CONTENTS

Objective ……………………………………….….2

User Profiles ……………………………………....4

Usage Summary ……………………………….….5

Requirement Summary ……………………….… 6

Success Criteria ……………………………..…….7

Design Goals …………………………..…………..8

Work Flow Diagram ……………….……………10

User Case Diagram …………………….………..11

Data Flow Diagram …………………..………….12

Languages Used …………………………..……...13

Codes ……………………………………….…….14

-Greetings Export ------------------------------14

-WhatsYourName Export --------------------17

-Function Export -------------------------------19

-Event Export -----------------------------------20

-Goodbye Export -------------------------------23

-Count Intent Lambda Function ------------25

Conclusions…………………………………… .29

Bibliography……………………………………30

OBJECTIVE

The objective of this project is to create a bot for a college campus which helps in answering the queries related to the campus.

The queries that we have dealt, are, events and teachers. The event function is to answer the user about the upcoming events and also the events that have ended.

The teachers queries allows the user to answer the queries of who is the teacher of a particular subject and vice versa.

Bot is a virtual helper and chatting with a bot feels like chatting with a real person, real time.

All the data used for dotty is stored in cloud.

High level solution architecture

USER PROFILES

ADMIN

Adds new information and deletes old and redundant information from the s3.

The work of an admin is to constantly add in new events and any change in events and teachers. The admin will have an IAM role of *Admin Access.*

USER

The user of a dotty has the access to ask question in a chat box format. The queries asked by the user is processed by *natural language processing technique* which helps in recognizing the *intent* of the user.

USAGE SUMMARY

The usage scenarios of dotty is given below

Read data from s3 bucket via lambda function

Add/delete/read event/teacher

REQUIREMENTS SUMMARY

*Business Requirements*

The business goal for this application is to increase the student-admin interaction and make the information on the campus readily to the student to increase the co curricular activity of the student.

*Admin Requirement*

The requirements of an admin are the following:

1. To understand the trends in the query
2. To monitor the error rate and the percentage of correct responses given in a set of response.
3. To help identify the intent of a query given by the user
4. Monitor databases (s3 buckets)
5. Update the databases to have the recent information

*Hardware Requirement*

The only specific hardware requirement is the constant internet support.

SUCCESS CRITERIA

To determine the success of this project, the following metrics can be quantified and used to analyze success factors:-

1. The error percentage
2. The percentage of correct responses given in a set of 100 queries
3. The percentage of time the bot could not recognize the intent of the user.
4. The amount of build failures occurred in a given time frame.

Design Goals and Constraints

*Performance*

No more than a 5-percent degradation in average query response is allowed while all concurrent user are using the system.

*Processor utilization*

Should not exceed 80 percent during all concurrent users are using the system.

*Availability*

Because the system is accessed by admin to update all the databases, and their should not be any single point of failure.

*Reliability*

Because of the need no single point failure, automatic failover will be required. In addition, existing disaster recovery and backup plans and procedures must be revised to incorporate Dotty.

*Scalability*

Aws allows 100’s of terra bytes of data to be stored in the cloud, and the lambda allows 75GB of code, which helps us in creating as many intents as we want.

*Security*

For the sensitive customer and order data, all IAM users will need to log on the system with their user id and password.

Every resource in the system are defined by the role and privileged. System administrator assigned user role and privileged for their access rights.

*Interoperability*

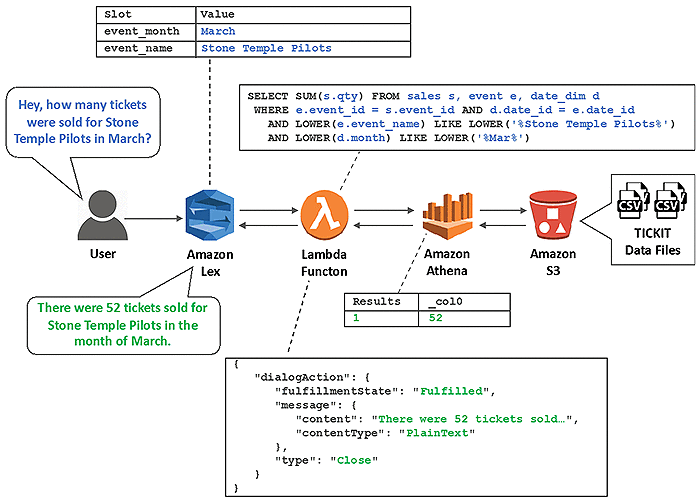
In Version 1.0 of the Dotty, there are no requirements for interoperability with other systems.

*Location*

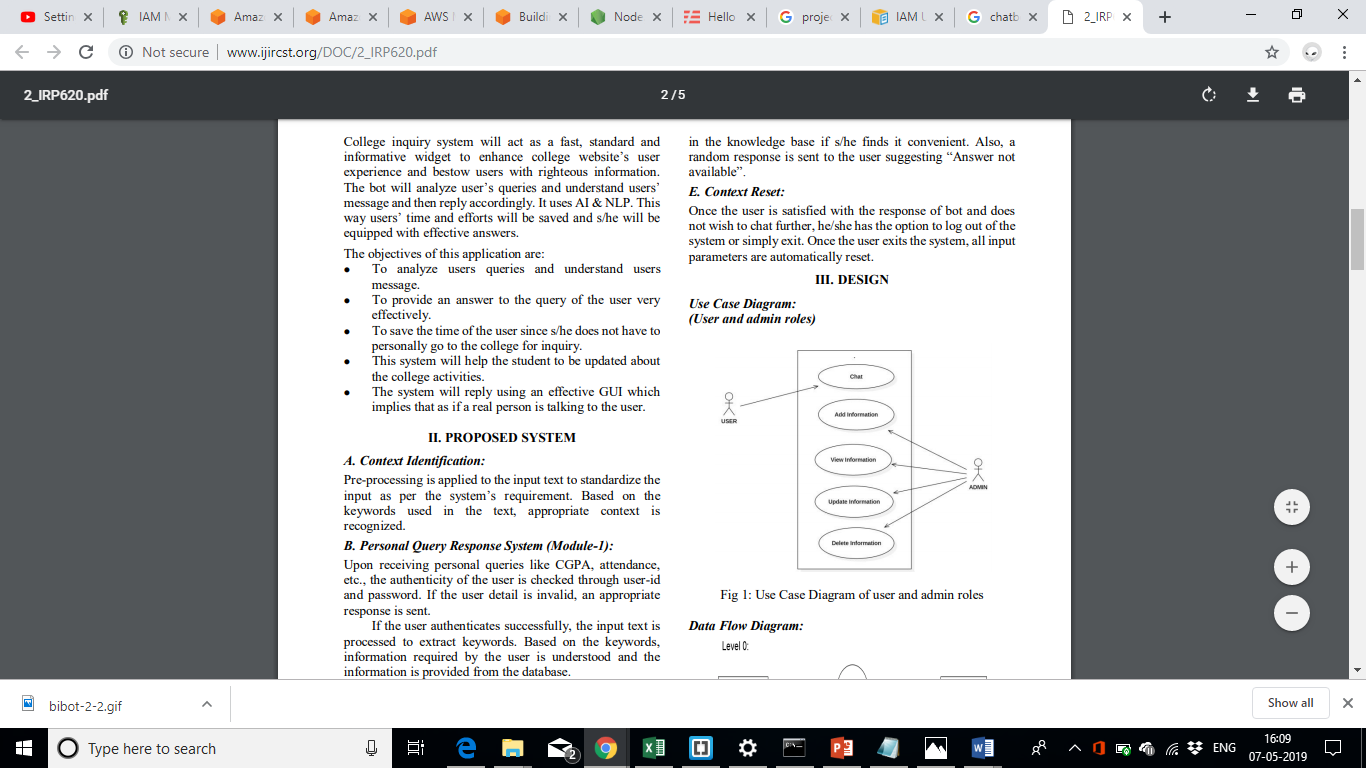
As dotty is based on the cloud, the operations on it can be done from anywhere in the world and is readily available for use anywhere and anytime.

WORK FLOW DIAGRAM

The work flow diagram with integration of Lex, Lambda and S3 is given below:

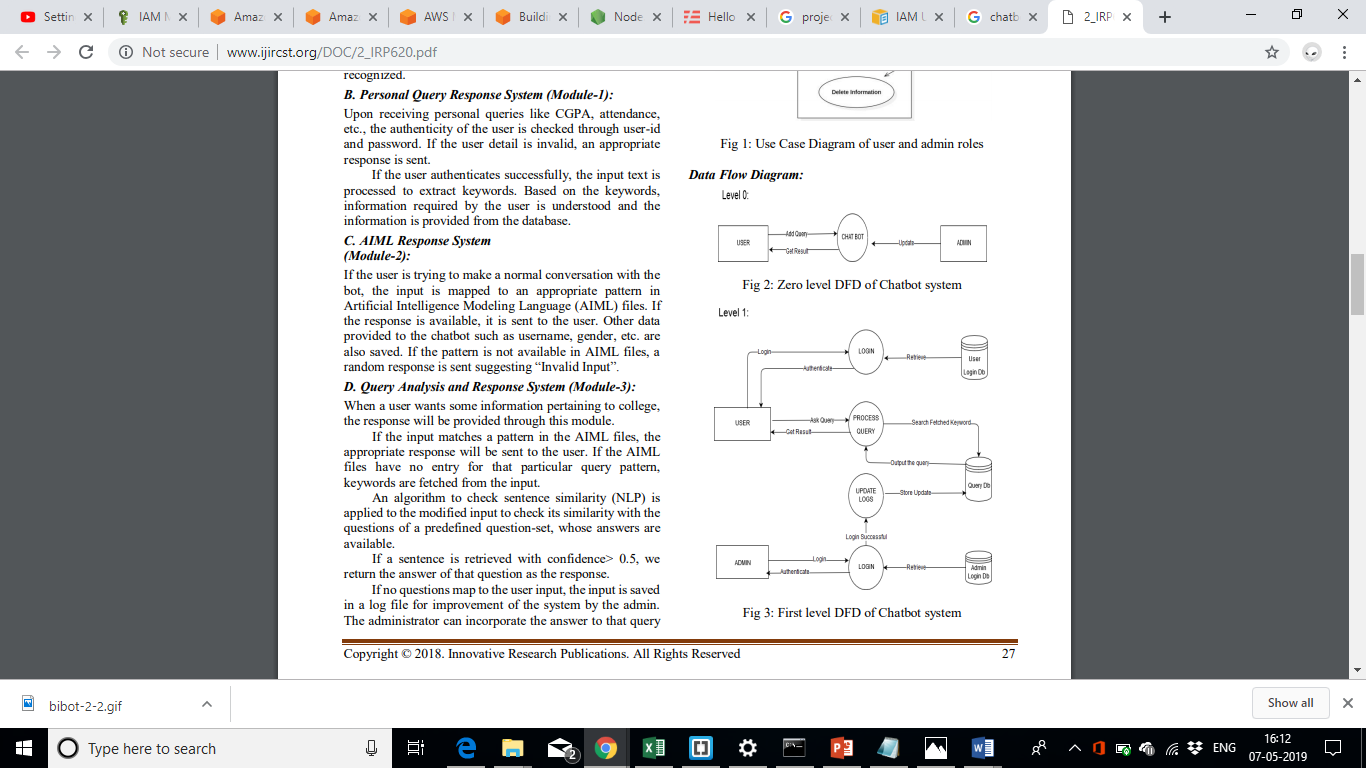


USE CASE DIAGRAM

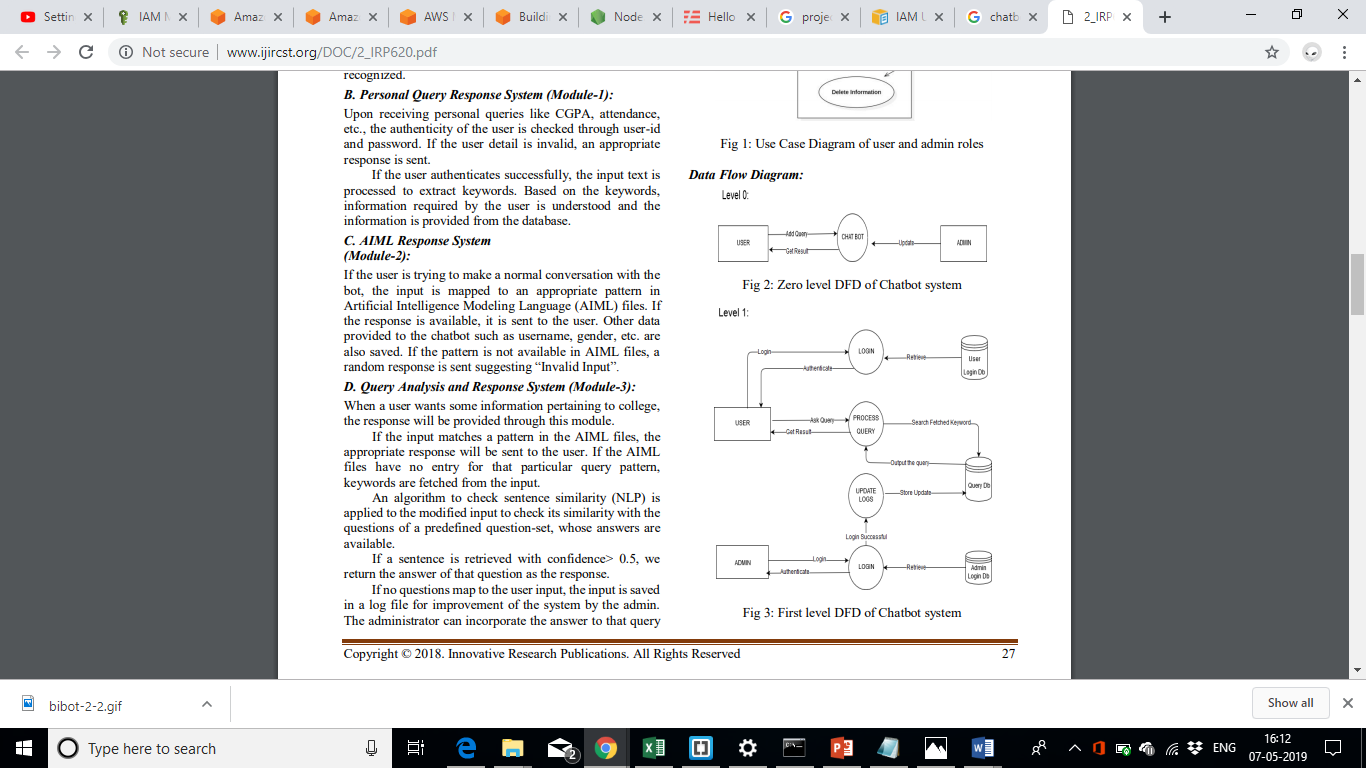


DATA FLOW DIAGRAM

*Level 0*



*Level 1*



LANGUAGES USED

The languages used in the project are:

For the Intents

Json

For the lambda function

SQL via Athena

Python

CODES

*GREETINGS EXPORT*

|  |
| --- |
| { |
|  | "metadata": { |
|  | "schemaVersion": "1.0", |
|  | "importType": "LEX", |
|  | "importFormat": "JSON" |
|  | }, |
|  | "resource": { |
|  | "name": "greeting", |
|  | "version": "3", |
|  | "fulfillmentActivity": { |
|  | "type": "ReturnIntent" |
|  | }, |
|  | "sampleUtterances": [ |
|  | "Hello", |
|  | "Hi", |
|  | "hey", |
|  | "heys", |
|  | "heya", |
|  | "namaste" |
|  | ], |
|  | "slots": [ |
|  | { |
|  | "sampleUtterances": [], |
|  | "slotType": "AMAZON.US\_FIRST\_NAME", |
|  | "slotConstraint": "Required", |
|  | "valueElicitationPrompt": { |
|  | "messages": [ |
|  | { |
|  | "contentType": "PlainText", |
|  | "content": "What is your name?" |
|  | } |
|  | ], |
|  | "maxAttempts": 2 |
|  | }, |
|  | "priority": 1, |
|  | "name": "studName" |
|  | } |
|  | ], |
|  | "conclusionStatement": { |
|  | "messages": [ |
|  | { |
|  | "groupNumber": 1, |
|  | "contentType": "PlainText", |
|  | "content": "Hello {studName}" |
|  | }, |
|  | { |
|  | "groupNumber": 1, |
|  | "contentType": "PlainText", |
|  | "content": "Hi there {studName}" |
|  | }, |
|  | { |
|  | "groupNumber": 1, |
|  | "contentType": "PlainText", |
|  | "content": "Namaste {studName}" |
|  | }, |
|  | { |
|  | "groupNumber": 1, |
|  | "contentType": "PlainText", |
|  | "content": "Hola Amigo {studName}" |
|  | }, |
|  | { |
|  | "groupNumber": 1, |
|  | "contentType": "PlainText", |
|  | "content": "Hola {studName}" |
|  | } |
|  | ] |
|  | }, |
|  | "slotTypes": [] |
|  | } |
|  | } |

*WHAT IS YOUR NAME EXPORT*

|  |
| --- |
| { |
|  | "metadata": { |
|  | "schemaVersion": "1.0", |
|  | "importType": "LEX", |
|  | "importFormat": "JSON" |
|  | }, |
|  | "resource": { |
|  | "name": "whatsyourname", |
|  | "version": "3", |
|  | "fulfillmentActivity": { |
|  | "type": "ReturnIntent" |
|  | }, |
|  | "sampleUtterances": [ |
|  | "what is your name", |
|  | "what are you called" |
|  | ], |
|  | "slots": [], |
|  | "conclusionStatement": { |
|  | "messages": [ |
|  | { |
|  | "groupNumber": 1, |
|  | "contentType": "PlainText", |
|  | "content": "The name is Bot, Dotty Bot" |
|  | }, |
|  | { |
|  | "groupNumber": 1, |
|  | "contentType": "PlainText", |
|  | "content": "My name is Dotty" |
|  | }, |
|  | { |
|  | "groupNumber": 1, |
|  | "contentType": "PlainText", |
|  | "content": "The name is Dotty" |
|  | }, |
|  | { |
|  | "groupNumber": 1, |
|  | "contentType": "PlainText", |
|  | "content": "Log mujhe Dotty kehte hai" |
|  | }, |
|  | { |
|  | "groupNumber": 1, |
|  | "contentType": "PlainText", |
|  | "content": "People call me Dotty" |
|  | } |
|  | ] |
|  | }, |
|  | "slotTypes": [] |
|  | } |
|  | } |

*FUNCTION EXPORT*

|  |
| --- |
| { |
|  | "metadata": { |
|  | "schemaVersion": "1.0", |
|  | "importType": "LEX", |
|  | "importFormat": "JSON" |
|  | }, |
|  | "resource": { |
|  | "name": "function", |
|  | "version": "2", |
|  | "fulfillmentActivity": { |
|  | "type": "ReturnIntent" |
|  | }, |
|  | "sampleUtterances": [ |
|  | "What can you do", |
|  | "help", |
|  | "function", |
|  | "what are your feature", |
|  | "kya kar skte ho tum" |
|  | ], |
|  | "slots": [], |
|  | "conclusionStatement": { |
|  | "messages": [ |
|  | { |
|  | "groupNumber": 1, |
|  | "contentType": "PlainText", |
|  | "content": "I can give you event details, holiday details and teacher details. What would you like to do?" |
|  | } |
|  | ] |
|  | }, |
|  | "slotTypes": [] |
|  | } |
|  | } |

*EVENTS EXPORT*

|  |
| --- |
| { |
|  | "metadata": { |
|  | "schemaVersion": "1.0", |
|  | "importType": "LEX", |
|  | "importFormat": "JSON" |
|  | }, |
|  | "resource": { |
|  | "name": "events", |
|  | "version": "4", |
|  | "fulfillmentActivity": { |
|  | "type": "ReturnIntent" |
|  | }, |
|  | "sampleUtterances": [ |
|  | "event date", |
|  | "when is my event", |
|  | "i want details of an event", |
|  | "event detail", |
|  | "events", |
|  | "when is {eventName}", |
|  | "{eventName} details" |
|  | ], |
|  | "slots": [ |
|  | { |
|  | "sampleUtterances": [], |
|  | "slotType": "events\_dotty", |
|  | "slotTypeVersion": "2", |
|  | "slotConstraint": "Required", |
|  | "valueElicitationPrompt": { |
|  | "messages": [ |
|  | { |
|  | "contentType": "PlainText", |
|  | "content": "What is the name of the event?" |
|  | } |
|  | ], |
|  | "maxAttempts": 2 |
|  | }, |
|  | "priority": 1, |
|  | "name": "eventName" |
|  | } |
|  | ], |
|  | "conclusionStatement": { |
|  | "messages": [ |
|  | { |
|  | "groupNumber": 1, |
|  | "contentType": "PlainText", |
|  | "content": "I will check and say" |
|  | } |
|  | ] |
|  | }, |
|  | "slotTypes": [ |
|  | { |
|  | "description": "Names of events in TICKIT database", |
|  | "name": "events\_dotty", |
|  | "version": "2", |
|  | "enumerationValues": [ |
|  | { |
|  | "value": "mid semester", |
|  | "synonyms": [ |
|  | "mid sem" |
|  | ] |
|  | }, |
|  | { |
|  | "value": "abstrosonic", |
|  | "synonyms": [ |
|  | "abstronic" |
|  | ] |
|  | }, |
|  | { |
|  | "value": "sample", |
|  | "synonyms": [ |
|  | "sammple" |
|  | ] |
|  | }, |
|  | { |
|  | "value": "karwaan", |
|  | "synonyms": [ |
|  | "karvan" |
|  | ] |
|  | } |
|  | ], |
|  | "valueSelectionStrategy": "TOP\_RESOLUTION" |
|  | } |
|  | ] |
|  | } |
|  | } |

*GOODBYE EXPORT*

|  |
| --- |
| { |
|  | "metadata": { |
|  | "schemaVersion": "1.0", |
|  | "importType": "LEX", |
|  | "importFormat": "JSON" |
|  | }, |
|  | "resource": { |
|  | "name": "Goodbye", |
|  | "version": "1", |
|  | "fulfillmentActivity": { |
|  | "type": "ReturnIntent" |
|  | }, |
|  | "sampleUtterances": [ |
|  | "bye", |
|  | "thankyou", |
|  | "i am done", |
|  | "okay bye", |
|  | "bye bye", |
|  | "goodbye", |
|  | "i am finished", |
|  | "all set", |
|  | "ok thanks bro", |
|  | "thanks", |
|  | "k bye" |
|  | ], |
|  | "slots": [], |
|  | "conclusionStatement": { |
|  | "messages": [ |
|  | { |
|  | "groupNumber": 1, |
|  | "contentType": "PlainText", |
|  | "content": "Bye Bye" |
|  | }, |
|  | { |
|  | "groupNumber": 1, |
|  | "contentType": "PlainText", |
|  | "content": "bye" |
|  | }, |
|  | { |
|  | "groupNumber": 1, |
|  | "contentType": "PlainText", |
|  | "content": "Thankyou. Have a good day." |
|  | }, |
|  | { |
|  | "groupNumber": 1, |
|  | "contentType": "PlainText", |
|  | "content": "Have a good day." |
|  | }, |
|  | { |
|  | "groupNumber": 1, |
|  | "contentType": "PlainText", |
|  | "content": "See you later." |
|  | } |
|  | ] |
|  | }, |
|  | "slotTypes": [] |
|  | } |
|  | } |

*COUNT INTENT LAMBDA FUNCTION*

import time

import logging

import json

import dotty\_config as dotty

import dotty\_helpers as helpers

import dotty\_userexits as userexits

# SELECT statement for Count query

COUNT\_SELECT = "SELECT SUM(s.qty) FROM sales s, event e, venue v, category c, date\_dim d "

COUNT\_JOIN = " WHERE e.event\_id = s.event\_id AND v.venue\_id = e.venue\_id AND c.cat\_id = e.cat\_id AND d.date\_id = e.date\_id "

COUNT\_WHERE = " AND LOWER({}) LIKE LOWER('%{}%') "

COUNT\_PHRASE = 'tickets sold'

logger = logging.getLogger()

logger.setLevel(logging.DEBUG)

def lambda\_handler(event, context):

logger.debug('<<Dotty>> Lex event info = ' + json.dumps(event))

config\_error = helpers.get\_dotty\_config()

session\_attributes = event['sessionAttributes']

logger.debug('<<Dotty>> lambda\_handler: session\_attributes = ' + json.dumps(session\_attributes))

if config\_error is not None:

return helpers.close(session\_attributes, 'Fulfilled',

{'contentType': 'PlainText', 'content': config\_error})

else:

return count\_intent\_handler(event, session\_attributes)

def count\_intent\_handler(intent\_request, session\_attributes):

method\_start = time.perf\_counter()

logger.debug('<<Dotty>> count\_intent\_handler: intent\_request = ' + json.dumps(intent\_request))

logger.debug('<<Dotty>> count\_intent\_handler: session\_attributes = ' + json.dumps(session\_attributes))

session\_attributes['greetingCount'] = '1'

session\_attributes['resetCount'] = '0'

session\_attributes['finishedCount'] = '0'

session\_attributes['lastIntent'] = 'Count\_Intent'

# Retrieve slot values from the current request

slot\_values = session\_attributes.get('slot\_values')

try:

slot\_values = helpers.get\_slot\_values(slot\_values, intent\_request)

except dotty.SlotError as err:

return helpers.close(session\_attributes, 'Fulfilled', {'contentType': 'PlainText','content': str(err)})

logger.debug('<<Dotty>> "count\_intent\_handler(): slot\_values: %s', slot\_values)

# Retrieve "remembered" slot values from session attributes

slot\_values = helpers.get\_remembered\_slot\_values(slot\_values, session\_attributes)

logger.debug('<<Dotty>> "count\_intent\_handler(): slot\_values afer get\_remembered\_slot\_values: %s', slot\_values)

# Remember updated slot values

helpers.remember\_slot\_values(slot\_values, session\_attributes)

# build and execute query

select\_clause = COUNT\_SELECT

where\_clause = COUNT\_JOIN

for dimension in dotty.DIMENSIONS:

slot\_key = dotty.DIMENSIONS.get(dimension).get('slot')

if slot\_values[slot\_key] is not None:

value = userexits.pre\_process\_query\_value(slot\_key, slot\_values[slot\_key])

where\_clause += COUNT\_WHERE.format(dotty.DIMENSIONS.get(dimension).get('column'), value)

query\_string = select\_clause + where\_clause

response = helpers.execute\_athena\_query(query\_string)

result = response['ResultSet']['Rows'][1]['Data'][0]

if result:

count = result['VarCharValue']

else:

count = 0

logger.debug('<<Dotty>> "Count value is: %s' % count)

# build response string

if count == 0:

response\_string = 'There were no {}'.format(COUNT\_PHRASE)

else:

response\_string = 'There were {} {}'.format(count, COUNT\_PHRASE)

# add the English versions of the WHERE clauses

for dimension in dotty.DIMENSIONS:

slot\_key = dotty.DIMENSIONS[dimension].get('slot')

logger.debug('<<Dotty>> pre top5\_formatter[%s] = %s', slot\_key, slot\_values.get(slot\_key))

if slot\_values.get(slot\_key) is not None:

# the DIMENSION\_FORMATTERS perform a post-process functions and then format the output

# Example: {... 'venue\_state': {'format': ' in the state of {}', 'function': get\_state\_name}, ...}

if userexits.DIMENSION\_FORMATTERS.get(slot\_key) is not None:

output\_text = userexits.DIMENSION\_FORMATTERS[slot\_key]['function'](slot\_values.get(slot\_key))

response\_string += ' ' + userexits.DIMENSION\_FORMATTERS[slot\_key]['format'].lower().format(output\_text)

logger.debug('<<Dotty>> dimension\_formatter[%s] = %s', slot\_key, output\_text)

response\_string += '.'

return helpers.close(session\_attributes, 'Fulfilled', {'contentType': 'PlainText','content': response\_string})

CONCLUSIONS AND FUTURE ENHANCEMENTS

This project was developed to fulfill student requirements and encouraging them to have co curricular activities however there are lots of scope to improve the performance of Dotty in the area of user interface, database performance, functions, intents and query processing time. Etc.

So there are many things for future enhancement of this project. The future enhancements that are possible in the project are as follows.

* Linking of social media account.
* Integration with college website.
* Payment gateway to facilitate online ticket purchasing.
* In the area of data security and system security.
* Online booking of tickets and registration of new users.
* To create more intents like result, student details etc.
* Provide more online tips and help.
* To optimize the query which is embedded in the system.

BIBLIOGRAPHY

[1]

<https://serverless.com/framework/docs/providers/aws/examples/hello-world/python/>

Accessed on 9.03.2019

[2]

<https://www.youtube.com/watch?v=Y2Uw0nCByi4>

Accessed on 23.03.2019

[3]

<https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html#id_users_perms>

Accessed on 23.03.2019

[4]

<https://aws.amazon.com/>

Accessed on 7.05.2019

[5]

<https://console.aws.amazon.com/lambda/home?region=us-east-1#/layers>

Accessed on 7.05.2019