

1 What is Amazon Lex?

Amazon Lex is a fully managed AWS service that allows developers to build conversational interfaces for applications using voice and text, essentially enabling the creation of sophisticated chatbots.

2 Setting Up a Lex chatbot

I created my chatbot from scratch with Amazon Lex. Setting it up took me 5 mins While creating my chatbot, I also created a role with basic permissions because Amazon Lex needs the ability to interact with other AWS services like AWS Lambda to add more advanced functionality to the chatbot. In terms of the intent classification confidence score, I kept the default value of 0.40. This means the chatbot needs to be atleast 40% confident of the input provided by the user to be able to give a response.

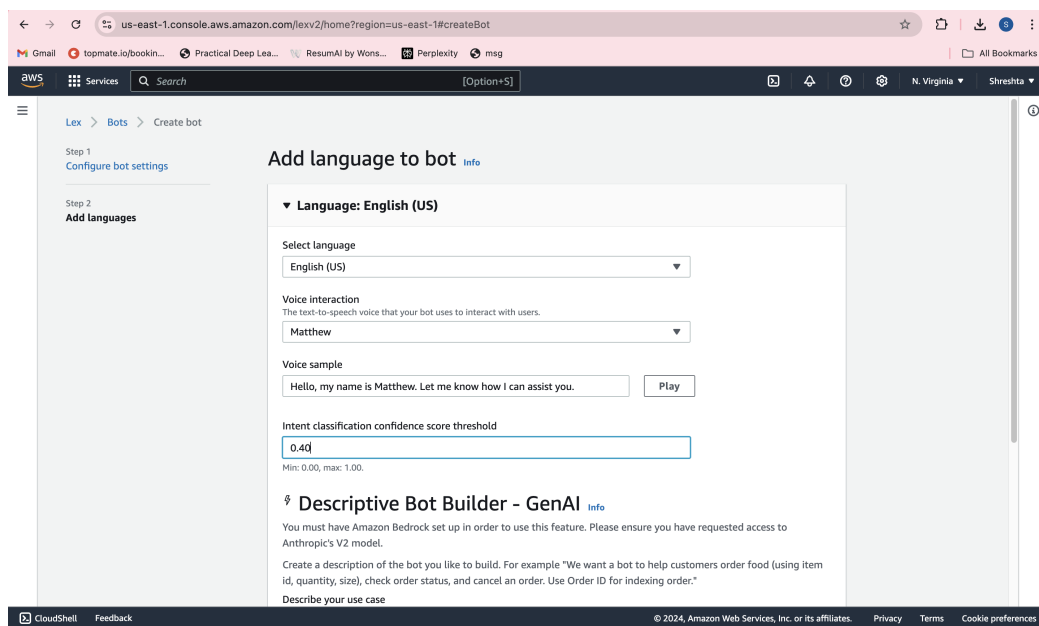


Figure 1: AWS Lex chatbot

3 Intents

An intent is what the user is trying to achieve in their conversation with the chatbot. These are essential categories of user requests that the chatbot is designed to understand and respond to. I created the WelcomeIntent to handle greetings and initial interactions with users. It activates when users input phrases like "Hi", "Hello", "I need help", or "Can you help me?".

4 FallbackIntent

I launched and tested my chatbot, which could respond successfully if I enter "Hi", "Hello", "I need help", or "Can you help me?". The bot replies with a predefined message: "Hi! I'm BB, the Banking Bot. How can I help you today?". My chatbot returned the error message 'Intent FallbackIntent is fulfilled' when I entered "heya". This error message occurred because Amazon Lex doesn't quite recognize the utterance.

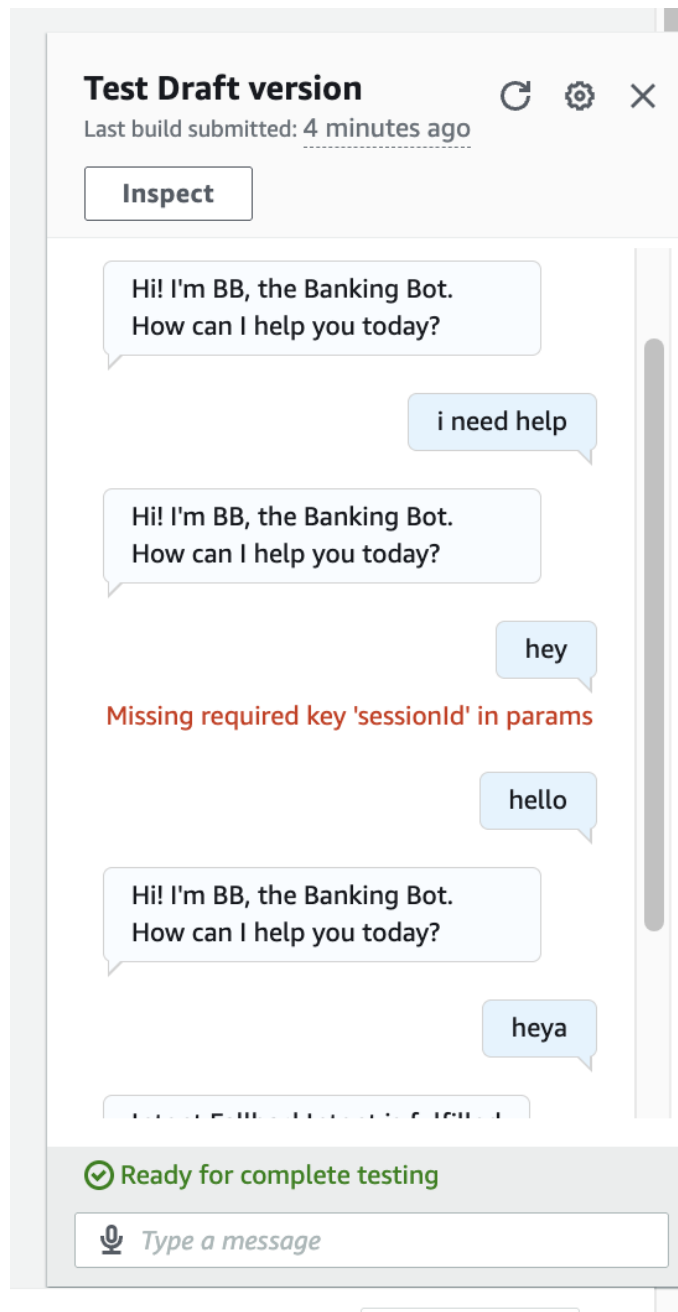


Figure 2: FallbackIntent

5 Configuring FallbackIntent

FallbackIntent is a default intent in every chatbot that gets triggered when there are user input that chatbot doesn't understand. It will fall back onto a predefined response asking user to rephrase his input. I wanted to configure FallbackIntent because when chatbot doesn't the user input it will automatically generate response asking the user to rephrase it. Without a FallbackIntent the chatbot will just show an error.

6 Variations

To configure FallbackIntent, I just added custom messages in the closing response. I also added variations! What this means for an end user is that they will have a dynamic range of conversations with chatbot that sound more real.

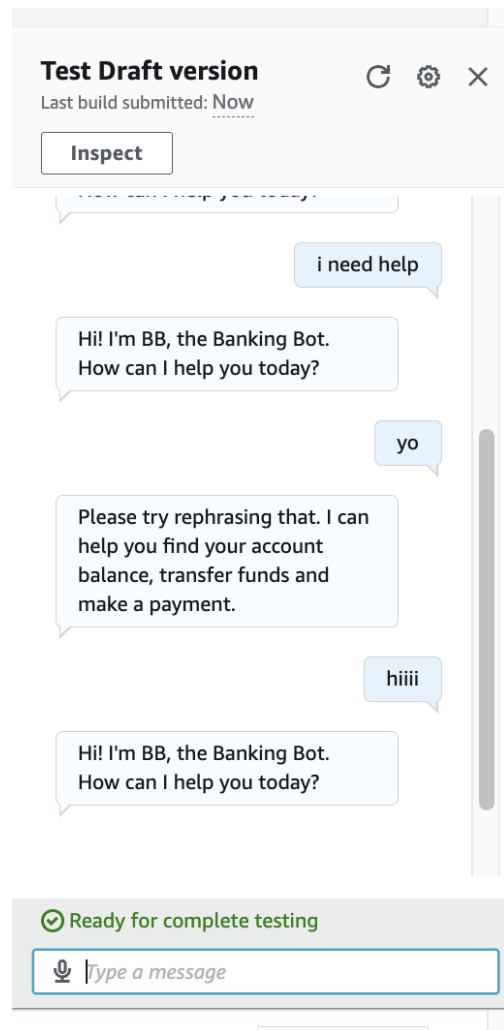


Figure 3: Variations

7 Slots

Slots are essential components in Amazon Lex chatbots that represent specific pieces of information required to fulfill a user's intent. They act as variables that capture and store user-provided data during a conversation.

In this project, I created a custom slot type to define specific account types that the Banker-Bot can recognize and work with. The custom slot type, named "accountType", serves several important purposes like standardization, restriction etc.

This slot type has restricted slot values, which means only these specific account types would be recognized as valid inputs. This prevents the bot from accepting incorrect or non-existent account types.

Slot type: accountType [Info](#)

A slot type is a list of values used to capture values for a slot.

► **Slot type details**

Slot value resolution
Amazon Lex resolves the slot values in an utterance to only the values you provide, or it expands the resolution to related or similar values.

☐ **Expand values (default)**
Values used as training data.
 ☒ **Restrict to slot values**
Use only values provided.

Slot type values
Modify the list of values used to train the machine learning model to recognize values for a slot.

Search slot type values

Checking	Tab or ; or enter return for new value	×
Savings	Tab or ; or enter return for new value	×
Credit	Tab or ; or enter return for new value credit card × visa × mastercard × amex × american express ×	×

Value Tab or ; or enter return for new value **Add value**

Maximum 140 characters. Valid characters: A-Z a-z 0-9 @ # \$

Figure 4: Slots

8 Connecting slots with intents

I associated my custom slot with CheckBalance, which is an intent designed to handle user requests for checking their account balances. This intent performs several key functions like determining balance inquiries, account type etc.

▼ **Slots (2) - optional** [Info](#) **Add slot**

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

Filter

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

Figure 5: Connecting slots with intents

9 Slot values in utterances

I included slot values in some of the utterances (i.e. user inputs) by using curly braces to enclose the slot name within the utterance text.

By adding custom slots in utterances, I enhanced the BankerBot's ability to understand and process user inputs more effectively. This approach offers context understanding and streamlined conversations.

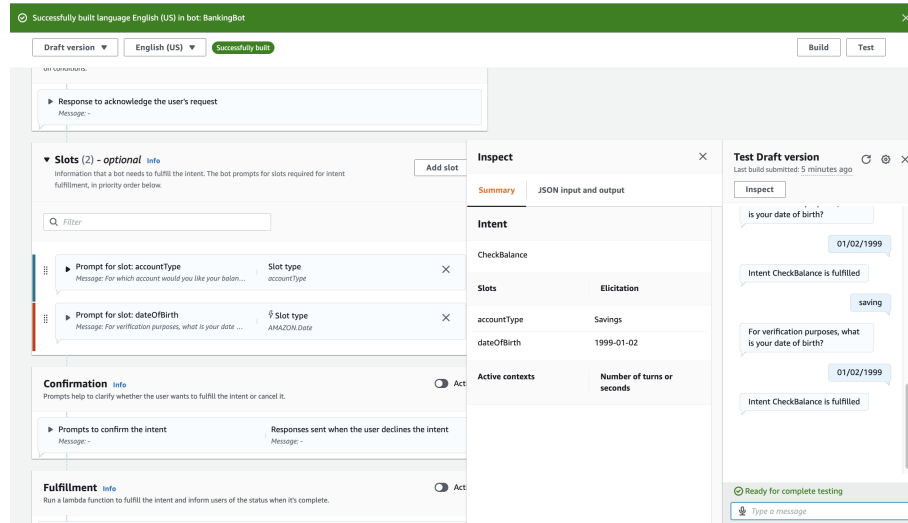


Figure 6: Slot values in utterances

10 AWS Lambda Functions

AWS Lambda is a service that runs code without provisioning or managing servers. Lambda runs the code only when needed and scales automatically, from a few requests per day to thousands per second.

AWS Lambda function that will get a user's balance. It will also connect with the chatbot taht we defined.

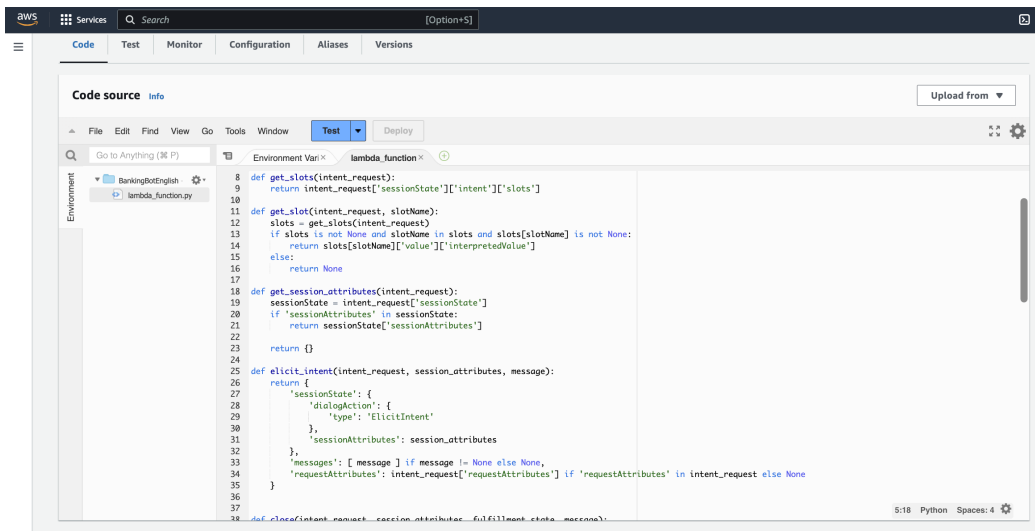


Figure 7: AWS Lambda Functions

11 Chatbot Alias

An alias is a pointer to a specific version of an Amazon Lex V2 bot. It allows you to manage and update the version that client applications use without requiring them to track which version is currently deployed.

TestBotAlias is a default version of the bot that's made for testing or development. This is the playground version of the bot that we use to make sure everything works smoothly before rolling out changes.

To connect Lambda with my BankerBot, I visited my bot's TestBotAlias and selected my Lambda function in the "Source" dropdown. I set the version to "\$LATEST" and saved the changes, linking the bot to Lambda.

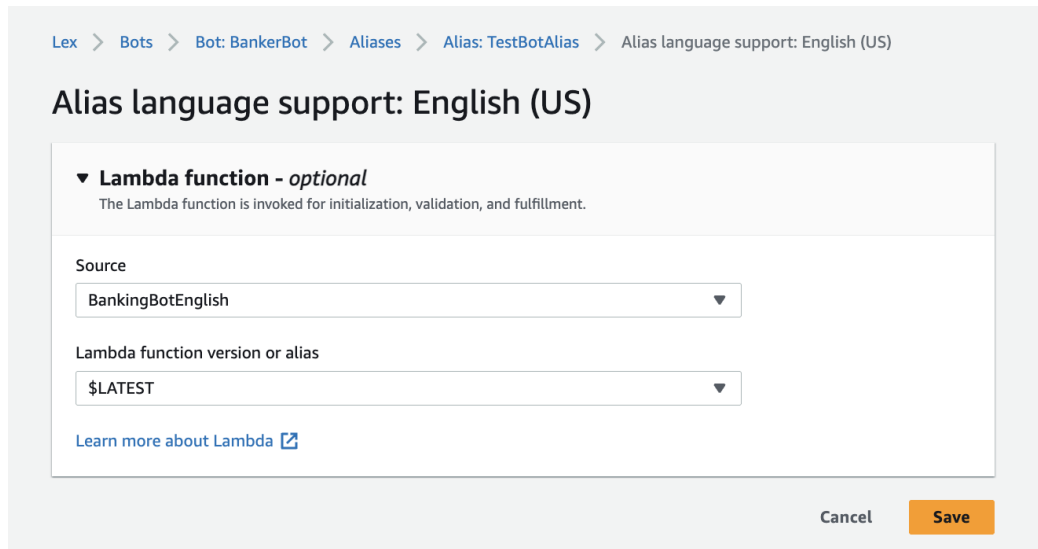


Figure 8: Chatbot Alias

12 Code Hooks

A code hook is a mechanism that allows you to inject custom code at specific points in a software system's execution flow or lifecycle. In the context of Amazon Lex V2, a code hook is a way to connect a Lambda function to your chatbot.

Even though I already connected my Lambda function with my chatbot's alias, I had to use code hooks because it lets me execute my Lambda function at crucial points like slot validation, intent confirmation, and fulfillment.

I could find code hooks at the intent level, specifically within the CheckBalance intent's advanced settings. By configuring this, I ensured that once all required slots (account type and DOB) are filled, my chatbot will trigger lambda function.

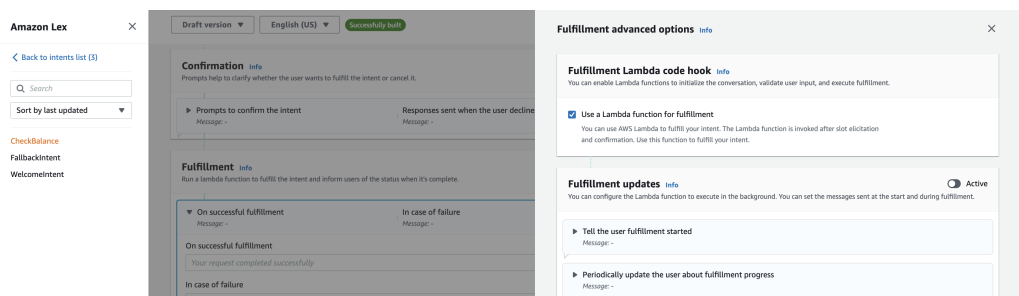


Figure 9: Code Hooks

13 The final result!

I've set up my chatbot to trigger Lambda and return a random dollar figure when the user has provided both the account type they want to check and their date of birth for verification.

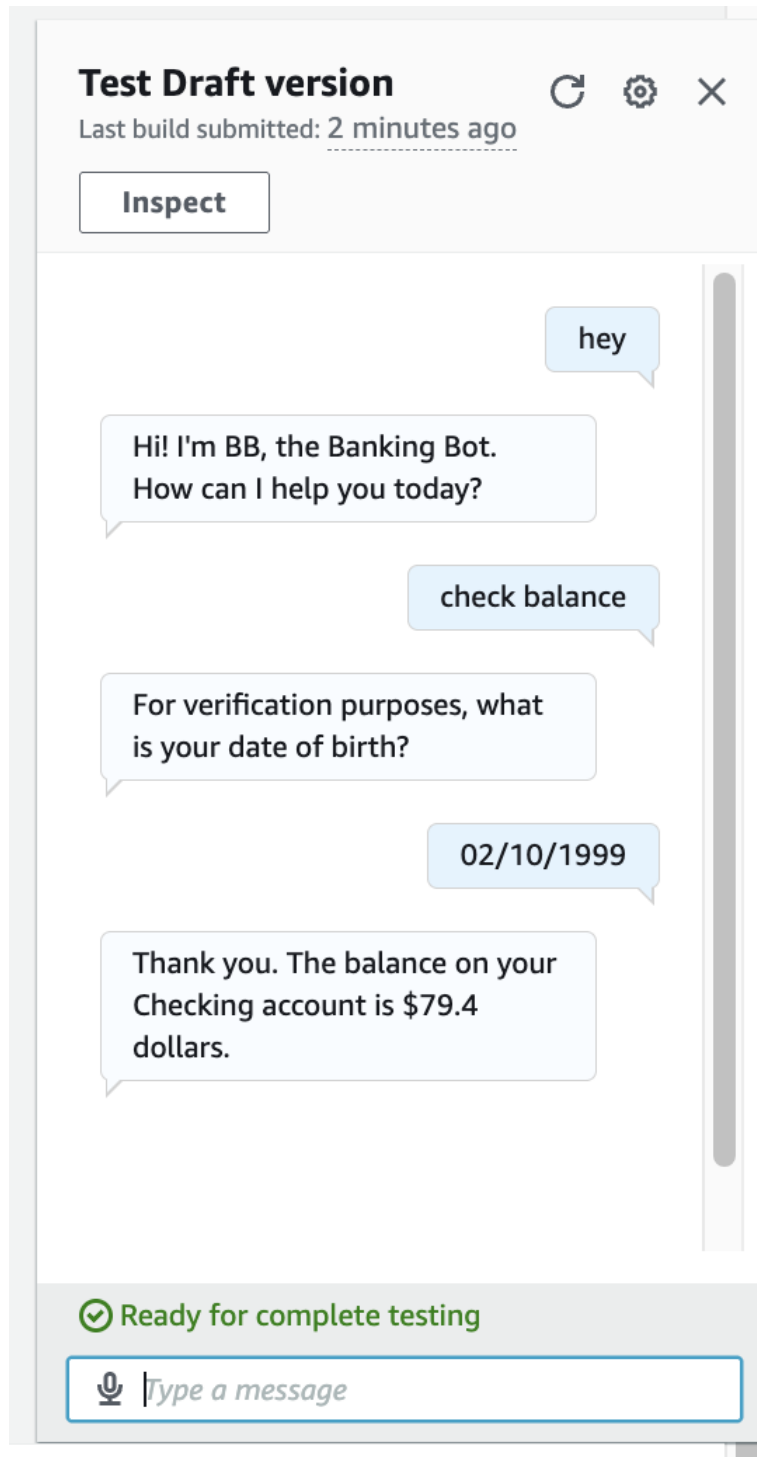


Figure 10: Final Result

14 Context Tags

Context tags are used in Amazon Lex to store and check specific information throughout a conversation, allowing users to avoid repeating certain details.

There are two types of context tags: Output context tags store details after an intent finishes for later use, like saving an account type from BalanceCheck. Input context tags check if details, like a date of birth, are already available.

I created a context tag called contextCheckBalance. This context tag was created in the intent I created a context tag called CheckBalance. This context tag was created in the intent CheckBalance. This tag stores information about checking balance

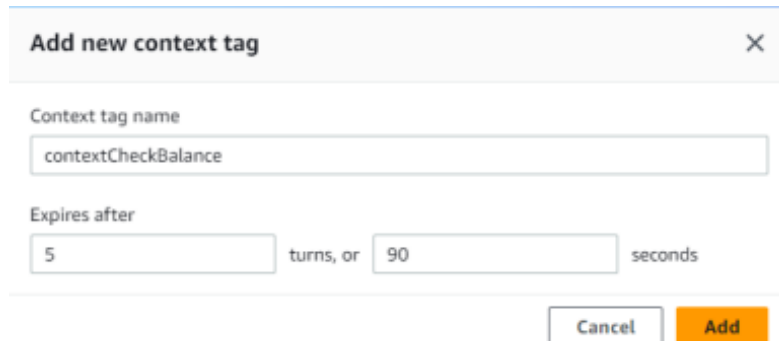


Figure 11: Context Tags

15 FollowUpCheckBalance

I created a new intent called FollowupCheckBalance. The purpose of this intent is to handle any additional queries or actions related to checking the user's balance, ensuring that any necessary follow-up information is gathered.

This intent is connected to my previous intent, CheckBalance, because it uses the context tag Balance to access stored balance information. This allows FollowupCheckBalance to efficiently handle follow-up queries without reasking for details

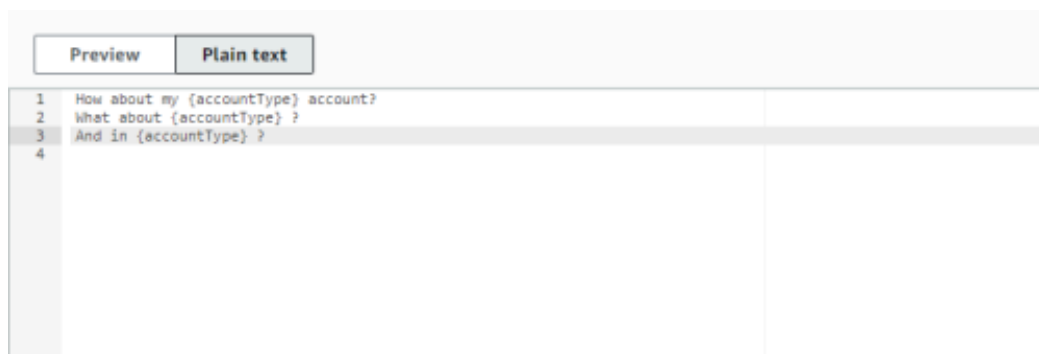


Figure 12: FollowUpCheckBalance

16 Input Context Tag

I created an input context, contextCheckBalance, that checks if the output context Balance is already available. This connection allows the system to use stored balance information without needing to ask the user again.

▼ 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

Cancel Update slot

Figure 13: Input Context Tag

17 The final result!

To see the context tags and the follow-up intent in action, I asked my chatbot for my balance using CheckBalance, then requested additional details, triggering FollowupCheckBalance. This demonstrated the seamless use of stored context.

If I had gone straight to trying to trigger FollowupCheckBalance without setting up any context, the intent would lack the necessary balance information. It might prompt the user for details that should have been previously gathered by CheckBalance.

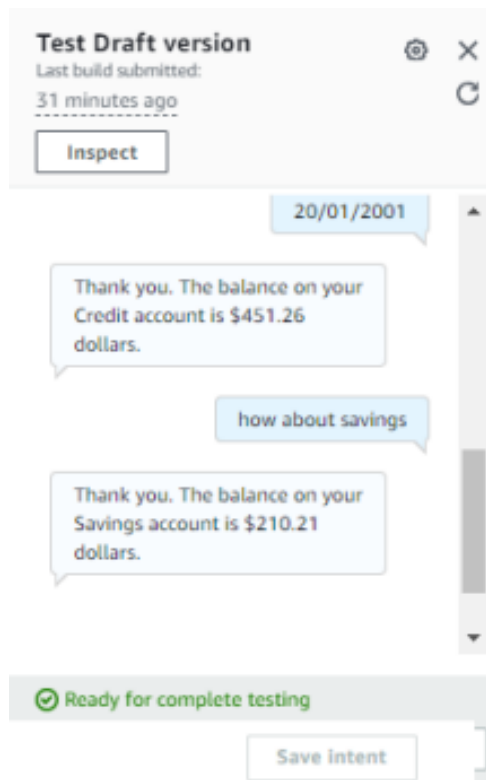


Figure 14: The final result

18 TransferFunds

I created an intent for my chatbot: TransferFunds, which will transfer money from a bank account type to another account type that the user provides in the conversation.

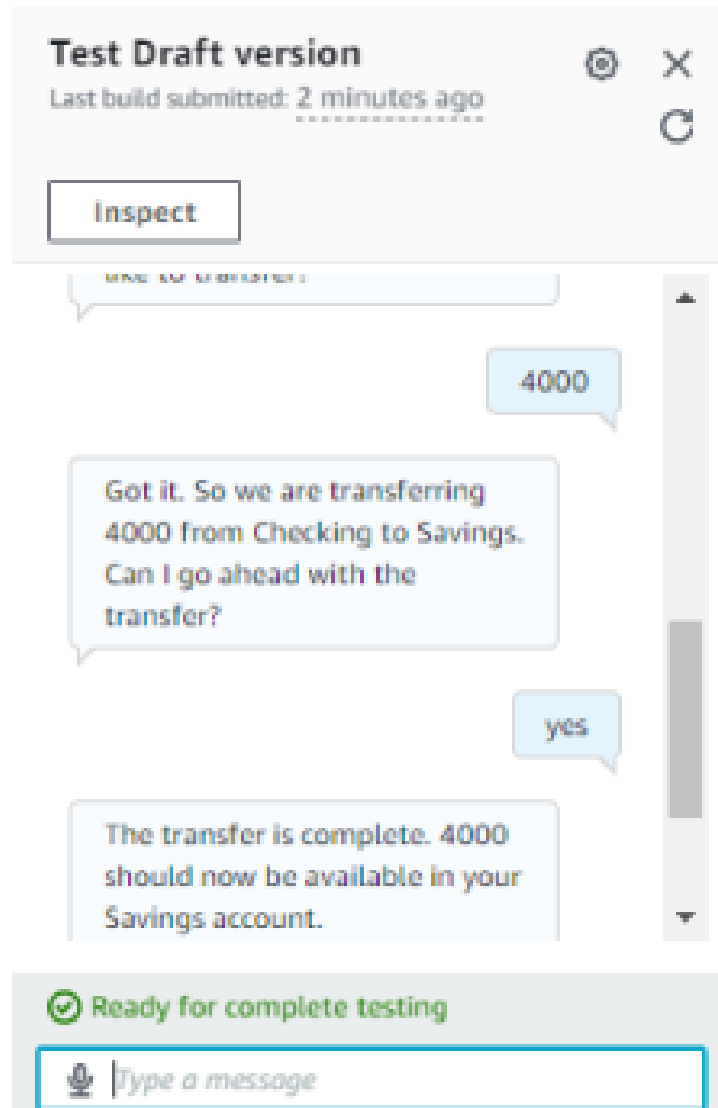


Figure 15: Tranfer Funds

19 Using multiple slots

For this intent, I had to use the same slot type twice. This is because the user provides two bank accounts; one account is the source where the chatbot will take money and the other account is the receiver where the chatbot will transfer the money.

I also learned how to create confirmation prompts, which are sent by the chatbot to verify with the user if they're sure they want the intent to be fulfilled or if they want to cancel it

20 Exploring Lex features

Lex also has a special conversation flow feature that provides a visualization between the user and the chatbot. The user can navigate with different dialog nodes and the flow gets created automatically as changes are made in the intent.

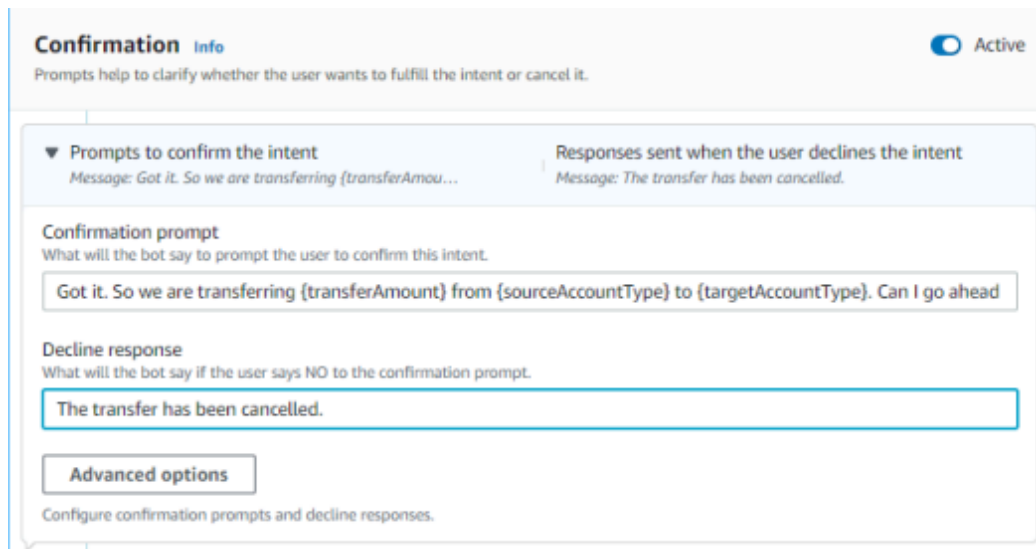


Figure 16: Using multiple slots

You could also set up your intent using a visual builder! A visual builder shows the user a graphic interface where they can see their intent and modify the conversation flow by simply dragging and dropping blocks.

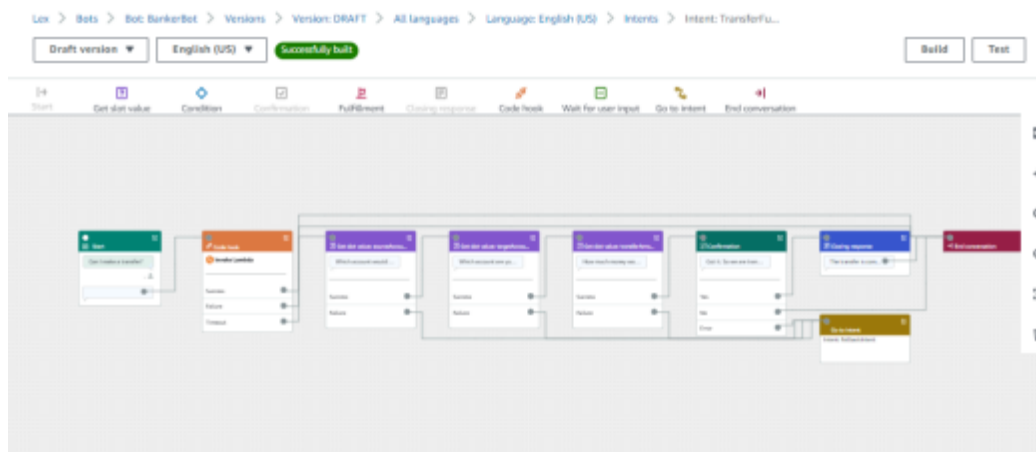


Figure 17: Exploring Lex features

21 AWS CloudFormation

AWS CloudFormation is a service that makes it fast and easy to set up AWS resources into a Stack; it's an Infrastructure as a Code service that uses a file that describes the resources and dependencies that need to be created.

I used CloudFormation to deploy a chatbot and all the depending resources (aliases, intents, Lambda functions) by creating a Stack

22 The final result!

Re-building my bot with CloudFormation took me less than 10 minutes to create. There was an error after I deployed my bot! I fixed this permission issue by creating a BankingBotEnglish function and associating it with TestBotAlias. I also added an invokeFunction permission to allow my TestBotAlias to communicate with it.

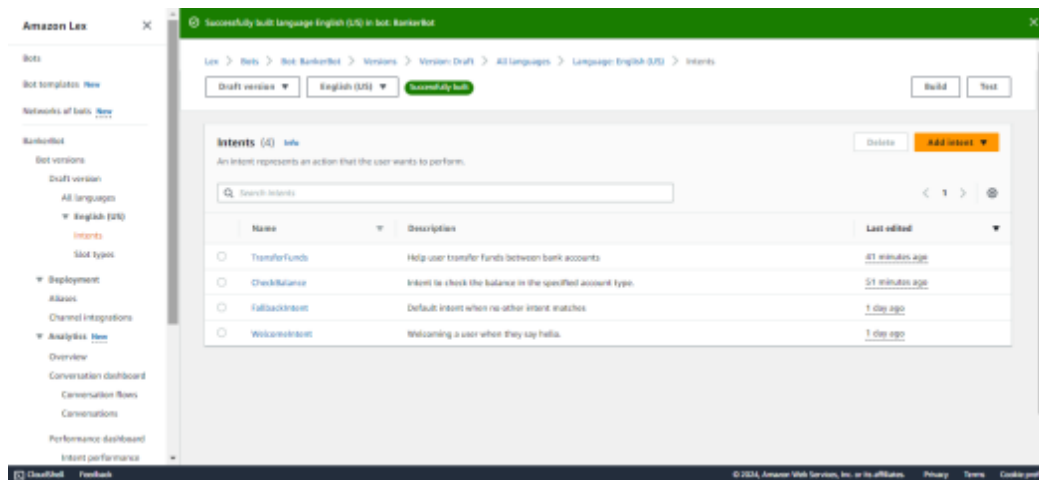


Figure 18: AWS CloudFormation

☐ AWS account
Grant permissions to another AWS account, user, or role.

☒ AWS service
Grant permissions to another AWS service.

☐ Function URL
Grant permissions to invoke your function through the function URL.

Service
The AWS service to grant permissions to.

Other

Statement ID
Enter a unique statement ID to differentiate this statement within the policy.

custom-permission-amazonlexchatbot

Principal
The service principal for this AWS service. [Learn more](#)

lexv2.amazonaws.com

Source ARN
The ARN for a resource. Find the ARN in the related service console.

arn:aws:lex:ca-central-1:799990344483:bot-alias/*

Action
Choose an action to allow.

lambda:InvokeFunction

Cancel

Save

Figure 19: The final result