# BUILDING CHATBOT WITH AMAZON LEX AND AMAZON LAMBDA

---

**GROUP-1**

B. CHANDANA PRIYA - 20BCS026
E. VAISHNAVI - 20BCS044
KHUSHI G K - 20BCS071
L. DIVYA SRI - 20BCS076
RAKSHITHA Y - 20BCS107

**Instructor:**

**Dr. MALAY KUMAR**
**INDIAN INSTITUTE OF INFORMATION TECHNOLOGY DHARWAD**

## CONTENT

## INTRODUCTION

Chatbots are computer programs designed to simulate conversations with human users over messaging apps or websites. They use natural language processing (NLP) to understand and respond to users' requests in a conversational manner.

Chatbot technology that utilizes Amazon Lambda and Amazon Lex allows users to interact with applications or services using natural language. It uses Amazon Lex to understand user requests and AWS Lambda is to create and deploy chatbots that can interpret the requests and perform tasks. Developers use Amazon Lex to define user intent and specific information that the chatbot requires to complete the user's request. AWS Lambda is used to create code that carries out specific tasks based on user input and integrates with other AWS services.

## AMAZON LEX

Amazon Lex is a service for building conversational interfaces using voice and text. It allows developers to create chatbots that can interact with users through natural language conversations. It has features like it provides a comprehensive set of tools for chatbot development, including automatic speech recognition, natural language understanding, and control of the conversation flow.

It uses Machine Learning to understand user requests and to respond appropriately to the users. it can also integrate with other services such as Amazon Lambda for customized responses and backed processing.

To use Amazon Lex in a chatbot, developers need to define the intents and slots for the chatbot and provide training data to help Amazon Lex understand the user's natural language input. Once the chatbot is deployed, users can interact with it by typing or speaking their requests. Amazon Lex then processes the user's input, extracts the intent and any required slot values, and invokes the appropriate AWS Lambda function to fulfill the user's request. It has benefits like it is Easy to use, cost-effective, and scalable. Provides a consistent  user experience across multiple channels.

## AMAZON LAMBDA

Amazon Lambda is a serverless computing service provided by Amazon Web Services (AWS) that allows developers to run code in response to events or triggers without managing server or infrastructure.

In the case of chatbots, Lambda can be used to write the code that manages the chatbot's operations, including responding to user requests and generating appropriate responses. The code can be written in one of the supported languages, such as Python, Node.js, Java, or C#, and can be executed in response to requests from the chatbot's user interface.

Amazon Lambda can be used in a chatbot to process user requests and generate responses. When a user interacts with the chatbot, the message is sent to Amazon Lex, which triggers a Lambda function to process the request and generate a response. The Lambda function can access databases and other resources as needed to generate the response, and once the response is generated, it is returned to Amazon Lex and sent back to the user. AWS automatically manages the scaling and provisioning of the Lambda function, ensuring that the chatbot is responsive and highly available, and developers can focus on building and improving the chatbot's functionality and user experience.

The Lambda function can also be used to integrate the chatbot with other AWS services, such as Amazon S3, Amazon DynamoDB, or Amazon API Gateway, to create a more complex and robust chatbot.

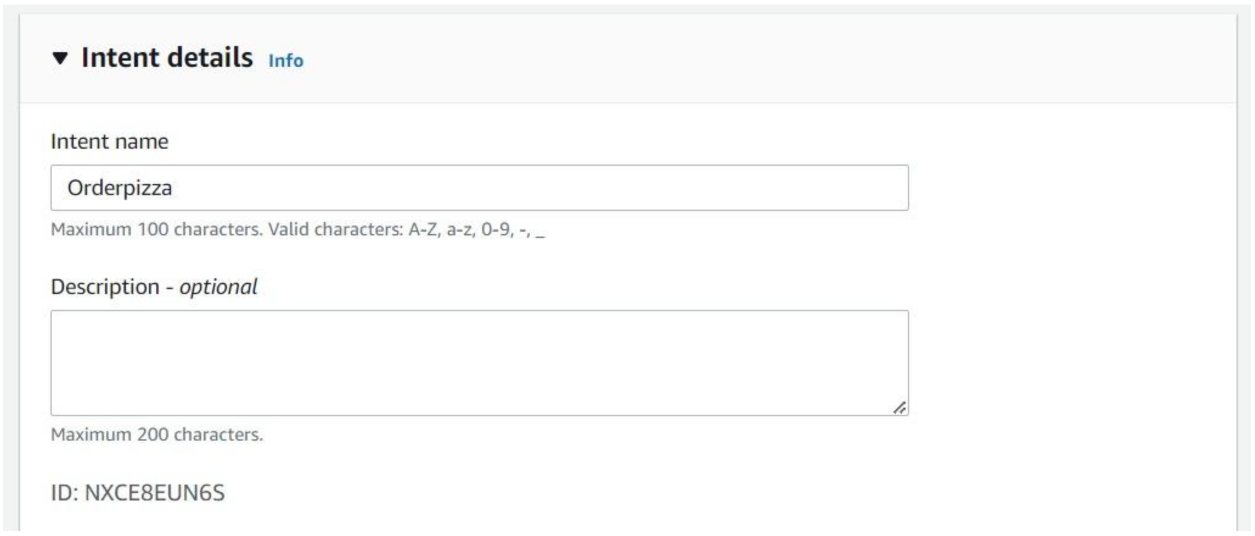
## IMPLEMENTATION

### Step-1: Creating an Amazon Lex Bot

We created a new bot in amazon lex service by giving the bot a name, description, and language.

### Step-2: Creating intents

Intents represent the different actions that users can perform with our chatbot. For ordering pizza, we created an intent called "OrderPizza" that includes slots for the type of pizza, toppings, and delivery information. we created intents using the Amazon Lex console.



▼ Intent details Info

Intent name

Orderpizza

Maximum 100 characters. Valid characters: A-Z, a-z, 0-9, -, _

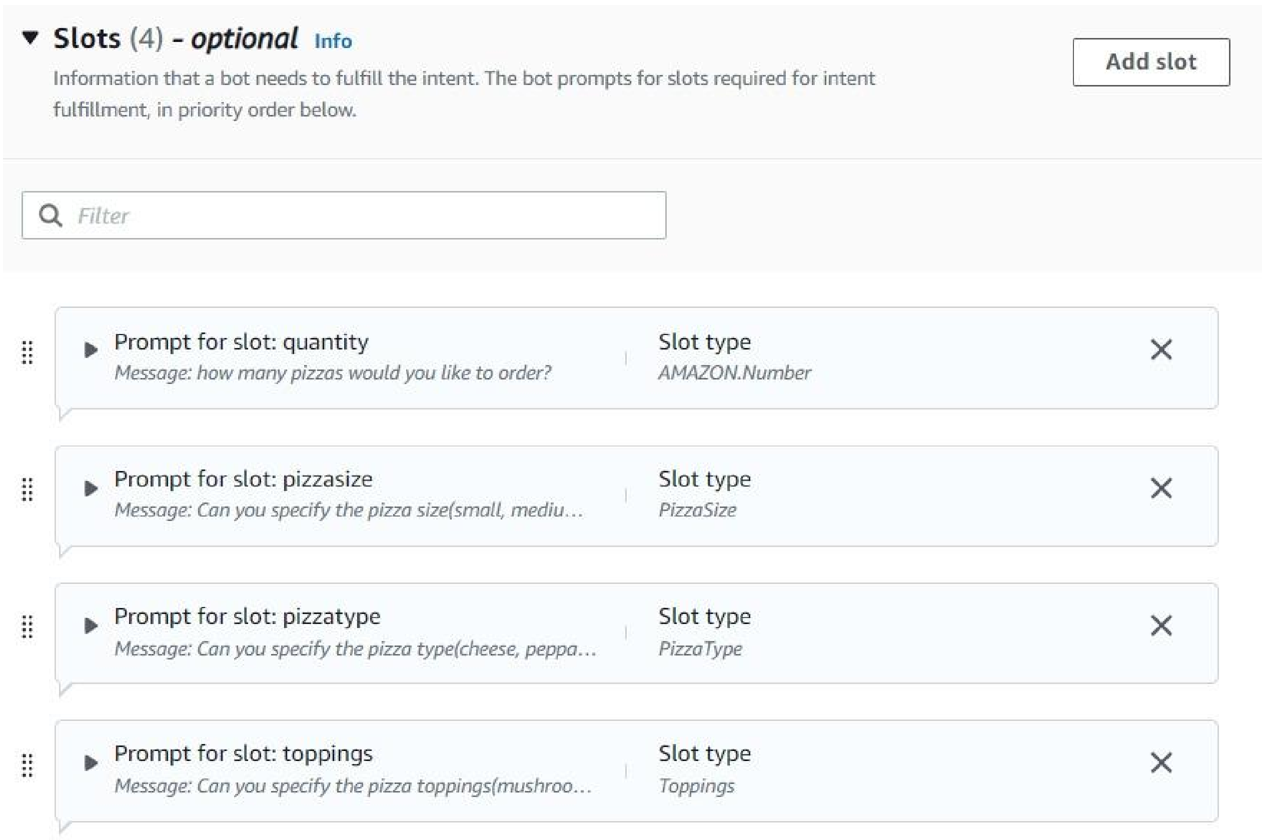
Description - *optional*

Maximum 200 characters.

ID: NXCE8EUN6S

### Step-3: Creating slots

Slots represent the different pieces of information that our chatbot needs to collect from the user to fulfill an intent. For example, in the "OrderPizza" intent, we created slots for the type of pizza, size of pizza, toppings, and quantity.



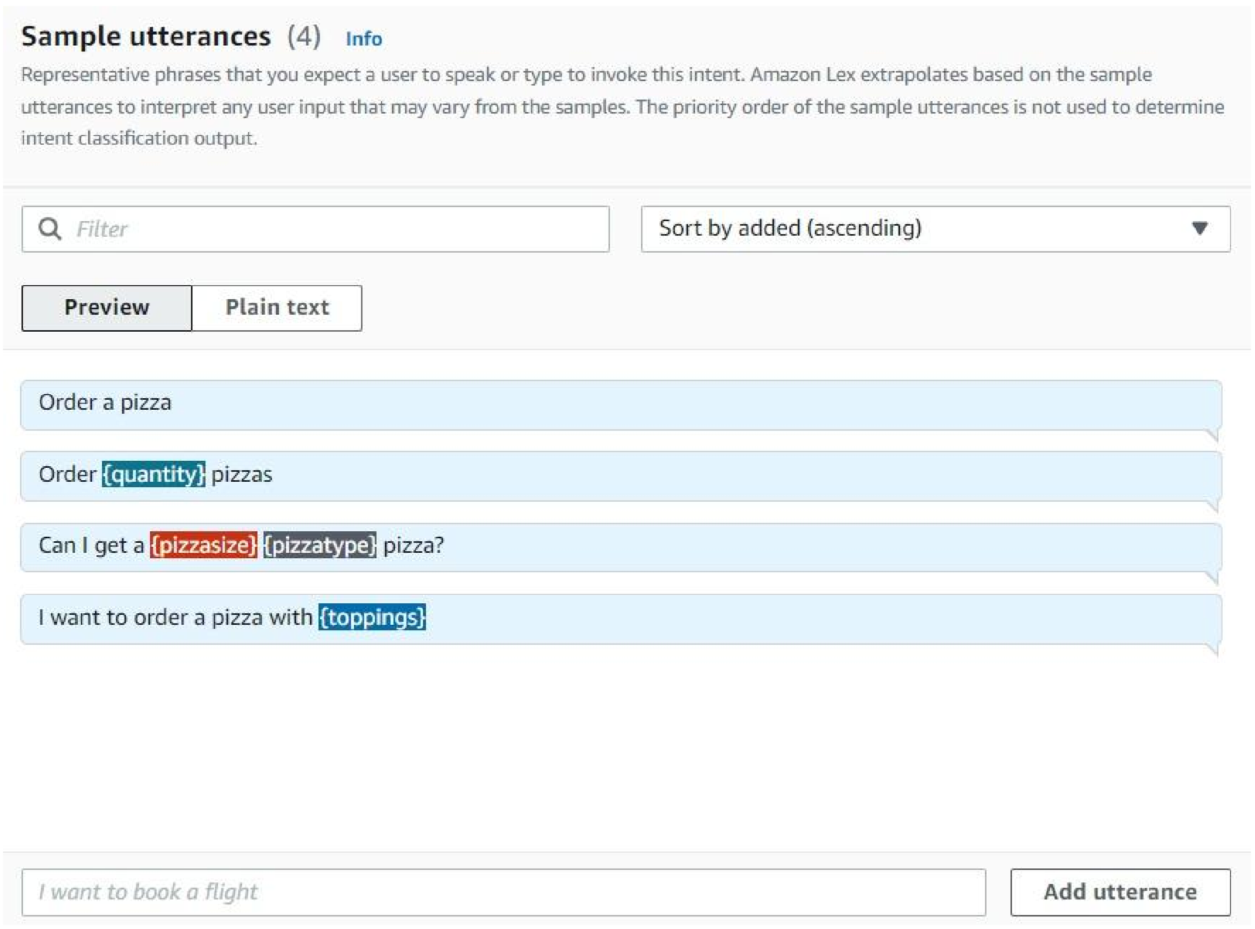
**Slots** (4) - *optional* Info

Information that a bot needs to fulfill the intent. The bot prompts for slots required for intent fulfillment, in priority order below.

Add slot

Filter

| | |
|---|---|
| Prompt for slot: quantity<br>*Message: how many pizzas would you like to order?* | Slot type<br>*AMAZON.Number* |
| Prompt for slot: pizzasize<br>*Message: Can you specify the pizza size(small, mediu…* | Slot type<br>*PizzaSize* |
| Prompt for slot: pizzatype<br>*Message: Can you specify the pizza type(cheese, peppa…* | Slot type<br>*PizzaType* |
| Prompt for slot: toppings<br>*Message: Can you specify the pizza toppings(mushroo…* | Slot type<br>*Toppings* |



Sample utterances (4) Info

Representative phrases that you expect a user to speak or type to invoke this intent. Amazon Lex extrapolates based on the sample utterances to interpret any user input that may vary from the samples. The priority order of the sample utterances is not used to determine intent classification output.

Order a pizza

Order {quantity} pizzas

Can I get a {pizzasize} {pizzatype} pizza?

I want to order a pizza with {toppings}

## Step-4: Creating a Lambda function

Lambda is a serverless computing service that used to run code without thinking of clusters or servers. We created a Lambda function to handle the logic of our chatbot. We created a Lambda function that retrieves pizza information from a database.

```
import json

import datetime

import time



vpizzasize = ['small','medium','large']
```

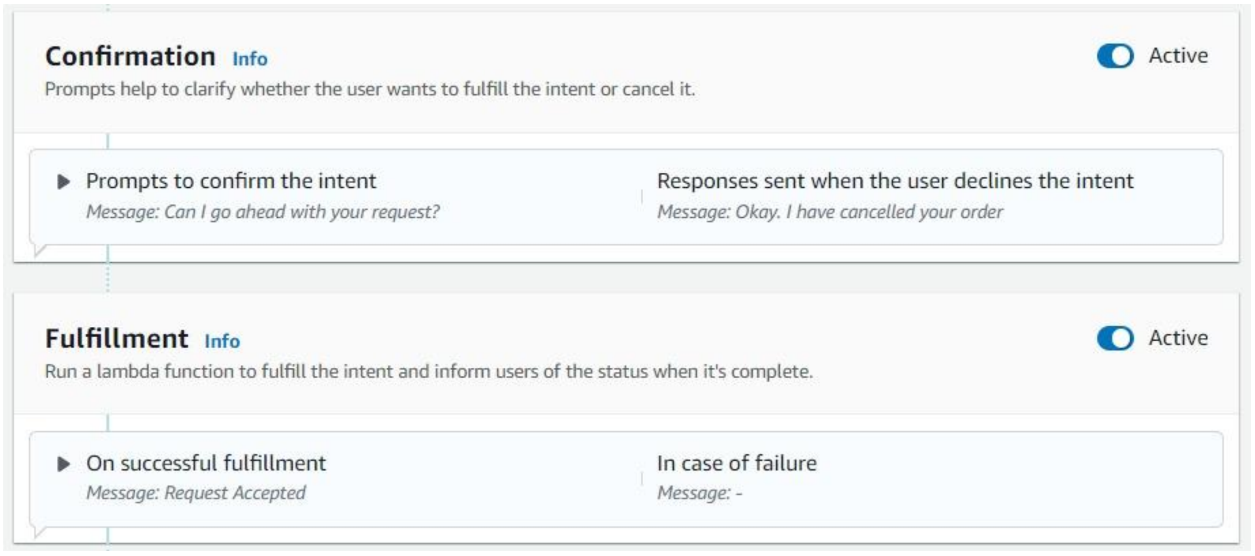
```
vpizzatype = ['cheese','pepparoni', 'veg']
vtoppings = ['mushrooms','olives','tomatoes']


def validate(slots):


    if not slots['quantity']:
        return {
        'isValid': False,
        'violatedSlot': 'quantity'
    }


    if not slots['pizzasize']:


        return {
        'isValid': False,
        'violatedSlot': 'pizzasize',
    }


    if slots['pizzasize']['value']['originalValue'] not in  vpizzasize:


        print("please choose a valid option")
        return {
        'isValid': False,
        'violatedSlot': 'pizzasize',
        'message': 'please choose a valid option'
        }



    if not slots['pizzatype']:
        return {
        'isValid': False,
        'violatedSlot': 'pizzatype'
    }


    if slots['pizzatype']['value']['originalValue'] not in  vpizzatype:


        print("please choose a valid option")


        return {
        'isValid': False,
        'violatedSlot': 'pizzatype',
        'message': 'please choose a valid option'
        }
        if not slots['toppings']:

            return {

            'isValid': False,

            'violatedSlot': 'toppings'

        }


        if slots['toppings']['value']['originalValue'] not in  vtoppings:


            print("please choose a valid option")


            return {

            'isValid': False,

            'violatedSlot': 'toppings',

            'message': 'please choose a valid option'

            }


        return {'isValid': True}


    def lambda_handler(event, context):


        # print(event)

        slots = event['sessionState']['intent']['slots']

        intent = event['sessionState']['intent']['name']
```

```python
    print(event['invocationSource'])

    print(slots)

    print(intent)

    validation_result = validate(slots)


    if event['invocationSource'] == 'DialogCodeHook':

        if not validation_result['isValid']:


            if 'message' in validation_result:


                response = {

                "sessionState": {

                    "dialogAction": {

                        'slotToElicit':validation_result['violatedSlot'],

                        "type": "ElicitSlot"

                    },

                    "intent": {

                        'name':intent,

                        'slots': slots


                        }

                },

                "messages": [
                {

                    "contentType": "PlainText",

                    "content": validation_result['message']

                }

            ]

           }

    else:

        response = {

        "sessionState": {

            "dialogAction": {

                'slotToElicit':validation_result['violatedSlot'],

                "type": "ElicitSlot"

            },

            "intent": {

                'name':intent,

                'slots': slots


                }

        }

       }
```

```
        else:
            response = {
            "sessionState": {
                "dialogAction": {
                    "type": "Delegate"
                },
                "intent": {
                    'name':intent,
                    'slots': slots

                    }

            }
        }

    if event['invocationSource'] == 'FulfillmentCodeHook':

        response = {
        "sessionState": {
            "dialogAction": {
                "type": "Close"
            },
            "intent": {
                'name':intent,
                'slots': slots,
                'state':'Fulfilled'

                }

        },
        "messages": [
            {
                "contentType": "PlainText",
                "content": "Request accepted"
            }
        ]
    }

    return response
```

**Step-5: Linking Lambda function to the bot**

After creating the lambda function, we linked it to Amazon Lex bot using the "Fulfillment" section of the console. This allows Amazon Lex to call our Lambda function when it needs to perform an action on behalf of the user.



**Confirmation** Info

Active

Prompts help to clarify whether the user wants to fulfill the intent or cancel it.

▶ Prompts to confirm the intent
*Message: Can I go ahead with your request?*

Responses sent when the user declines the intent
*Message: Okay. I have cancelled your order*

**Fulfillment** Info

Active

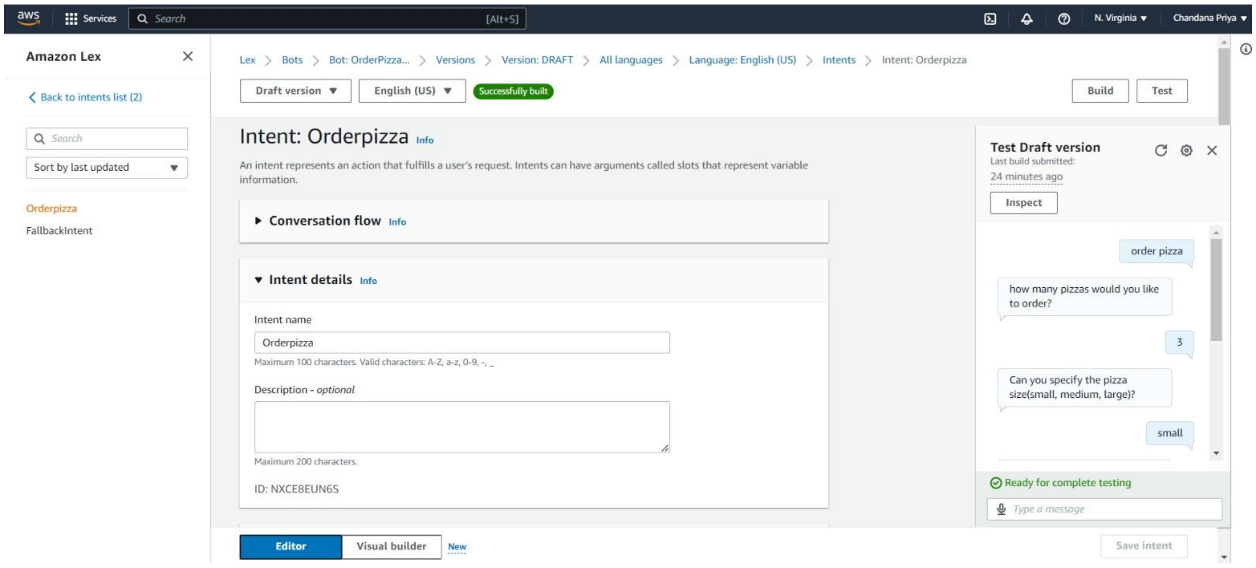
Run a lambda function to fulfill the intent and inform users of the status when it's complete.

▶ On successful fulfillment
*Message: Request Accepted*

In case of failure
*Message: -*

## Step-6: Testing the bot

After creating the bot, intents, slots, and Lambda function, we tested the bot using the Amazon Lex console. We typed in sample user input to see how our bot responds.



Intent: Orderpizza Info

An intent represents an action that fulfills a user's request. Intents can have arguments called slots that represent variable information.

▶ Conversation flow Info

▼ Intent details Info

Intent name

Orderpizza

Maximum 100 characters. Valid characters: A-Z, a-z, 0-9, -, _

Description - *optional*

Maximum 200 characters.

ID: NXCE8EUN6S

Test Draft version
Last build submitted:
24 minutes ago

Inspect

order pizza

how many pizzas would you like to order?

3

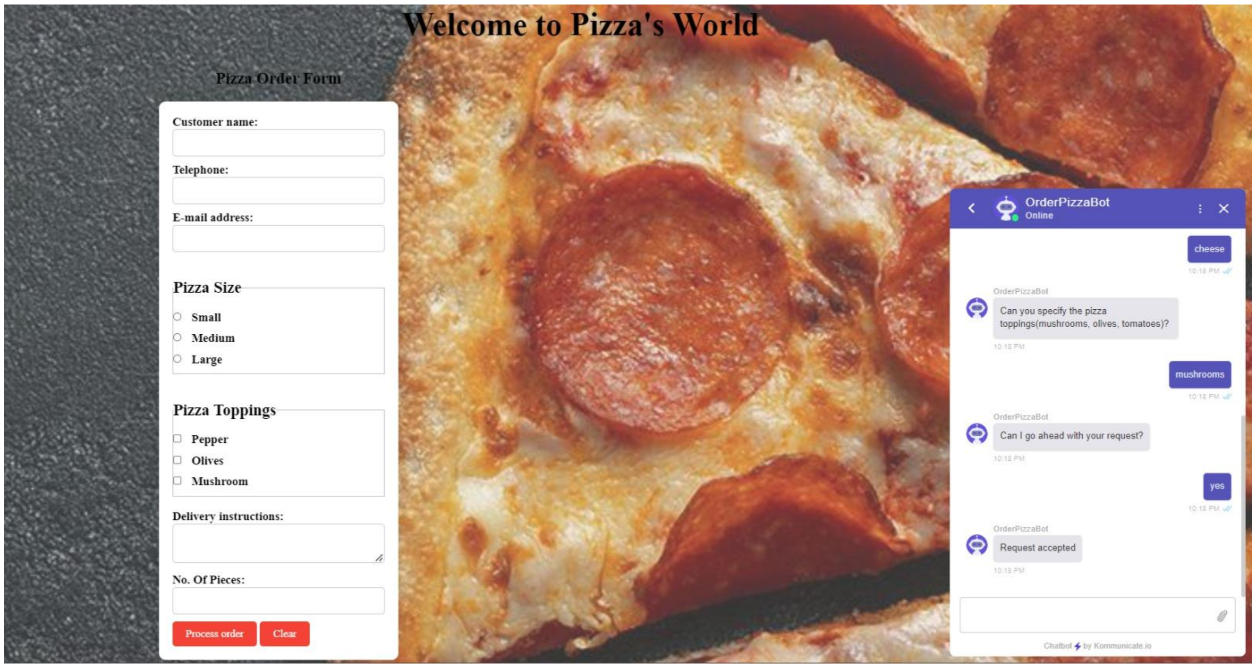
Can you specify the pizza size(small, medium, large)?

small

Ready for complete testing

## Step-7: Deploying the bot

Once we were satisfied with the chatbot, We deployed it to a website. We did this using the “Kommunicate”.


Welcome to Pizza's World

Pizza Order Form

Customer name:

Telephone:

E-mail address:

Pizza Size
- Small
- Medium
- Large

Pizza Toppings
- Pepper
- Olives
- Mushroom

Delivery instructions:

No. Of Pieces:

Process order  Clear

OrderPizzaBot
Online

cheese
10:18 PM

OrderPizzaBot
Can you specify the pizza toppings(mushrooms, olives, tomatoes)?
10:18 PM

mushrooms
10:18 PM

OrderPizzaBot
Can I go ahead with your request?
10:18 PM

yes
10:18 PM

OrderPizzaBot
Request accepted
10:18 PM

Chatbot by Kommunicate.io

## DISCUSSION

Building a chatbot with Amazon Lex and Amazon Lambda is a powerful way to provide conversational interfaces to users. By using natural language processing capabilities of Amazon Lex, the chatbot can understand and interpret user requests, and then trigger a Lambda function to generate an appropriate response. With Amazon Lambda, developers can focus on writing the business logic for the chatbot and accessing resources as needed, without having to worry about server management or scaling.

In the case of building a chatbot for ordering pizza, Amazon Lex and Amazon Lambda can be used to provide a perfect and intuitive user experience. By using Amazon Lex's built-in slot types and intents, the chatbot can easily understand and respond to user requests for pizza orders, while the Lambda function can handle the business logic for processing the order and retrieving relevant information such as pricing and delivery options. Overall, building a chatbot with Amazon Lex and Amazon Lambda can provide a powerful and scalable solution for businesses looking to engage with customers in a conversational and personalized way.

## CONCLUSION

Developing a chatbot using Amazon Lambda and Amazon Lex can be an efficient and cost-effective solution for businesses and organizations to automate their customer support and engagement processes. Amazon Lex is a natural language processing tool that helps developers build chatbots with conversational capabilities. Amazon Lambda offers a serverless computing platform, which means developers can write and execute code without needing to manage servers. These services help to build chatbots quickly and easily that can understand and respond to customer queries and provide personalized assistance.

## REFERENCES

https://aws.amazon.com/lex/

https://aws.amazon.com/lambda/

https://aws.amazon.com/blogs/aws/building-a-chatbot-with-amazon-lex-and-aws-lambda/

https://medium.com/analytics-vidhya/building-chatbots-with-amazon-lex-and-aws-lambda-a78c7d5b5f5c

https://aws.amazon.com/solutions/case-studies/zomato-chatbot/

https://aws.amazon.com/getting-started/hands-on/build-chatbot-lex-lambda-api-gateway/

https://docs.aws.amazon.com/lambda/latest/dg/welcome.html

https://aws.amazon.com/blogs/machine-learning/building-chatbots-with-amazon-lex-and-aws-lambda/

https://www.zehntech.com/amazon-lex-and-aws-lambda-building-chatbot/