TABLE OF CONTENTS

* Abstract iii
* Acknowledgement iv
* Chapter 1 Introduction 3

1.1 Project Overview 4

1.2 Scope 5

1.3 Objective and Details of Project 5

* Chapter 2 System Analysis 7

2.1 System Information 8

2.2 Tools & Technology 8

* Chapter 3 System Design 9

3.1 Flow of System and Steps for Building LEX 10

* Chapter 4 Snapshots 12
* Chapter 5 Constraints 18
* Chapter 6 Conclusion 20
* References 21

**LIST OF FIGURES**

* Fig 1 What is Chatbot 2
* Fig 2 Flow of Technologies 5
* Fig 3 Example with Flow of technologies 12
* Fig 4 Add Slot to Indent 15
* Fig 5 Create alias and publish bot with Version 22
* Fig 6 Serverless Installation 25
* Fig 7 Working with Serverless 32
* Fig 8 Normal code for Lambda Function 35
* Fig 9 Test bot on LEX window 42
* Fig 10 Deploy Bot in SLACK 43
* Fig 11 Facebook Page where BOT is Added 43
* Fig 12 Reply from BOT in Messenger 43

# **CHAPTER 1: Introduction**

# **Project Overview**

* + 1. **Introduction:**

Amazon(AWS) Lex is a service develop for conversational interfaces in some application by use of voice and text. Amazon (AWS) Lex provides the advanced deep learning functionalities of automatic speech recognition (ASR) for converting from speech to the text format, and natural language understanding (NLU) to identify the intent of the text, to enable you to build applications with better engaging user experiences and act like conversational interactions. With Amazon(AWS) Lex, the same deep learning technologies that backups Amazon(AWS) Alexa are now available to any project makers, enabling you to fast and simple build simple, natural language, conversational bots (“chatbots”).

Natural language understanding is the most challenging problems to resolve in computer science and information technology, needing sophisticated deep learning algorithms to be trained on large amounts of data and infrastructure. Amazon(AWS) Lex uses these deep learning technologies by putting the power of Amazon Alexa within reach of all programers and project makers. Harnessing these technologies, Amazon(AWS) Lex gives acces to you to define entirely latest categories of products that are made possible through conversational interfaces.

Here I have developed a chatbot which can be used in a café or a restaurant. I have provided few basics commands which can be used to test the Bot. Later on, after the bot is completely tested and verified, I have made a facebook page name “Café Botio” and integrated the Chatbot on that facebook page. I ave also integrated it on Slack which is group chat work forum.

# **1.2 Scope:**

We tried to develop a chatbot that can incorporate a general database to provide simple but appropriate responses while having conversation. We need to successfully understand human input and find the most appropriatet keywords with few understanding of the context. A very basic response would be simply stating a fact that is related,

for example: Human: “I live by the bay”. Chatbot: “Crab live in the bay”. However, we hope to achieve a more dynamic response: Human: “I live by the bay”. Chatbot: “Do you Crab?”

Project planning was done to define the scope of the project and estimate and schedule project activities and thereby lay the foundation for the execution, monitoring and control of the project. The main purpose of the chatbot using Amazon(AWS) lex is to communicate to wider range of customers without using the human inter fearnce and to allow user to ask the questions and queries which they would ask to any physical human.

The value of using Amazon(AWS) is to minimize the cost that has been recognized by companies. They realize it's important for their success and their company ' future health. This sector also provides answers to existing business issues and gives an insight into future trends. This trains companies to produce products for the future and aspires to communicate with tomorrow's consumers.

# **1.3 Objective**

Our main goal of the project is to create such a chatbot which will provide us the detailed information about the particular question asked.

This document reports on the design of a "Working with Amazon(AWS) Lex" software created as a team project in an undergraduate software engineering course. The overall goal of the project was to create an chatbot where both utility and usability are addressed. In particular, the software should be easy to navigate and use while providing recommendations and security features. When deployed, the application will serve as a single working “point of management” system that acts as both a terminal for taking orders and a terminal for generating reports and managing changes to various records.

**Details of The Project**

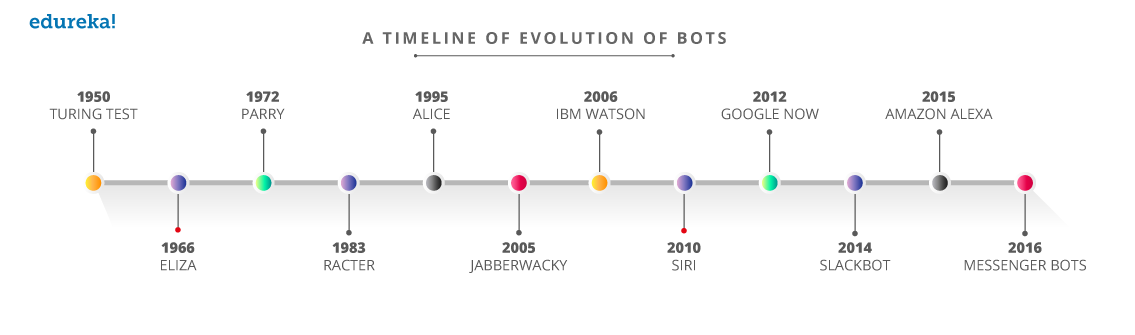
**What is a chatbot?**

*“A chatbot is a computer application that conducts a conversation in simple human language via voice or text inputs, understands the need of the user, and sends a reply appropriate for business rules and data of the organization.”*

In simple terms, chatbot is a service or tool that you can communicate with via chat interface. Chatbot conveys what you are trying to ask and responds with a relevant reply answer or directly completes the asked task you asked for.

### How And When Did Chatbot Evolution Start?

If you have thought chatbots are new technology, then you are wrong. In fact, the first chatbot, **Eliza**, was built in 1966 at the MIT AI Laboratory by Joseph to mimic psychotherapist from only 200 lines of code. Then in 1988, when Rollo Carpenter started the Jabberwacky project a voice-operated entertainment AI chatbot. Here’s how chatbots evolved since then:



## What Is Amazon(AWS) Lex Bot?

Amazon(AWS) Lex is a service develop for conversational interfaces in some application by use of voice and text. Amazon (AWS) Lex provides the advanced deep learning functionalities of automatic speech recognition (ASR) for converting from speech to the text format, and natural language understanding (NLU) to identify the intent of the text, to enable you to build applications with better engaging user experiences and act like conversational interactions. With Amazon(AWS) Lex, the same deep learning technologies that backups Amazon(AWS) Alexa are now available to any project makers, enabling you to fast and simple build simple, natural language, conversational bots (“chatbots”).

### Benefits of using Amazon(AWS) Lex:

* **Simplicity:** It offers an simple console to built your own chatbot in seonds & predefined bots to help you get started.
* **Inbuilt Technologies:** You supply just little example phrases, and Amazon(AWS) Lex builds a pure natural language model through that the bot can communicate using voice and text.
* **Seamless deployment and scaling:** As the user traffic increases, you don’t need to worry about provisioning extra hardware and managing infrastructure to give power to your bot experience.
* **Built-in integration with AWS:** Amazon(AWS) Lex allows integrating with multiple other services on the AWS platform including [AWS Lambda](https://www.edureka.co/blog/aws-lambda-tutorial), [Amazon(AWS) CloudWatch](https://www.edureka.co/blog/amazon-cloudwatch-monitoring-tool/), Amazon(AWS) Cognito, and Amazon(AWS) DynamoDB & many others.
* **Cost-Effective:** With Amazon(AWS) Lex, there are no upfront costs or minimum fees. You pay only for the text or speech requests that you make. No fixed or extra charges.

Let’s consider a use-case to understand the capabilities of Amazon(AWS) Lex.

### Use-Case: To get hospital information through an Amazon(AWS) Lex chatbot.

Using Amazon(AWS) Chatbot, you can build powerful interfaces to use with mobile apps also. You can deploy a voice or text chat conversation to built bots on mobile devices that can help users with basic tasks.

# **CHAPTER 2: SYSTEM ANALYSIS**

# **2.1 System Information**

**Informational Bots**

You can use Amazon(AWS) Lex to develop chatbots for regular consumer requests, such as surfing the latest news updates, updates in scores, or weather. After you develop your Amazon(AWS) Lex bot, you can use them on mobile devices, chat services, and IoT devices, with support for rich and better looking message formatting.

# **2.2 Target Area**

This system can be placed at any website. Nowadays you can see the chat bot available at each and every website but most of them are inefficient. We have placed this Bot in a facebook page where it can receive the order and give the order ID. Regarding to the module that had been identified, the flow of an activity will be described in term of Regards, Asking for Order, Confirming the order after clarified Input, Canceling order if it has negative input.

There is no primary technical knowledge required by the user. The user must be having a facebook acoount and he just need to register the ID at café. And whwnever he visits,his order can be received on the Facebook messenger.

# **2.3 Tools and Technologies used**

Platform: Amazon(AWS) Web Services

Service: Amazon(AWS) LEX

Language: Python/Node.JS used

Database: Dynamodb

# **Chapter 3: System Design and Technologies**

# **Steps for building Amazon (AWS) LEX**

Amazon (AWS) Lex helps you to build applications using a voice or text interface developed by the same technology that is used in Amazon Alexa. These are the simple steps you perform while working with Amazon(AWS) Lex:

Develop a bot and configure it with one or multiple intents which you want to support. Configure the chatbot as if it understands the user's need, engages in interface with the user to provide information, and satisfy the user's need.

Try to implement the bot. You can do the trial in the window client provided in the Amazon(AWS) Lex console.

Publish a update and build an alias.

Integrate the bot. You can Merge the bot on platforms like as mobile apps or messaging platforms such as Twitter Messenger.

Before you get working, get comfortable with the following Amazon(AWS) Lex core concepts and terms:

Bot – A bot does automated tasks like giving order for pizza, book a resort, ordering fruits, and so on. An Amazon(AWS) Lex bot is powered by Automatic Speech Recognition (ASR) and Natural Language Understanding (NLU) capabilities, the same technology that is used in Amazon(AWS) Alexa.

Amazon(AWS) Lex bots can interpret user input provided with textual or voice and use the natural language. You can build Lambda functions and integrate them as code hooks in your deployed configuration to do user data validation and fulfillment tasks.

Intent – An intent represents an deed that the user needs to perform. You develop a chatbot to support one or multiple related intents. For example, you may develop a bot that can order burger and coffee. For each intent, you provide the following required information:

Intent name– A appropriate name for the intent. For example, OrderCoffee.

Sample utterances – How a user would ask the need. For example, a user would say "Can I order a burger please" or "I want to order a burger".

How to fulfill the intent – How you want to satisfactory the need after the user gives the necessary data (for example, place order with a local soda shop). We insist that you create a Lambda function to fulfill the need.

You can optionally configure the need so Amazon (AWS) Lex easily returns the data reply back to the client application to do the necessary fulfillment.

Also to custom needs such as ordering a Burger, Amazon(AWS) Lex also serves built-in intents to quickly make up your bot. For more information, see Built-in Intents and Slot Types.

Slot – An intent can require zero or more slots or parameters. You add slots as part of the intent need. At runtime, Amazon(AWS) Lex prompts the user for some slot values. The user may provide data for all required slots before Amazon (AWS) Lex can satisfy the intent.

For example, the OrderBurger intent needs slots such as burger size, bread type, and quantity. In the intent configuration, you add these slots. For every slot, you give slot type and a prompt for Amazon(AWS) Lex to convey to the client to send data from the user. A user can respond with a slot value that would have additional words, such as "large burger please" or "let's have small." Amazon(AWS) Lex can still understand the intended slot value.

Slot type – Each slot has a type. You can built your slot types or use built-in slot types. For instance, you can built and use the few slot types for the OrderBurger intent:

Size – With some values Small, Medium, and Large.

Crust – With some values Thick and Thin.

# **Chapter 4: Snapshots**

# **Snapshot for the project:**

A screenshot of a cell phone

Description automatically generated

Fig 1. What is Chatbot

A screenshot of a cell phone

Description automatically generated

Fig 2. Flow of Technologies

A screenshot of a cell phone

Description automatically generated

Fig 3. Example with Flow of Technologies

**Snapshot while building BOT:**

A screenshot of a cell phone

Description automatically generated

Fig 4. Add Slots for Indent

A screenshot of a social media post

Description automatically generated

Fig 5. Creating Alias and Publishing BOT with Verion

A screenshot of a computer

Description automatically generated

Fig 6. Installing Serverless

A screenshot of a cell phone

Description automatically generated

Fig 7. Working with Serverless

A screenshot of a social media post

Description automatically generated

Fig 8. Normal Code For Lamba Function

# **On-Screen Snapshot of the BOT**

A screenshot of a cell phone

Description automatically generated

Fig 9. BOT testing on Lex Window

A screenshot of a cell phone

Description automatically generated

Fig 10. Deploying Bot in Slack Screen

A screenshot of a computer

Description automatically generated

Fig 11. Facebook Page where BOT is Added

A screenshot of a cell phone

Description automatically generated

Fig 12. Reply from BOT in messenger Page

# **Chapter 5: Constraints**

# **Constraints:**

Various barriers and constraints exist when introducing a new system such as:

**Lack of data**: Lack of Data Maybe the biggest problem posed by recommender systems is that they need a lot of data to make the chatbot work effectively. It's no surprise that those chatbot with a ton of utternces and slot data run most successful businesses with outstanding responses.

**Changing data**: As we all know the trends are changing day by day so we always need data that match the current data.

**Changing User Preferences**: The problem here is that while I have a specific intention to order, for instance. Pizza–I might have another goal tomorrow. A classic example is that one day I'm going to order pizza, but the next day I'm going to order coffee. So sometimes the syatem must be updated with the trending foods and need to be introduced.

**Language:** For the time, Amazon(AWS) Lex supports only US English language. Currently, it don’t understand any other language.So, Amazon(AWS) Lex trains your bots to understand only US English inputs.

**API Limits: API Limits**

Speech input to the PostContent operation can be up to 15 seconds long.

In both the runtime API operations PostContent and PostText, the input text size can be up to 1024 Unicode characters.

The maximum size of PostContent headers is 16 KB. The maximum size of request and session headers combined is 12 KB.

The maximum input size to a Lambda function is 12 KB. The maximum output size is 25 KB, of which 12 KB can be session attributes.

# **Chapter 6. Conclusion**

# **Conclusion:**

# Recommender systems have great value in recommending relevant resources to users. It can be quite useful in every sector and has functionality similar to human behavior.

# The effectiveness of recommender system relies on the algorithm it uses to find interesting reply which is natural language processing.

# This report presents chatbot system which can be used wherever we want. Here we have implemented it in a caffe whose platform is Amazon(AWS) Web Services and is written in Python and NodeJS programming language in PyCharm IDE environment. The main aim of the system is to improve sparsity problem successfully.

# Subsequently, these techniques are examined in both positive and negative directions.

# In the most crucial part, comprehensive amount of study is done about overall system design and the string matching approach.

**Refernces:**

[https://docs.aws.Amazon(AWS).com/lex/](https://docs.aws.amazon.com/lex/)