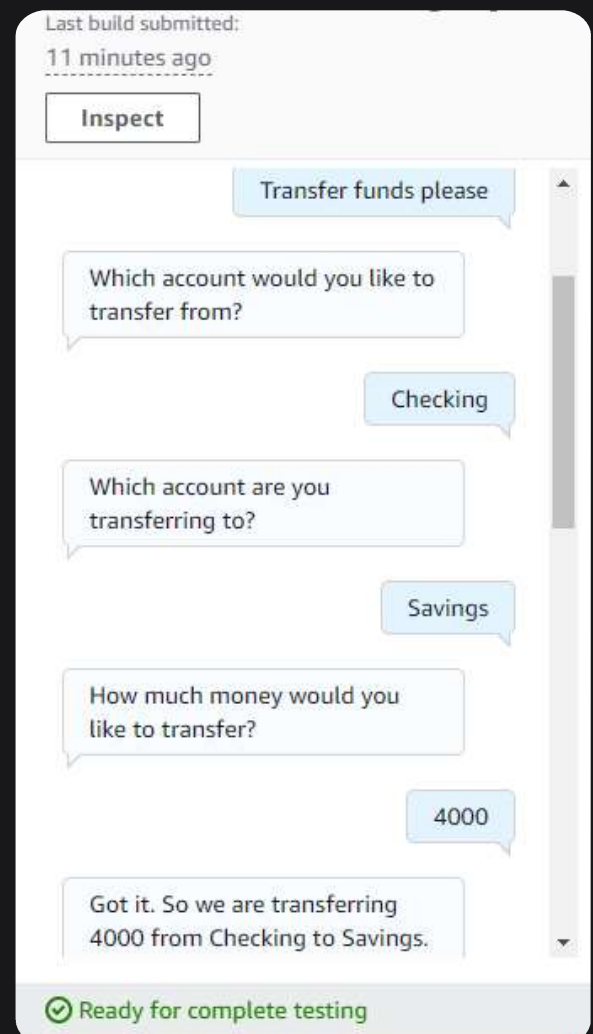


How I built a chatbot with Amazon Lex



with multiple slots! 🎰🎰🎰



Kanika Mathur

@github.com/KanikaGenesis



What is Amazon Lex?

What it does:

- **Helps you build voice and text chatbots in minutes.**

Why it's useful:

- It uses AI/ ML capabilities to classify user intents and understand intents that are beyond what I've programmed.

How I'm using it in today's project:

- In this project I'm using Amazon Lex to create BankerBot, a chatbot that will now not only have natural conversational speech recognition, but is also set up to check their account balance and transfer money between accounts!



Kanika Mathur

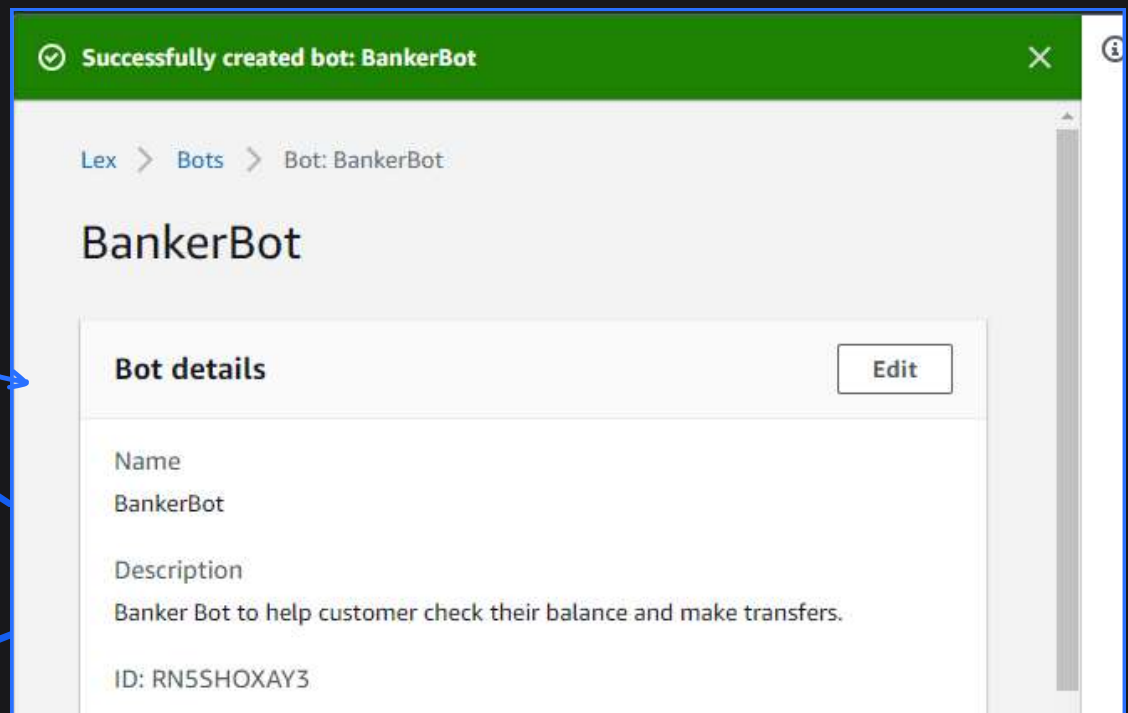
 [@github.com/KanikaGenesis](https://github.com/KanikaGenesis)



Set up a Lex chatbot

- I created BankerBot from scratch and used most default settings on Lex.
- In terms of the **intent classification confidence score**, I kept the default value of 0.40. What this means for my chatbot is it should at least be 40% confident about the intent/ goal of the chatbot user to respond. In more technical terms there should at least be 40% match between the user's input and intent I program for my banker bot to respond accordingly.

Setting up my Lex
chatbot...



Kanika Mathur

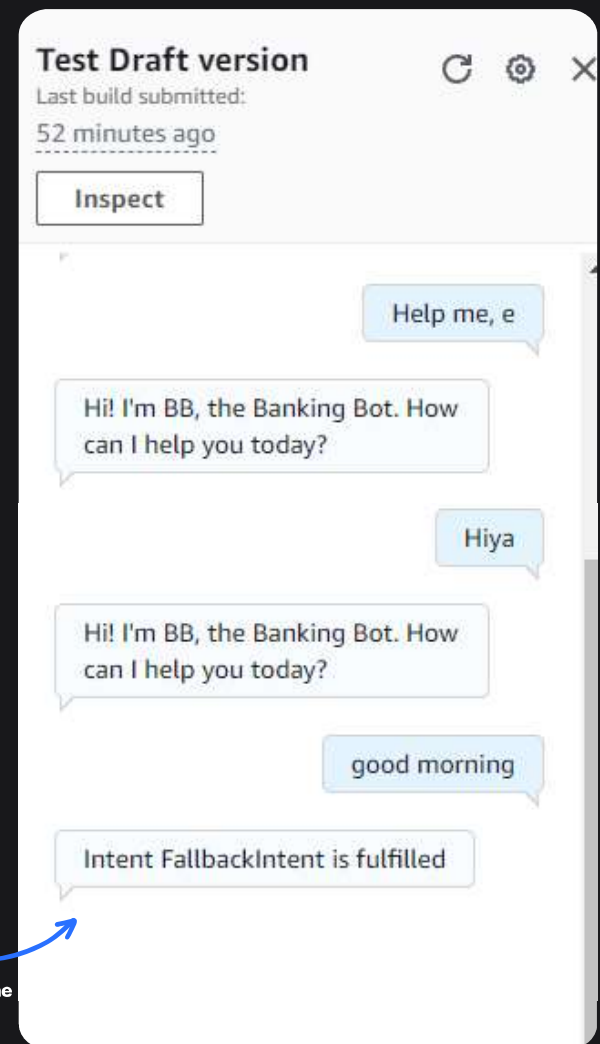
 github.com/KanikaGenesis



Create an intent in Lex



- Intents represent user's goals/ purposes for using the chat bot. In Amazon Lex a chat bot is defined by the intents it supports.
- My first intent, WelcomeIntent, was created to greet the user when they say Hello.
- To set up this intent, I created sample utterances(eg. "Hi", "Hello", "I need help") and a closing response i.e how a chat bot will respond.
- I launched and tested the chatbot, which could still respond if I enter similar utterances (eg: Hiya).
- However, the chatbot returned the error message "Intent FallbackIntent is fulfilled" when I entered "Good Morning".
- This error message occurred because my chat bot could not understand the intent of the phrase "Good Morning".



My first test of the
chatbot



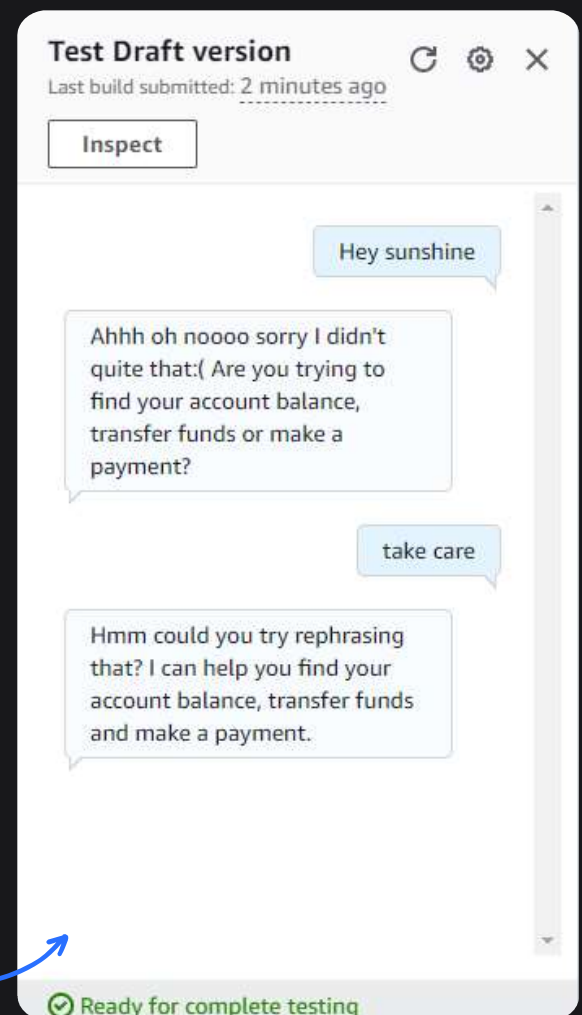
Kanika Mathur

 github.com/KanikaGenesis



Manage FallbackIntent

- FallbackIntent is a default intent in every chatbot that gets triggered when the chatbot does not recognize the user's goal/purpose.
- I wanted to configure FallbackIntent because the default closing response to the user is not easily understandable.
- To configure FallbackIntent, I had to create my own closing response in the intent's setup page. "Sorry I am having trouble understanding. Can you describe what you'd like to do in a few words? I can help you find your account balance, transfer funds and make a payment."
- I also added variations! What this means for an end user is they get to see different forms of my chatbot's closing response.



Perfect! The error message is now much clearer, and there are variations too



Kanika Mathur

 github.com/KanikaGenesis



Create custom slots

- **Slots** are like placeholders/ ' blanks ' of information that my chatbot will seek to fill in order to fulfil an intent.
- In this project, I created a **custom slot type** to represent the different account types that a banking customer could have. A custom slot type was required as the default slot types did not accommodate for account types (e.g. Checking, Credit, Savings).
- I then associated the custom slot with a new intent, CheckBalance, which is a new intent I created that will help my bank's customers check its users account balances.

My custom slots

The screenshot shows the 'Slots (2) - optional' section in the Amazon Lex console. It lists two custom slots:

Slot name	Slot type
Prompt for slot: accountType Message: For which account would you like your balan...	accountType
Prompt for slot: dateOfBirth Message: For verification purposes, what is your date ...	AMAZON.Date

An arrow points from the text 'My custom slots' to the first slot entry.

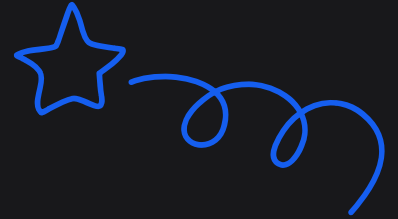


Kanika Mathur

@github.com/KanikaGenesis

Simplifying the user experience

- I included slot values in some of the utterances (i.e. user inputs) for this intent too. For example, I included the utterance **What's the balance in my {accountType} account?**
 - This is an example of an utterance that expects the slot accountType.
- By adding custom slots in the utterance, the user experience is enhanced - the bot will automatically register which account type the user is trying to check, and will not ask for it again. This saves the user's time and makes the conversation much more efficient.



Slot values getting recognised in a conversation

Intent	
CheckBalance	
Slots	Elicitation
accountType	Savings
dateOfBirth	2000-02-03
Active contexts	Number of turns or seconds

01/01/2017

Intent CheckBalance is fulfilled

What's the balance in my savings account?

For verification purposes, what is your date of birth?

02/03/2000



Kanika Mathur

@github.com/KanikaGenesis

Using AWS Lambda

- **AWS Lambda** is an AWS service that helps you to run code without having to manage servers.
- In this project, a Lambda function was created to generate the user's bank balance. In this example, a random figure was generated, however in the real world the Lambda function can be used to extract the user's bank balance from a database. The Amazon Lex chatbot, on its own, would not be able to generate a bank balance. That's why this connection to AWS Lambda is crucial!

A peek into the Python code I uploaded into AWS Lambda!

```
def CheckBalance(intent_request):
    session_attributes = get_session_attributes(intent_request)
    slots = get_slots(intent_request)
    account = get_slot(intent_request, 'accountType')
    #The account balance in this case is a random number
    #Here is where you could query a system to get this information
    balance = str(random_num())
    text = "Thank you. The balance on your "+account+" account is $" +balance+" dollars."
    message = {
        'contentType': 'PlainText',
        'content': text
    }
    fulfillment_state = "Fulfilled"
    return close(intent_request, session_attributes, fulfillment_state, message)
```



Kanika Mathur

@github.com/KanikaGenesis



Connecting Lambda with Lex

There were two steps to connecting the Lambda function with my chatbot:

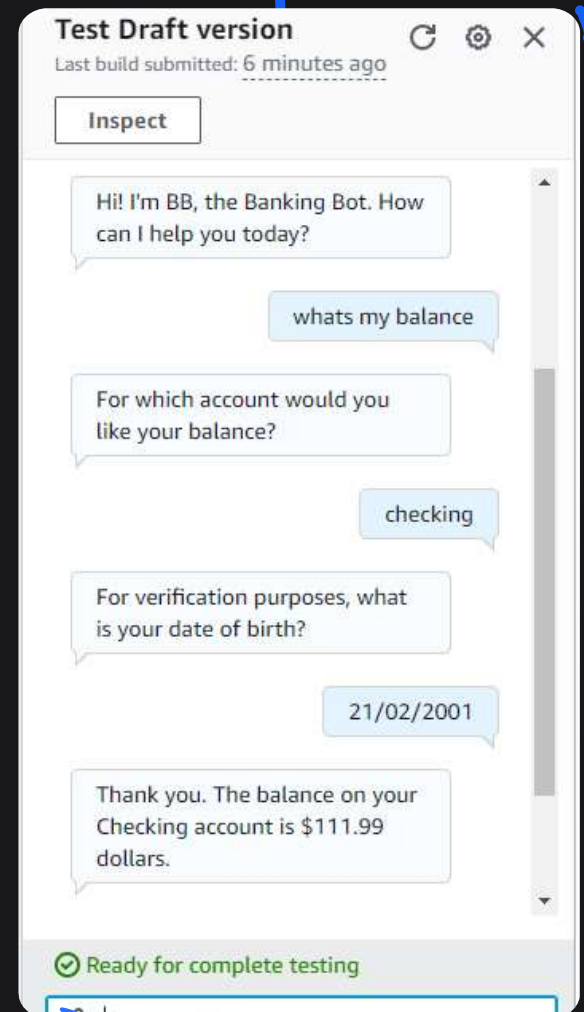
Step 1

- To connect Lambda with my chatbot alias, I visited the Alias page of my chatbot and connected my TestBotAlias (my chatbot's default alias, made for development/ testing) with the latest version of the AWS Lambda function defined.

Step 2

- Another intent setting to configure is **code hooks**.
- A code hook is a piece of code that can be connected to my chatbot to perform functions/ actions that my chatbot cannot do alone/ by default.
- In this project, I had to use code hooks because the chatbot is not able to calculate/ return a bank balance figure on its own.

After connecting Lambda with my Lex bot, my chatbot could immediately start returning specific bank balance figures. The AWS Lambda function would generate a random number each time.



My chatbot now returns
a bank balance number
thanks to Lambda!



Kanika Mathur

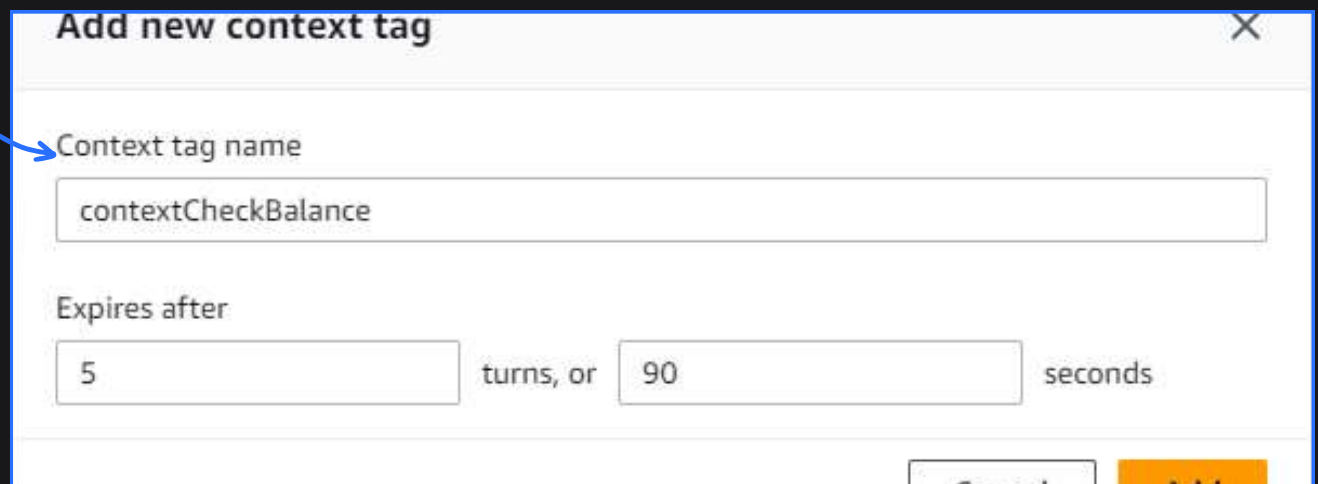
@github.com/KanikaGenesis



Context Tags

- Context tags are tools for Amazon Lex to remember specific pieces of information gathered from a conversation, and reuse that information throughout the session with its user.
- There are two types of context tags, they are output context tags and input context tags.
- I created an output context tag called contextCheckBalance, and I created this in the intent CheckBalance.

A look at output contexts



Add new context tag

Context tag name

contextCheckBalance

Expires after

5 turns, or 90 seconds

Cancel Add



Kanika Mathur

@github.com/KanikaGenesis



A Follow-Up Intent



- I created a new intent called **FollowupCheckBalance**. The purpose of this intent is to let the user check another account's balance without having to provide their date of birth again.
- This intent is related to the previous intent I made, **CheckBalance**, because FollowUpCheckBalance will only get triggered after the user has checked their balance once already (i.e. triggered CheckBalance).
- I created an input context, **contextCheckBalance**, that is using the exact same tag as the output context tag I've set up in the CheckBalance intent. What this means is, input information we are looking for in this intent (FollowUpCheckBalance) can now be retrieved from the CheckBalance intent through this tag.

A look at input contexts

▼ Default values - optional

No default values

You haven't added any default values yet.

Provide a default value, #value for a context value, or [variable] for session variable.

#contextCheckBalance.dateOfBirth

Add default value



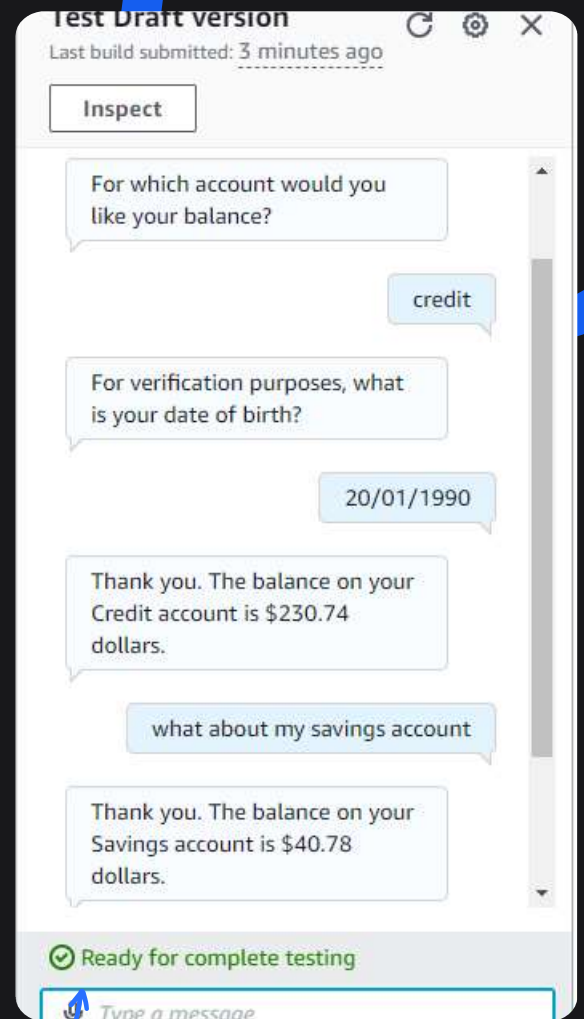
Kanika Mathur

@github.com/KanikaGenesis



Context Tags in Action

- Conversation time! I built and tested my bot after creating the context tags and new intent.
- To see the context tags and the follow up in intent in action, I first triggered the CheckBalance intent, then I followed up with the utterance **“What about my savings account”** to trigger FollowUpCheckBalance.
- If I had gone straight to trying to trigger FollowUpCheckBalance without setting up any context, my chatbot would not have the context needed to fulfil the conversation. As a result, it will return the FallbackIntent i.e. let the user know it doesn't understand the request being made.



My chatbot now carries over the user's date of birth to the next intent!



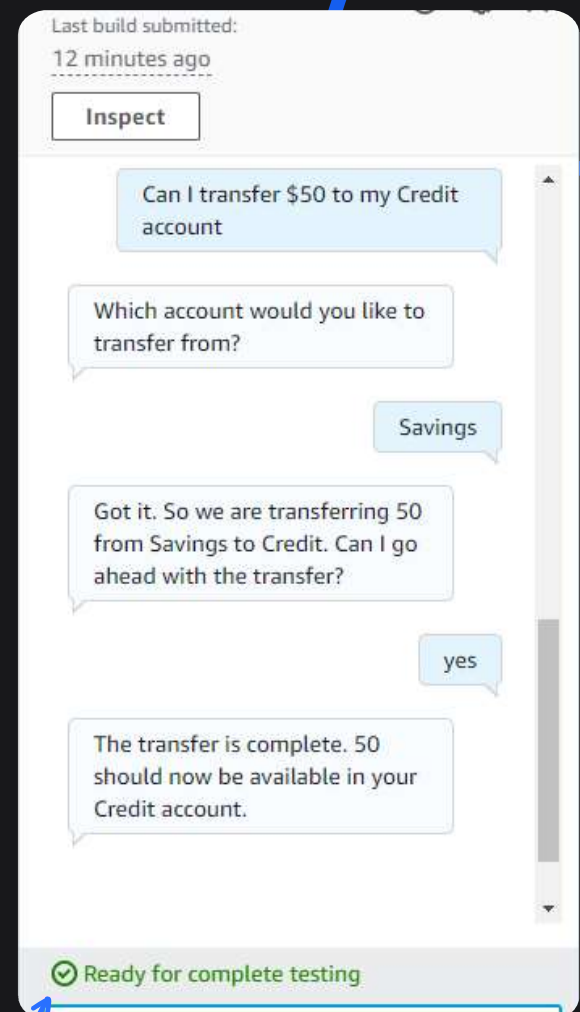
Kanika Mathur

@github.com/KanikaGenesis



More slots!

- Slots are pieces of information that my chatbot needs in order to fulfil an intent.
- The final intent for my chatbot was TransferFunds, which will help the user transfer money between bank accounts.
- For this intent, I had to use the same slot type twice. This is because the TransferFunds intent involved two different accounts - the source account (i.e. the account that we are transferring money from) and the target account (i.e. the account that the money will land in).
- I also learnt how to create confirmation prompts, which are prompts designed for the chatbot to confirm the user's intention to carry out the intent. In this project, a confirmation prompt was used for the chatbot to confirm that the user is wanting to transfer a specific amount of money between two of their bank accounts.



A conversation demonstrating the two slots and the confirmation prompts in action!



Kanika Mathur

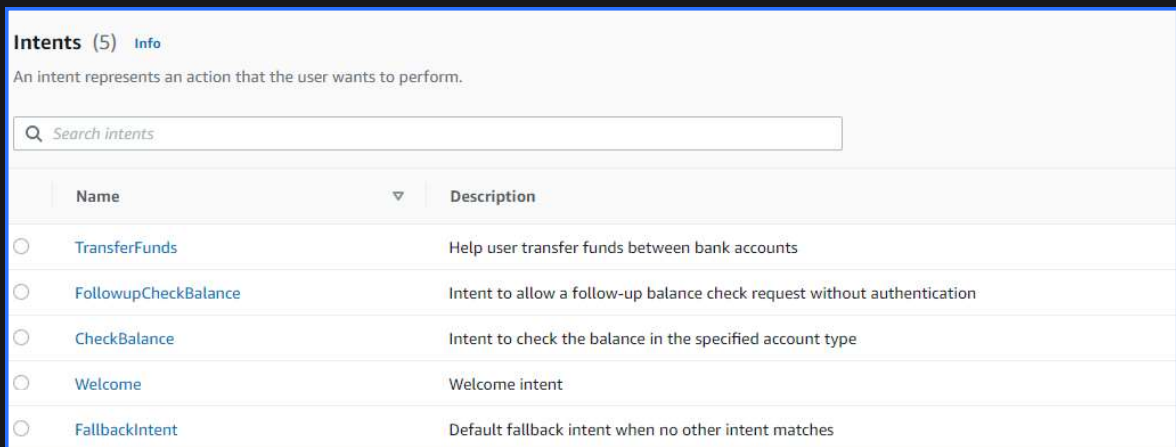
@github.com/KanikaGenesis

A LITTLE EXTRA...

Deploying with CloudFormation

- AWS CloudFormation is service that helps users deploy AWS resources in seconds, by defining the resources and their characteristics in a code file (called a YAML file).
- As an extension to this project, I learnt how to deploy the entire BankerBot using a single CloudFormation stack.
- Doing this took me 1 minute to set up the CloudFormation stack. 2-4 minutes to wait for deployment to complete.
- Something I learnt from deploying with CloudFormation was that you can deploy resources from all different AWS services in the exact same YAML file.

CloudFront
deployed this
for me!



Intents (5) Info	
An intent represents an action that the user wants to perform.	
<input type="text" value="Search intents"/>	
Name	Description
<input type="radio"/> TransferFunds	Help user transfer funds between bank accounts
<input type="radio"/> FollowupCheckBalance	Intent to allow a follow-up balance check request without authentication
<input type="radio"/> CheckBalance	Intent to check the balance in the specified account type
<input type="radio"/> Welcome	Welcome intent
<input type="radio"/> FallbackIntent	Default fallback intent when no other intent matches



Kanika Mathur

@github.com/KanikaGenesis



Final thoughts...

- This project took me approximately 90 minutes to complete.
- Delete EVERYTHING at the end! Let's keep this project free :)
- Now that I know how to use Lex, in the future I'd use it to integrate into my projects such as developing a comprehensive customer support system for an e-commerce platform and create interactive voice response systems for various applications.
- One thing I didn't expect was the ease of use with CloudFormation, found it a lot simpler and less work when using a template.
- An area of Lex I'd like to explore further is connecting the chatbot to an application/database. This would enable dynamic data retrieval and updates, allowing the chatbot to provide real-time information and perform actions based on user inputs. Additionally, I'd like to explore other types of responses, such as acknowledge intent, slot capture success/failure responses, to enhance the user experience and make interactions more intuitive and informative.



Kanika Mathur

 @github.com/KanikaGenesis

Find this helpful?



Like this post

yes!



Leave a comment



Save for later



Let's connect!



Kanika Mathur



@github.com/KanikaGenesis



Thanks NextWork for the
free project guide!

 **NEXTWORK**