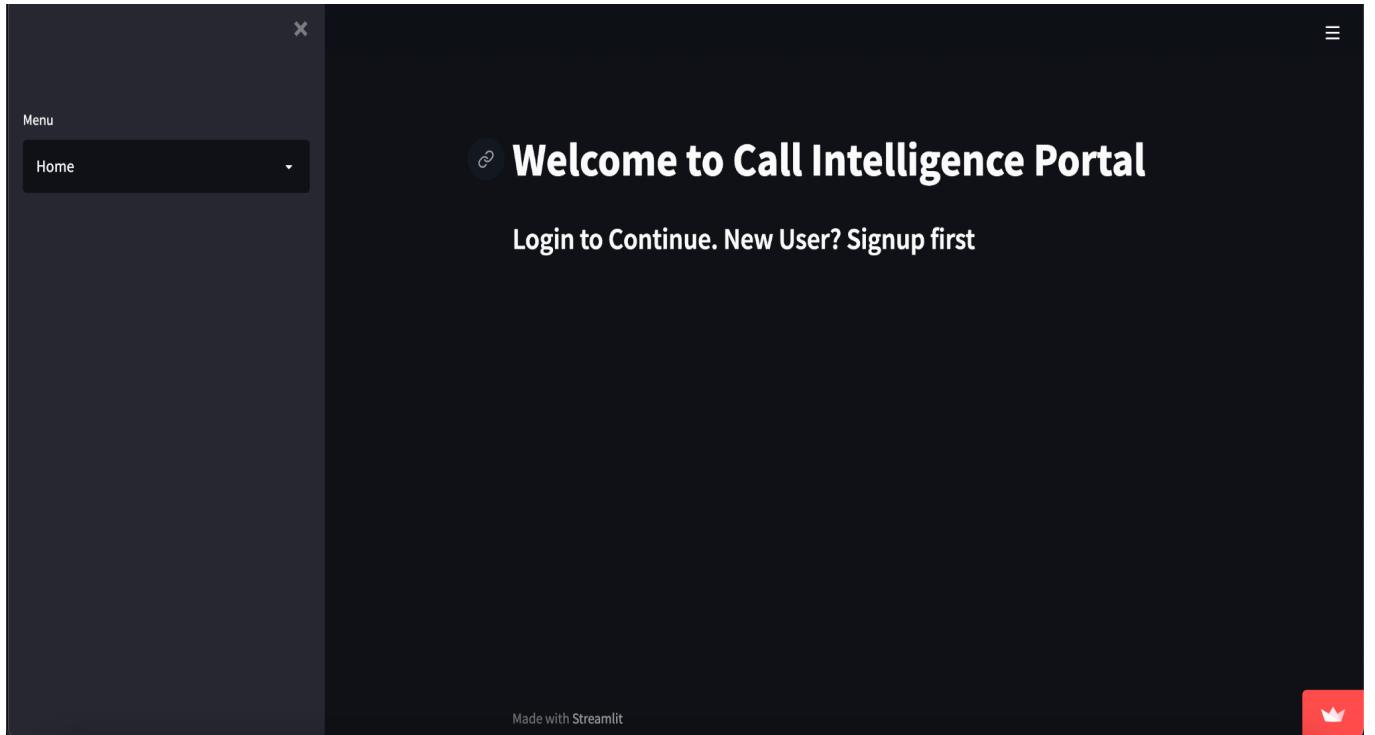


The dataset consists of call recordings between agents and customers. We will be working on these audio files to convert them to text files in order to perform sentiment analysis and visualizations so that the admins can get insights on how well their employees are performing, and how satisfied their customers are.

**Dataset Link :** <https://media.talkbank.org/ca/CallHome/eng/>

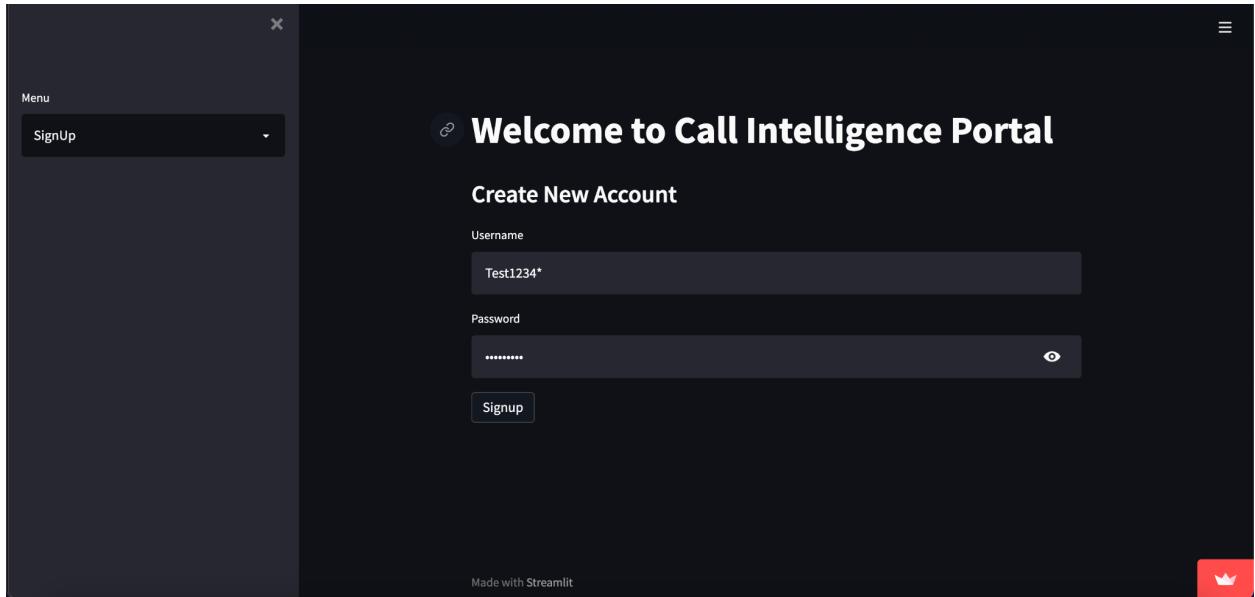
# Implementation

## Login Page



Services used : Amazon Cognito for authentication of the user. Once a new user is created, the user is authenticated at the backend. Only authenticated users have access to login into the portal.

## Signing Up



## Login



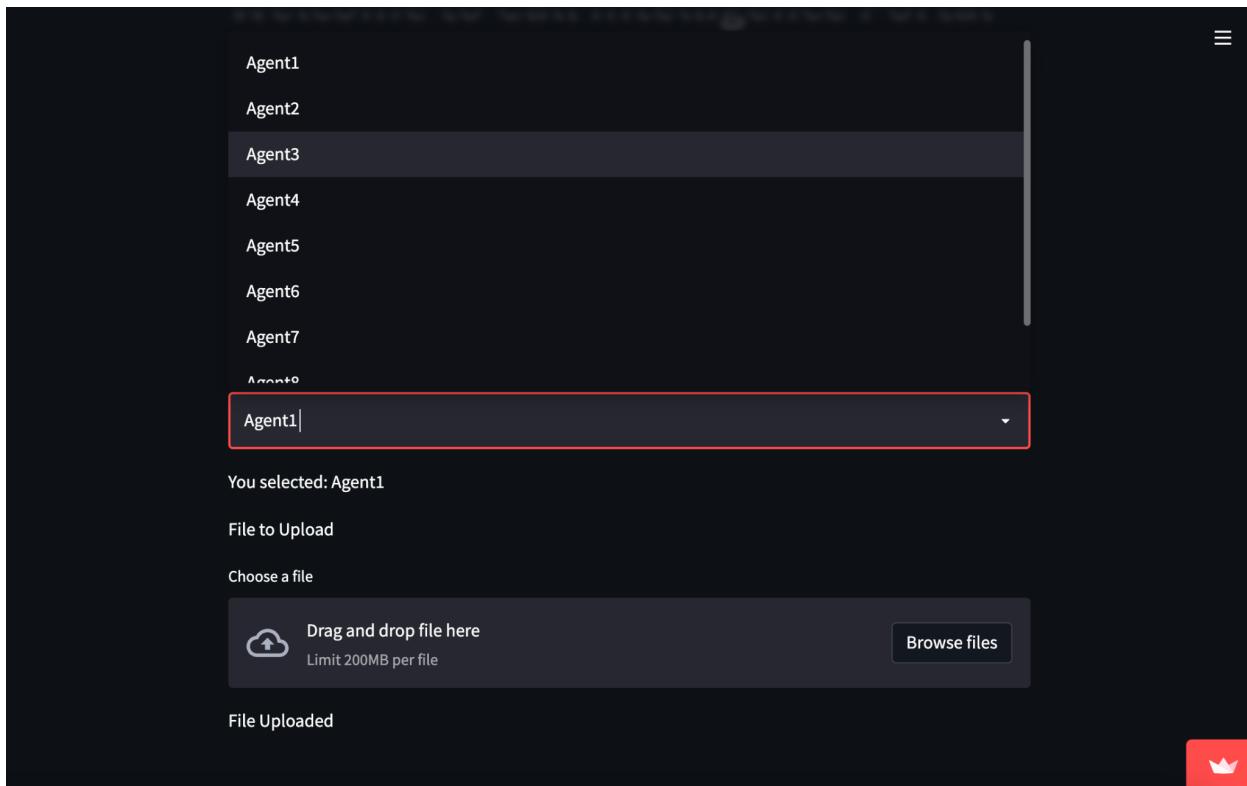
## Call Intelligence Portal

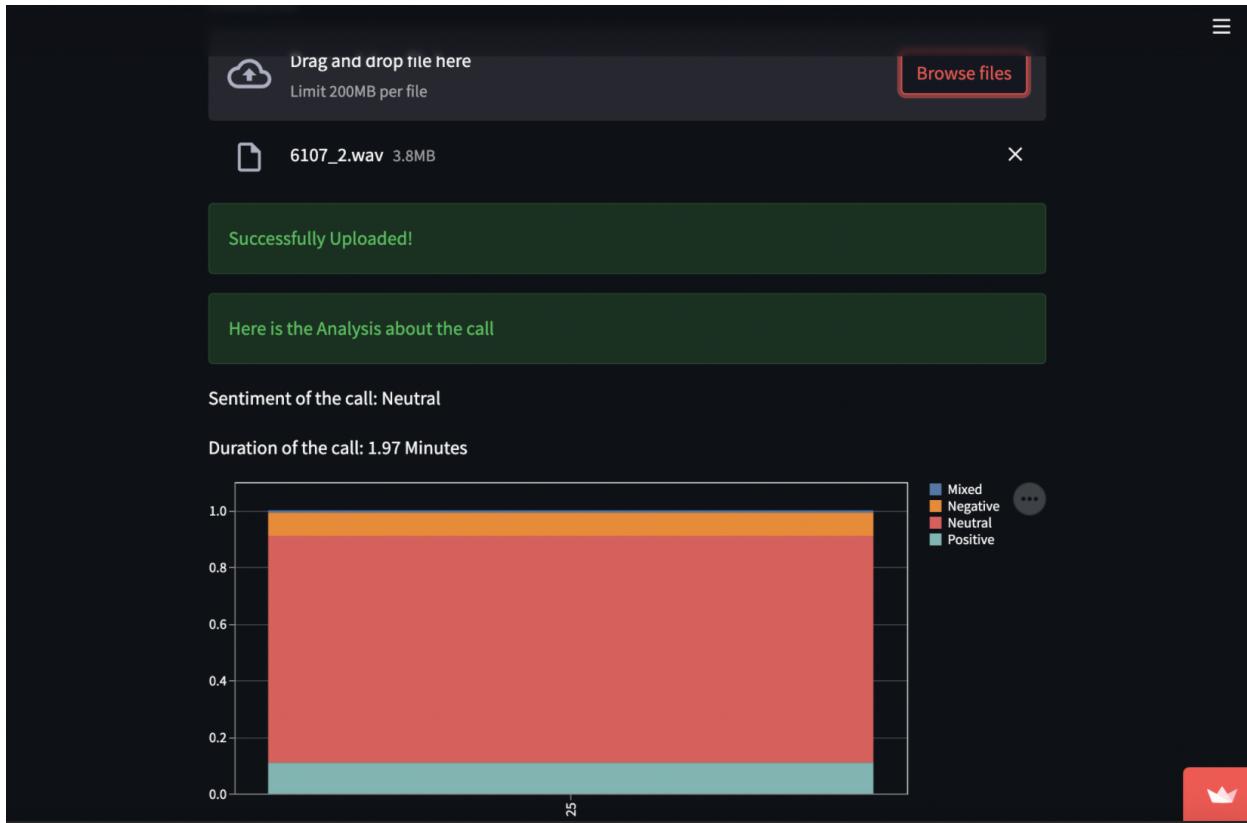
After logging in, the user (admin) has three options to select from -

- Upload Audio - To upload an audio file recording of an existing agent with a customer
- Agent Performance - To look up an agent's performance, get insights into how the agents are catering to the customers
- Create Agent - To create a new agent, if there is a need

## Uploading Audio

When the admin chooses to upload audio, there is an option to choose what agent to upload for, and is able to upload a file with <200MB limit





## Analysis -

### Storing the audio file in an S3 bucket

The screenshot shows the AWS S3 console. The left sidebar has sections for Buckets, Storage Lens, Feature spotlight, and AWS Marketplace for S3. The main area displays a table of buckets:

Name	AWS Region	Access	Creation date
agent-recordings	US East (N. Virginia) us-east-1	Bucket and objects not public	April 25, 2022, 19:29:32 (UTC-04:00)
amazon-connect-b5cd9177cf78	US East (N. Virginia) us-east-1	Objects can be public	April 22, 2022, 14:29:30 (UTC-04:00)
audio-analysis-data	US East (N. Virginia) us-east-1	Objects can be public	April 23, 2022, 15:16:35 (UTC-04:00)
aws-athena-query-results-us-east-1-825406695460	US East (N. Virginia) us-east-1	Bucket and objects not public	April 28, 2022, 18:20:45 (UTC-04:00)
files789	US East (N. Virginia) us-east-1	Bucket and objects not public	April 29, 2022, 09:26:49 (UTC-04:00)
insights-sentiment	US East (N. Virginia) us-east-1	Bucket and objects not public	April 28, 2022, 16:07:21 (UTC-04:00)
insights-sentiments-spill-bucket	US East (N. Virginia) us-east-1	Objects can be public	April 28, 2022, 17:58:29 (UTC-04:00)
testteam19	US East (N. Virginia) us-east-1	Bucket and objects not public	April 26, 2022, 22:09:33 (UTC-04:00)

At the bottom, there are links for Feedback, Unified Settings, Copyright notice, Privacy, Terms, and Cookie preferences.

The screenshot shows the AWS S3 console interface. On the left, there's a sidebar with various options like Buckets, Storage Lens, and Feature spotlight. The main area is titled 'Objects (9)' and contains a table with nine entries, each representing a folder named 'Agent1/' through 'Agent9/'. The table includes columns for Name, Type, Last modified, Size, and Storage class. At the top of the main area, there are several action buttons: Copy S3 URI, Copy URL, Download, Open, Delete, Actions (with a dropdown arrow), and Create folder. Below the table is a search bar labeled 'Find objects by prefix'.

	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	Agent1/	Folder	-	-	-
<input type="checkbox"/>	Agent2/	Folder	-	-	-
<input type="checkbox"/>	Agent3/	Folder	-	-	-
<input type="checkbox"/>	Agent4/	Folder	-	-	-
<input type="checkbox"/>	Agent5/	Folder	-	-	-
<input type="checkbox"/>	Agent6/	Folder	-	-	-
<input type="checkbox"/>	Agent7/	Folder	-	-	-
<input type="checkbox"/>	Agent8/	Folder	-	-	-
<input type="checkbox"/>	Agent9/	Folder	-	-	-

## Transcription of the audio file

Transcription is done by Amazon Transcribe and stored in the S3 bucket in a .json format under the object 'transcripts', as and when the lambda function is triggered for the same

## Transcribed file

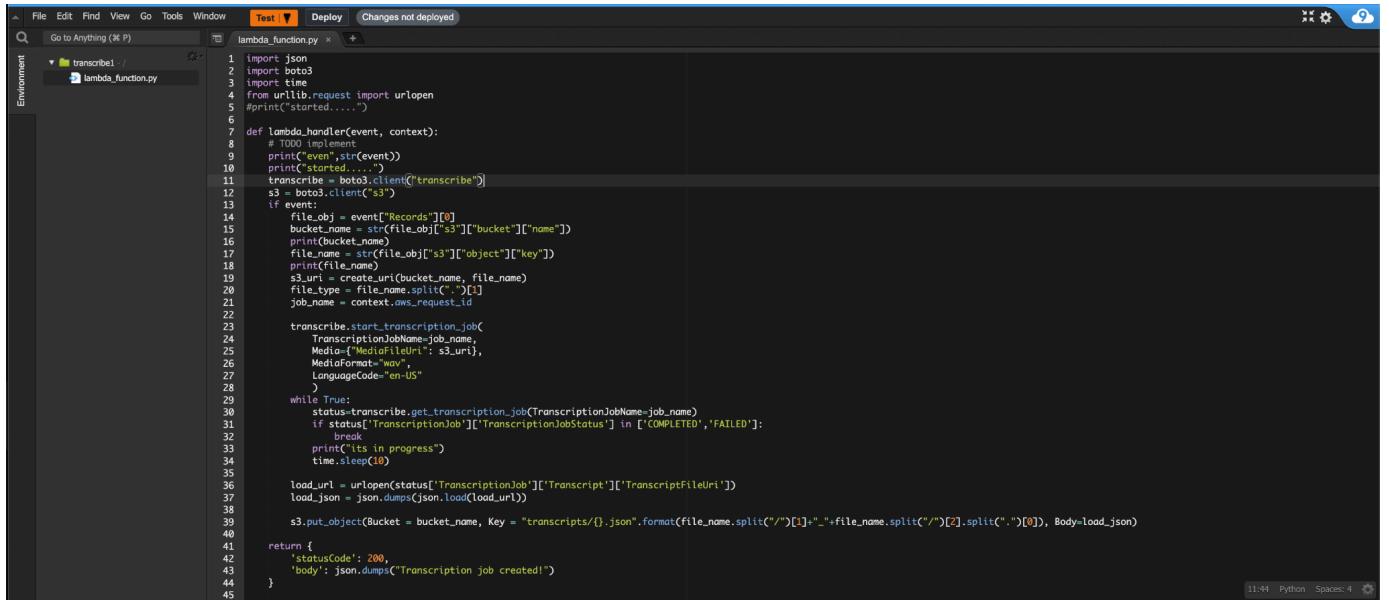
The screenshot shows the AWS S3 console interface. On the left, there's a sidebar with various services like 'Buckets', 'Access Points', 'Object Lambda Access Points', etc. The main area is titled 'transcripts/' and shows a list of objects. At the top of the list table, there are buttons for 'Upload' and a search bar. The table columns are 'Name', 'Type', 'Last modified', 'Size', and 'Storage class'. There are 39 objects listed, all of which are 'Agent1\_<some\_id>.json' files, each being a JSON file of type 'json'. The last object listed is 'Agent1\_6070.json'.

Name	Type	Last modified	Size	Storage class
Agent1_001.json	json	May 2, 2022, 18:24:44 (UTC-04:00)	27.9 KB	Standard
Agent1_004.json	json	May 3, 2022, 21:36:13 (UTC-04:00)	16.5 KB	Standard
Agent1_4093.json	json	May 3, 2022, 06:05:38 (UTC-04:00)	725.4 KB	Standard
Agent1_4112.json	json	April 28, 2022, 22:35:35 (UTC-04:00)	589.5 KB	Standard
Agent1_4156_1.json	json	May 2, 2022, 18:45:28 (UTC-04:00)	85.5 KB	Standard
Agent1_4170.json	json	April 28, 2022, 22:52:22 (UTC-04:00)	720.3 KB	Standard
Agent1_6070.json	json	April 28, 2022, 21:50:10 (UTC-04:00)	620.7 KB	Standard

## Sample text file

```
[{"jobName": "531d07db-153c-489b-ae87-194faf0ec412", "accountId": "825406695460", "results": {"transcripts": [{"transcript": "Hello ma'am good morning how can I assist you today? Hi I'm facing an issue with my blender. It's not working after a month of its purchase. Can you please help me with this? Can you please let me know the order already? Sure it is 4356789. Thank you. I am pulling the information regarding the order. Please hold a few minutes. Sure. Happy to hold. I see that the written period is closed for this item. Yes I'm aware of that. But can you please help me? barely use this item and it has stopped working suddenly. Sure. Let me see what I can do over here. I can send you a new product and you can ship the old product back to us through ups. Does that work for you? Yes so absolutely thank you so much for your help. You made my day. Really happy with this resolution. It was a pleasure assisting you today. Is there anything else I can do for you know I'm all good thank you so much for your assistance. You have a good day. Have a good day. Thank you. Bye bye."}]}]
```

## Lambda Function to Trigger Amazon Transcribe



The screenshot shows a code editor interface with a dark theme. At the top, there's a menu bar with options like File, Edit, Find, View, Go, Tools, Window, and a status bar at the bottom right showing "11:44 Python Spaces: 4". The main area displays a Python script named "lambda\_function.py". The code implements a Lambda function for AWS Transcribe. It starts by importing json, boto3, time, and urlopen. It prints a start message and then defines a lambda\_handler function. This function retrieves an event from the trigger, extracts the S3 bucket name and file key, creates a transcription job with the S3 URI, and then loops until the job is completed or failed. Once completed, it reads the transcript URL, converts it to JSON, and stores it in the S3 bucket under a 'comprehend' object. Finally, it returns a success response.

```
File Edit Find View Go Tools Window Test Deploy Changes not deployed
Go to Anything (⌘ P) lambda_function.py +
Environment
transcribe1 / lambda_function.py
1 import json
2 import boto3
3 import time
4 from urllib.request import urlopen
5 #print("started....")
6
7 def lambda_handler(event, context):
8     # TODO implement
9     print("event",str(event))
10    print("started....")
11    transcribe = boto3.client('transcribe')
12    s3 = boto3.client('s3')
13    if event:
14        file_obj = event['Records'][0]
15        bucket_name = str(file_obj['s3']['bucket']['name'])
16        print(bucket_name)
17        file_name = str(file_obj['s3']['object']['key'])
18        print(file_name)
19        s3_uri = create_uri(bucket_name, file_name)
20        file_type = file_name.split('.')[1]
21        job_name = context.aws_request_id
22
23        transcribe.start_transcription_job(
24            TranscriptionJobName=job_name,
25            Media={'MediaFileUri': s3_uri},
26            MediaFormat="wav",
27            LanguageCode="en-US"
28        )
29        while True:
30            status=transcribe.get_transcription_job(TranscriptionJobName=job_name)
31            if status['TranscriptionJob']['TranscriptionJobStatus'] in ['COMPLETED','FAILED']:
32                break
33            print("its in progress")
34            time.sleep(10)
35
36        load_url = urlopen(status['TranscriptionJob']['Transcript']['TranscriptFileUri'])
37        load_json = json.dumps(json.load(load_url))
38
39        s3.put_object(Bucket = bucket_name, Key = "transcripts/{}.json".format(file_name.split('/')[1] + "_" + file_name.split('/')[2].split(".")[0]), Body=load_json)
40
41    return {
42        'statusCode': 200,
43        'body': json.dumps('Transcription job created!')
44    }
45
```

## Amazon Comprehend for Sentimental Analysis

After the transcribed file is created, another lambda function is triggered to start the comprehend job. Amazon Comprehend is used for sentimental analysis of the transcribed file and the results are stored in the S3 bucket in .json format under 'comprehend' object

The screenshot shows the AWS Amazon S3 console. On the left, there's a sidebar with various options like Buckets, Storage Lens, and Feature spotlight. The main area is titled 'comprehend/' and shows a list of objects. The 'Objects' tab is selected. There are 51 objects listed, including '0.json', '2017-Oct-26-AMZN.txt', '2018-Jul-26-AMZN.json', '4000bb34-c9b2-49a5-ae5d-72cbf48f95f8.json', '4065.json', and '757a8883-4af8-4cb5-ac54-bc5f81851d9.json'. The objects are sorted by name.

## Lambda Function to Trigger Amazon Comprehend

```

1 #!/usr/bin/python
2
3 import json
4 import boto3
5
6 def lambda_handler(event, context):
7     print(event)
8     print('event['s3']')
9     s3 = boto3.client('s3')
10    file_obj = event['Records'][0]['s3']
11    bucket = str(file_obj['bucket']['name'])
12    key = str(file_obj['object']['key'])
13    print(key)
14    file = s3.get_object(Bucket=bucket, Key=key)
15    para = file['Body'].read()
16    para_json.loads(para)
17    print(para)
18    #json.load = json.loads(para)
19    #for i in range(0, len(para)):
20    #print(i)
21    if len(para['results']['transcripts']) > 0:
22        paragraph = para['results']['transcripts'][0].get('transcript')
23        print(paragraph)
24
25        #paragraph['results']['transcripts'][0]['transcript']
26        print(paragraph)
27        para['results']['items']
28        print(para)
29        dur=para['results']['items']
30        dur=para['results']['items']
31        for i in range(0, len(dur)):
32            if dur[i].get('end_time'):
33                n = dur[i].get('end_time')
34                break
35        print('time',round(float(n)/60,2))
36
37        comprehend = boto3.client('comprehend')
38        sentiment = comprehend.batch_detect_sentiment(TextList=[datachunk(paragraph)], LanguageCode="en")
39        max_key = max(datachunk(paragraph), key=datachunk(paragraph).get)
40        final = datachunk(max_key)

```

## DynamoDb Table

The results of the comprehend job are stored in DynamoDb tables

AWS Services Search for services, features, blogs, docs, and more [Option+S] N. Virginia Kashyap Datta Kovvali

**DynamoDB** X Items returned (40)

Dashboard Tables Update settings Explore items PartQL editor New Backups Exports to S3 Reserved capacity

▼ DAX Clusters Subnet groups Parameter groups Events

Tell us what you think Return to the previous console experience Density settings

Feedback Looking for language selection? Find it in the new Unified Settings © 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

	partition...	Agent	Duration	Mixed	Negative	Neutral	Positive	Sentiment
Agent5_69...	Agent5	3.13	0.0005281...	0.1301430...	0.5359882...	0.3337213...	Neutral	
Agent2_002	Agent2	2.01	0.4192600...	0.2791180...	0.1991618...	0.1032102...	Mixed	
Agent7_69...	Agent7	3.72	0.1028493...	0.0648359...	0.2051488...	0.6275120...	Positive	
Agent1_4093	Agent1	30	0.0414408...	0.3059719...	0.4727717...	0.1798558...	Neutral	
Agent1_4112	Agent1	29.2	0.0828700...	0.2665969...	0.3405159...	0.3100655...	Neutral	
Agent7_69...	Agent7	3.66	0.0997970...	0.2956887...	0.0780446...	0.5268009...	Positive	
Agent3_004	Agent3	0.9	0.0004506...	0.0004987...	0.0527235...	0.9480968...	Positive	
Agent2_4077	Agent2	29.5	0.0527205...	0.4400082...	0.3270637...	0.1802493...	Negative	
Agent8_68...	Agent8	3.23	0.0006298...	0.0236386...	0.5527685...	0.4233947...	Neutral	
Agent4_4156	Agent4	23.37	0.2155790...	0.3472274...	0.2520953...	0.1851480...	Negative	
Agent8_68...	Agent8	2.92	0.0024313...	0.4619277...	0.4915968...	0.0445084...	Neutral	
Agent3_003	Agent3	1	0.0793610...	0.5286007...	0.3166761...	0.0768613...	Negative	
Agent6_68...	Agent6	3.18	0.0083754...	0.6353543...	0.3313363...	0.0253063...	Negative	
Agent5_68...	Agent5	3.35	0.0090158...	0.1111063...	0.2725491...	0.6077179...	Positive	

## Athena

The results in DynamoDb are visualized in AWS Quicksight via the Athena connectors. Table items are returned in Athena tables through SQL queries.

AWS Services Search for services, features, blogs, docs, and more [Option+S] N. Virginia Kashyap Datta Kovvali

sentiment1 sentiment2 Views (0) Run again Cancel Save Clear Create

Completed Time in queue: 0.675 sec Run time: 5.793 sec Data scanned: 4.98 KB

Results (43) Copy Download results

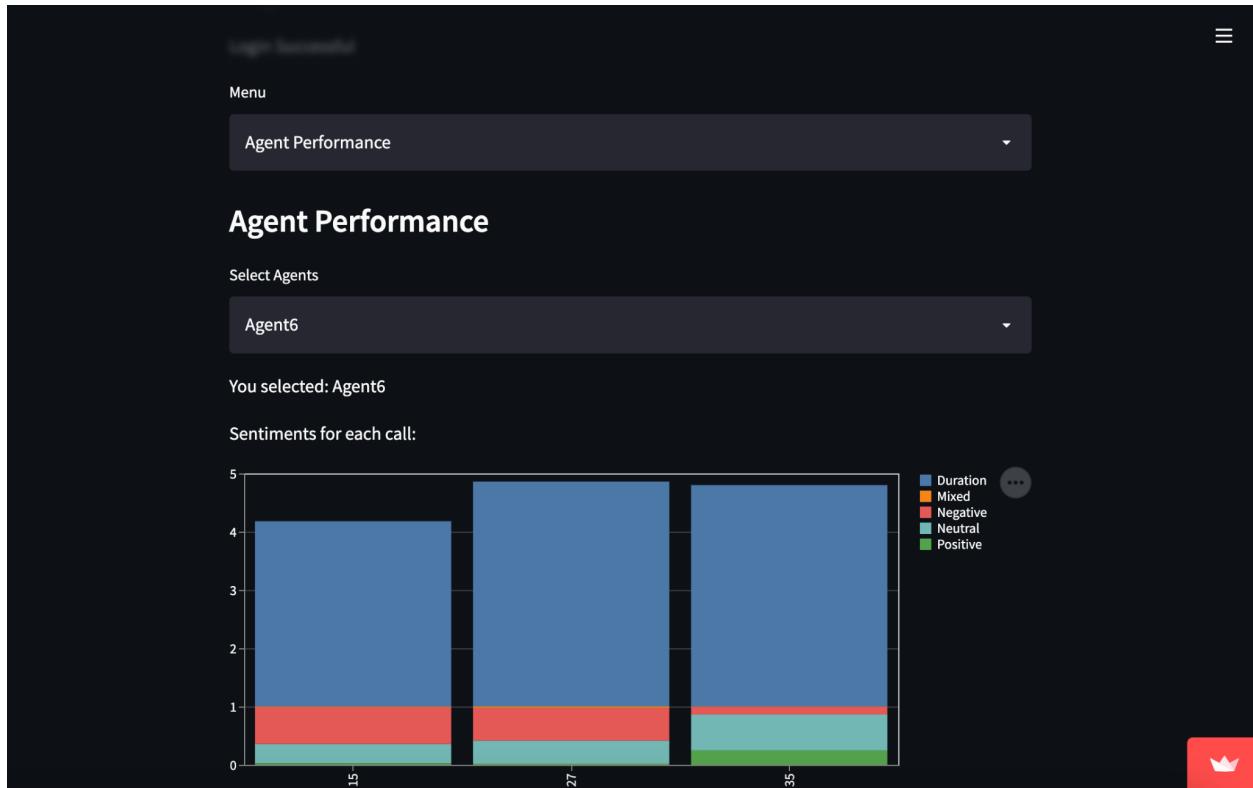
# neutral agent negative partition\_key mixed positive duration sentiment

1	0.147457276	Agent4	0.411638248	Agent4_4104	0.061696333	0.379270681	30.000000000	Negative
2	0.273601817	Agent4	0.579975452	Agent4_0638	0.042789175	0.103722542	14.990000000	Negative
3	0.181473111	Agent9	0.018464579	Agent9_6880_1	0.004427046	0.796053435	3.410000000	Positive
4	0.535988275	Agent5	0.130143026	Agent5_6912_1	0.000528146	0.333721357	3.130000000	Neutral
5	0.199161815	Agent2	0.279118083	Agent2_002	0.419260018	0.103210273	2.040000000	Mixed
6	0.205148813	Agent7	0.064835974	Agent7_6912_6	0.102849383	0.627512030	3.720000000	Positive
7	0.472771734	Agent1	0.305971962	Agent1_4093	0.041440820	0.179855880	30.000000000	Neutral
8	0.340515937	Agent1	0.266596982	Agent1_4112	0.082870079	0.310065556	29.990000000	Neutral
9	0.078044637	Agent7	0.295688733	Agent7_6912_7	0.099797079	0.526800961	3.660000000	Positive
10	0.052723561	Agent3	0.000498763	Agent3_004	0.000450614	0.948096885	0.680000000	Positive
11	0.327063708	Agent2	0.440008296	Agent2_4077	0.052720581	0.180249318	29.990000000	Negative
12	0.552768502	Agent8	0.037678661	Agent8_6880_2	0.009020849	0.423704785	7.270000000	Neutral

Feedback Looking for language selection? Find it in the new Unified Settings © 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

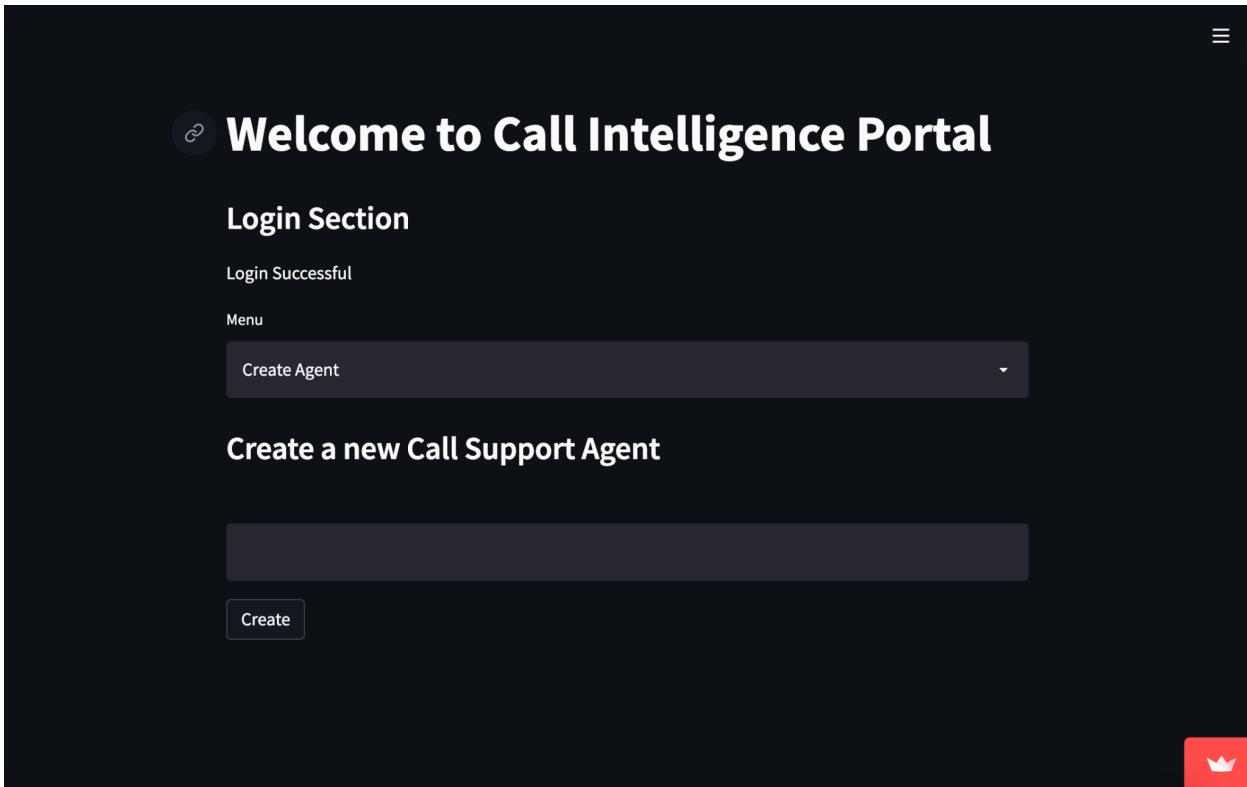
## Agent Performance

The admin will be able to view each agent's performance based on the duration of the call, sentiments for each call - mixed, neutral, positive or negative



## Create Agent

Admin will be able to create a new agent



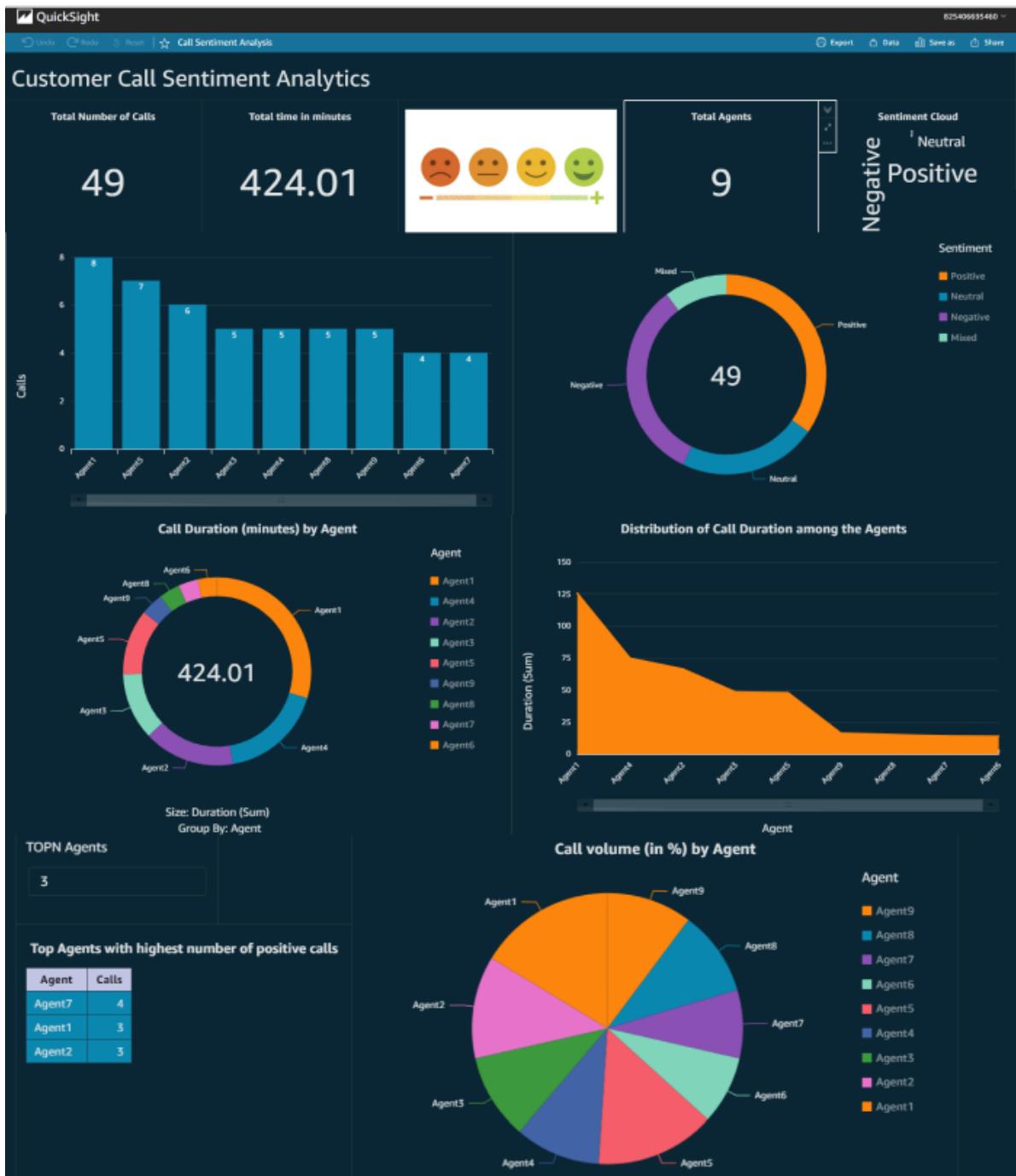
## QuickSight access

The marketing manager and the customer service manager will be given access to the quicksight dashboard to view the analytics. They need to follow the below steps to access the dashboard via the link:

<https://us-east-1.quicksight.aws.amazon.com/sn/dashboards/aab311a1-27ba-40fc-80ba-532ddd1c8dd2>

1. Create a quicksight account once the email with the invitation is received.
2. account name: bigdata-team7, make sure the correct account name is passed during login.

### 3. View the interactive dashboard.



# Testing



## Unit Testing with pytest

For unit testing, pytest framework has been used. The pytest framework makes it easy to write small, readable tests

To Install Pytest run the following command in your command line -

- *pip install -U pytest*

A new file called test\_pytest has been created, containing functions to test the code.

```
❶ test_pytest.py > ...
1  from dynamotodf import checkfile,getAgentData,dbtodf
2  from userlogin import login,signup
3  import pandas as pd
4  data = dbtodf()
5
6
7  def test_checkfile():
8
9      assert checkfile(dbtodf(),'Agent0_9').empty == True
10     assert checkfile(dbtodf(),'').empty == True
11     assert checkfile(dbtodf(),'Agent1_001').equals(data.loc[data['partition_key'] == 'Agent1_001'])
12     assert checkfile(dbtodf(),'Agent1_001').equals(data.loc[data['partition_key'] == 'Agent1_001']) == True
13     assert checkfile(dbtodf(),'Agent3_4074').equals(data.loc[data['partition_key'] == 'Agent3_4157']) == False
14
15
16  def test_getAgentData():
17      agent_df,sentimentcountdf = getAgentData(data,'Agent2')
18      assert agent_df.equals(data.loc[data['Agent']=='Agent2']) == True
19      assert sentimentcountdf.equals(data.loc[data['Agent']=='Agent2']['Sentiment'].value_counts()) == True
20      agent_df,sentimentcountdf = getAgentData(data,'Agent5')
21      assert agent_df.equals(data.loc[data['Agent']=='Agent3']) == False
22      assert sentimentcountdf.equals(data.loc[data['Agent']=='Agent3']['Sentiment'].value_counts()) == False
23
24
25  def test_fileformat():
26      filename = "3123.txt"
27      wavfilename = "3123.wav"
28      assert (filename.rsplit( ".", 1 )[ 1 ] == "wav") == False
29      assert (wavfilename.rsplit( ".", 1 )[ 1 ] == "wav") == True
30
31  def test_login():
32      assert login('knjbnmbhj','sdhjb436') == 400
33      assert login('','') == 400
34      assert login('test12345*','Test12345*') == 200
35
36  def test_signup():
37      assert signup('scsjj','ksjsh') == 400
38      assert signup('Pytest004*','Pytest004*') == 200
```

Run the test using the following command in your command line -

- `pytest test_pytest.py`

Running the tests will give us the following results

```
(finalprojenv) C:\Users\16178\Downloads\FinalProject_Streamlit>coverage run -m pytest test_pytest.py
===== test session starts =====
platform win32 -- Python 3.10.1, pytest-7.1.2, pluggy-1.0.0
rootdir: C:\Users\16178\Downloads\FinalProject_Streamlit
collected 5 items

test_pytest.py .... [100%]

===== 5 passed in 7.20s =====
```

### **Code Coverage with coverage.py**

For measuring the coverage of the tests, we have used coverage.py package. coverage.py uses the code analysis tools and tracing hooks provided in the Python standard library to determine which lines are executable, and which have been executed.

To Install coverage.py run the following command in your command line -

- `pip install coverage`

Use coverage run to run the test suite and gather data in command line as below -

- `coverage run -m pytest test_pytest.py`

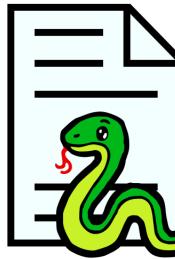
Use coverage report to get the report of the test -

- `coverage report -m`

Running the report will give us the following result

```
(finalprojenv) C:\Users\16178\Downloads\FinalProject_Streamlit>coverage report -m
  Name      Stmts   Miss  Cover   Missing
-----
dynamotodf.py      23      0  100%
test_pytest.py     29      0  100%
userlogin.py       16      0  100%
-----
TOTAL              68      0  100%
```

## Documentation



Code documentation is crucial for any project. It increases code maintainability, helps troubleshoot production issues, help manage any additional integrations and is good for knowledge transfer.

We have created a documentation for streamlit code using the pdoc package and hosted it using Github Pages for public access.

Documentation Link: <https://kashyap-datta.github.io/callanalyticsdocumentation/>