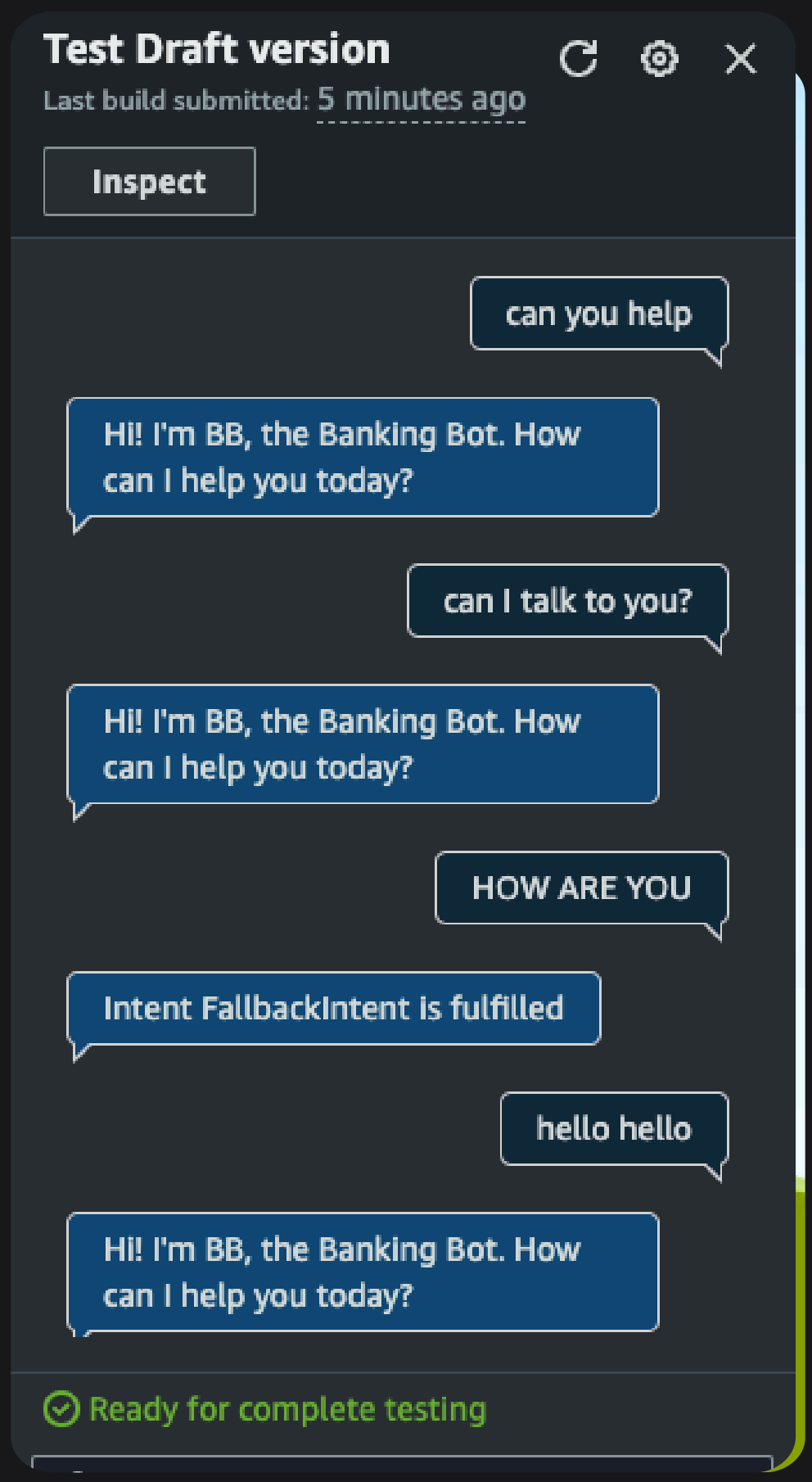
NEXTWORK

How I built a chatbot with Amazon Lex



Rasha M
@Badry2022
Rasha M.

TABLE OF CONTENTS

What is Amazon Lex?

What it does:

- **Amazon Lex builds conversational interfaces using voice and text.**

Why it's useful:

- It simplifies creating chatbots that understand natural language and can perform complex tasks.

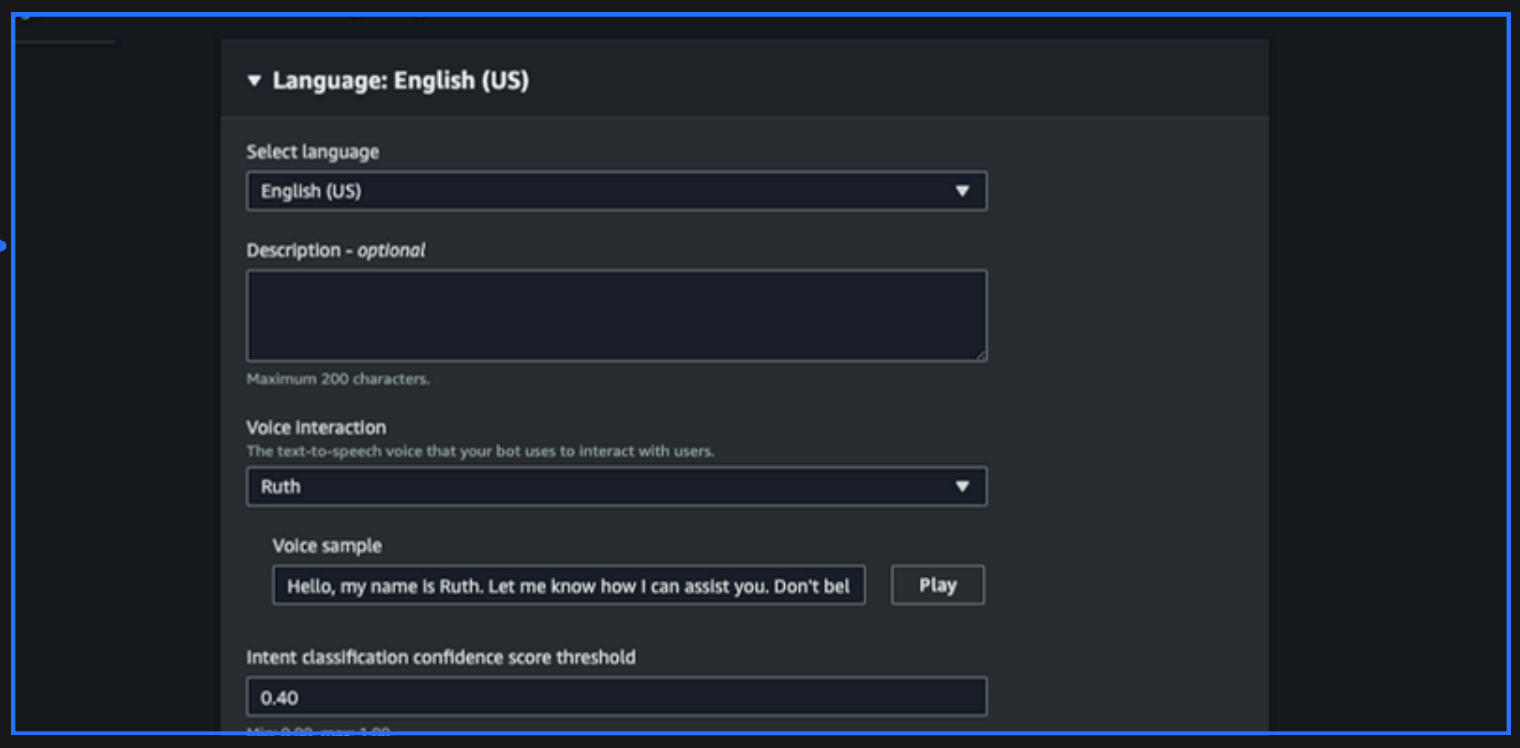
How I'm using it in today's project:

- In this project, I'm using Amazon Lex to create BankerBot, a chatbot that helps customers check their account balances and transfer money.

Set up a Lex chatbot

- I created BankerBot from scratch and used most default settings on Lex. My favorite voice option was Ruth .
- In terms of the intent classification confidence score, I kept the default value of 0.40. What this means for my chatbot is that it needs to be at least 40% confident in understanding user input to respond appropriately. This helps ensure the bot provides accurate and relevant responses.

Setting up my Lex chatbot...



Create an intent in Lex

Intents are specific goals users want to achieve with the chatbot, like checking a balance or making a transfer.

My first intent, WelcomeIntent, was created to greet users and assist them when they initiate a conversation.

To set up this intent:

- I Named it WelcomeIntent.
- Added sample utterances like "Hi," "Hello," and "I need help."
- Set a response message: "Hi! I'm BB, the Banking Bot. How can I help you today?"

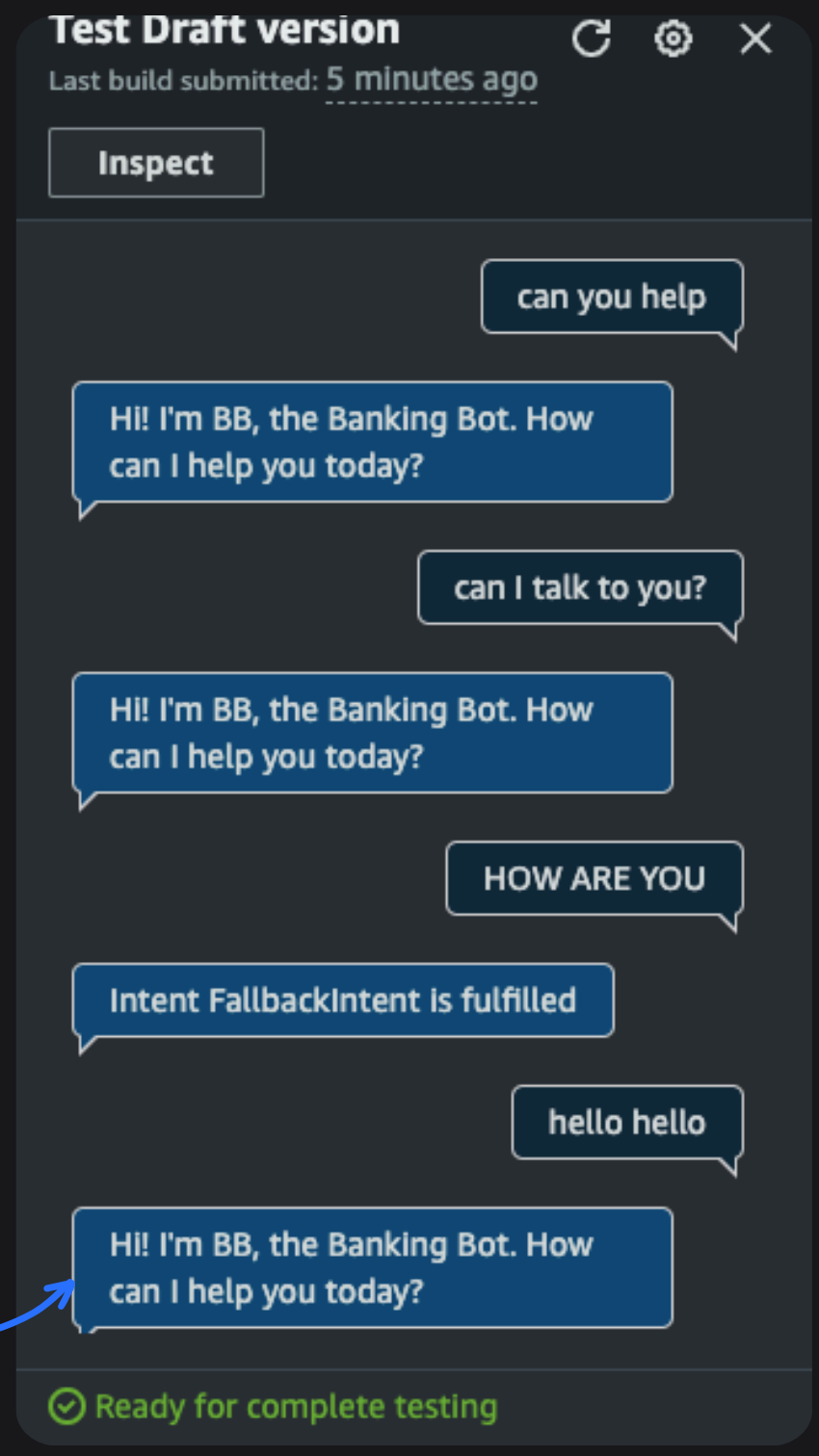
I launched and tested the chatbot, which could still respond if I entered:

- "Help me"
- "Help m,e"
- "Hiya"

However, the chatbot returned the error message “Intent FallbackIntent is fulfilled” when I entered:

- "How are you"
- Any weird questions

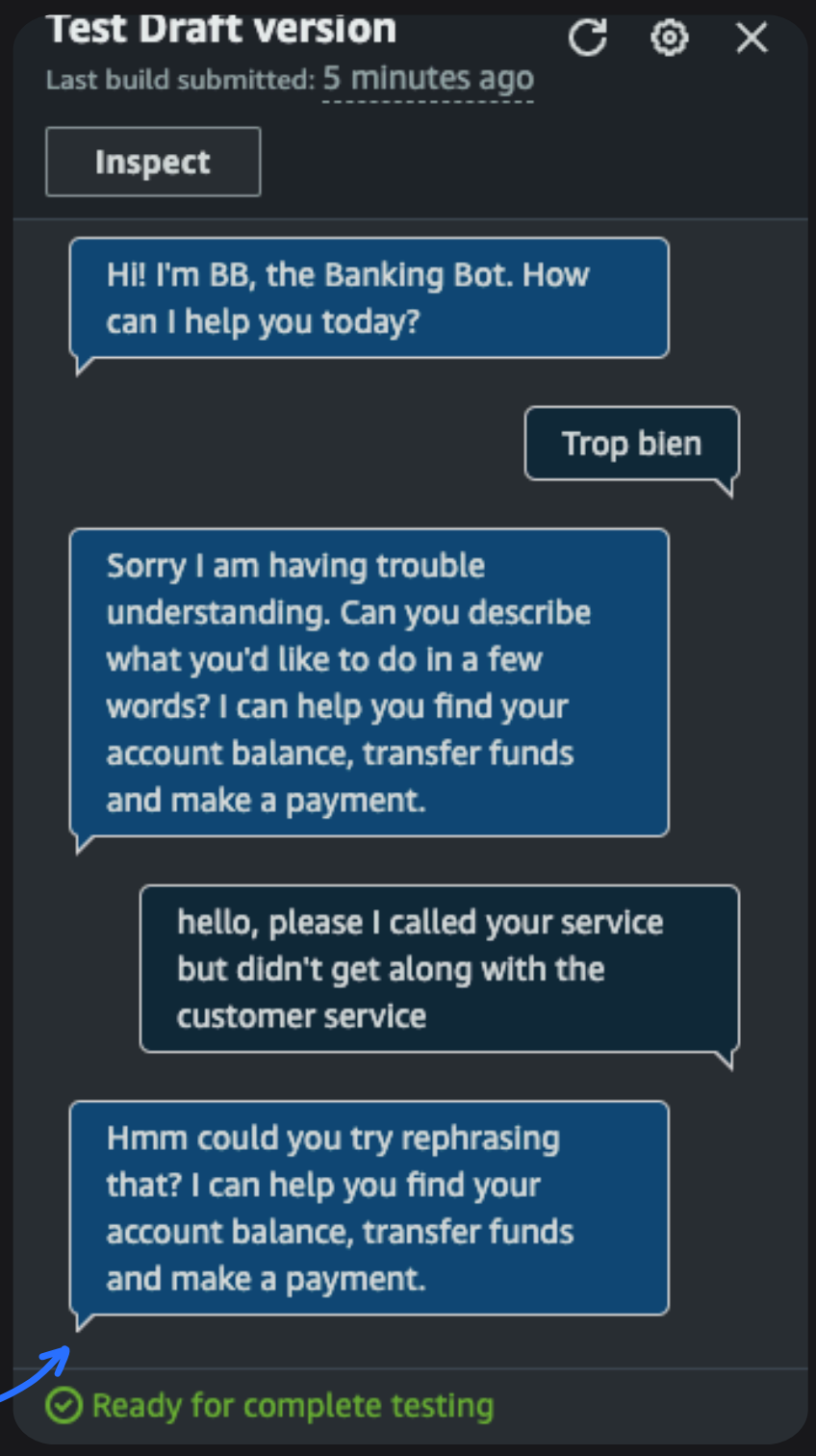
This error message occurred because the chatbot's confidence score for these inputs was below the 0.40 threshold.



My first test of the chatbot

Manage FallbackIntent

- FallbackIntent is a default intent in every chatbot that gets triggered when the confidence score is below the threshold (0.4 in our case).
- I wanted to configure FallbackIntent to provide clearer error messages when the chatbot doesn't understand the user's input.
- To configure FallbackIntent, I updated the closing response message to guide users on how to phrase their requests and added more context.
- I also added variations! What this means for an end user is a more dynamic and conversational experience when the chatbot doesn't understand their input.



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

My Key Learnings

01

- Amazon Lex is a service that enables anyone to build conversational interfaces, or chatbots, using voice and text
- It's a no-code tool, which means, you don't need to be an AI expert to create a chatbot

02

Intents in Amazon Lex represent the goals or actions that users want to achieve through their interactions with the chatbot.

03

AI/ML is used in Amazon Lex to train the chatbot on recognizing intents and processing user inputs accurately.

04

FallbackIntent is employed for handling user inputs that the chatbot cannot confidently classify based on its training, ensuring a fallback response is provided for clarity.

05

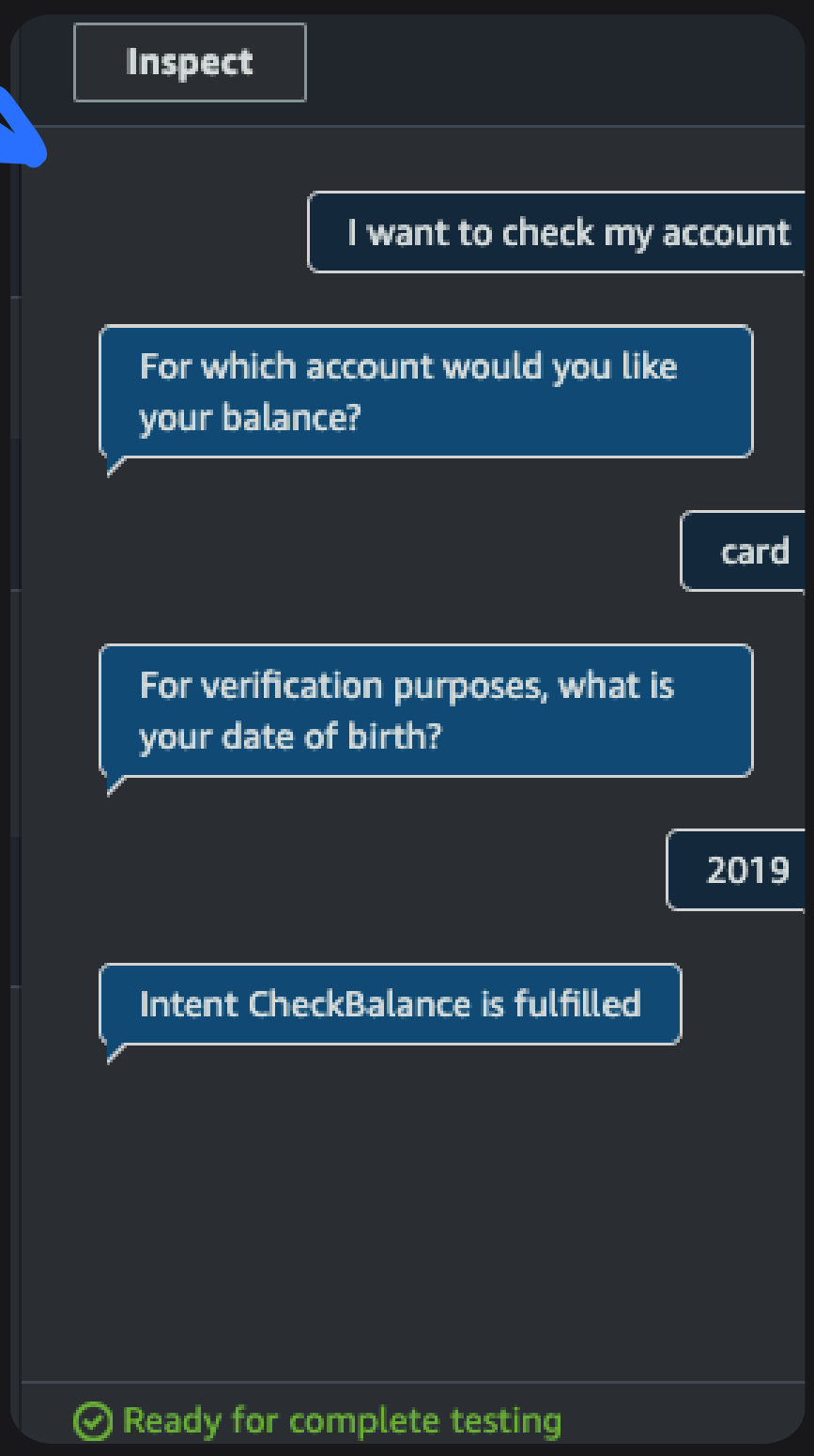
Creating a chatbot with Amazon Lex highlighted how accessible it is for businesses to develop tailored conversational interfaces, enhancing customer interaction or service delivery.

Final thoughts...

- This project took me approximately 1 hour to complete, and documentation took me about 1 hour.
- Delete EVERYTHING at the end! Let's keep this project free :)
- One thing I didn't expect was how straightforward it was to set up a functional chatbot using Amazon Lex.
- What's next? In the next phase of this project, I'll be adding a new flow that lets users check their account balances and verify their identity with their birthday. I'll be creating a custom slot type to handle the different bank account types. Excited to bring this feature to life and make our BankerBot smarter and more interactive! 🚀👀

Part 2 : How I built a chatbot with Amazon Lex

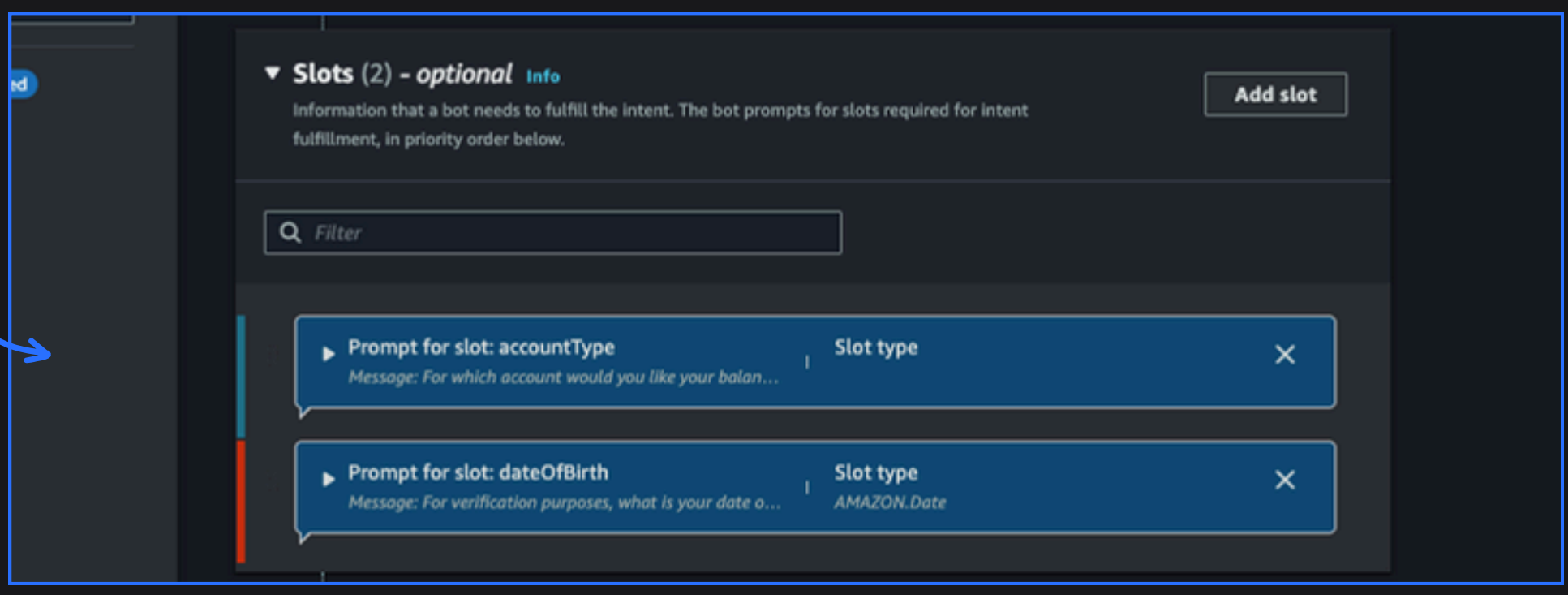
featuring custom slots!



Create custom slots

- Slots are pieces of information that a chatbot needs to complete a user's request, like filling in blanks on a form or a date of birth.
- Amazon Lex supports built-in slot types that define how data in the slot is recognized and handled. You can create slots of these types in your intents. This eliminates the need to create enumeration values for commonly used slot data such as date, time, and location.
- In this project, I created a custom slot to represent different bank account types: Checking, Credit, and Savings.
- I then associated the custom slot with a new intent, CheckBalance, which allows users to inquire about the balance of a specific type of bank account, ensuring accurate and efficient responses.

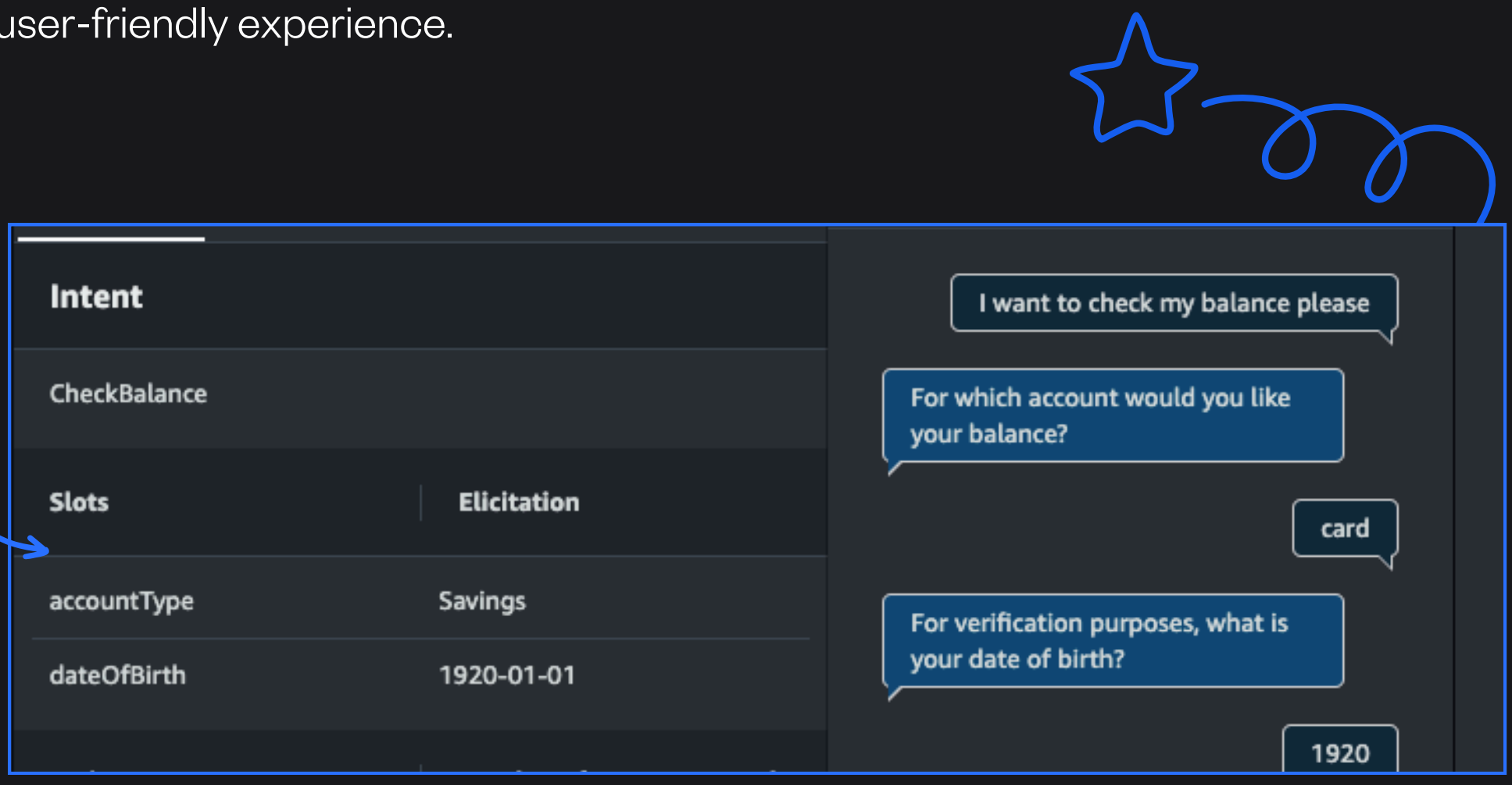
My custom slots



Simplifying the user experience

- I included slot values in some of the utterances (user inputs) for this intent too. For example, utterances like "What's the balance in my {accountType} account?" and "How much do I have in {accountType}?" include the slot value {accountType}.
- By adding custom slots in the utterance, Amazon Lex can automatically fill in the slot information if it matches the user's input. This saves the bot from having to ask for the account type separately, streamlining the interaction and saving the user from needing to declare it explicitly. This results in a more efficient and user-friendly experience.

Slot values getting recognised in a conversation



My Key Learnings

01 Slots are pieces of information that a chatbot needs to complete a user's request, similar to blanks in a form.

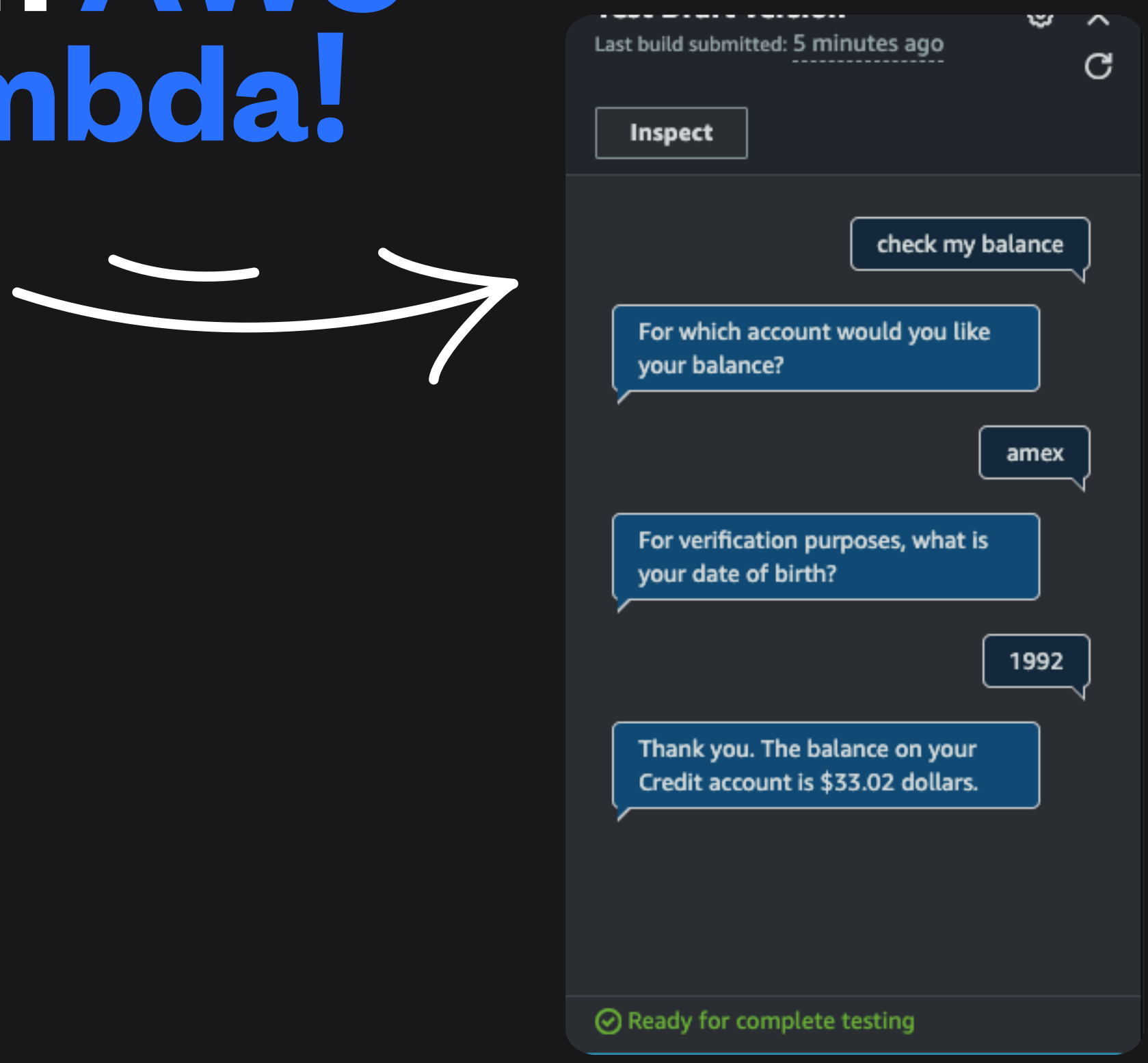
02 Slot types define the kind of data a slot can hold, such as dates, times, or custom values. Custom slot types are user-defined categories specific to particular needs.

03 I used a custom slot type in this project to represent different bank account types (Checking, Credit, Savings) to ensure accurate recognition and handling of these specific categories.

04 I parsed my custom slot in the utterance by including the slot value {accountType} in a prompt. For example, in "What's the balance in my {accountType} account?", Lex recognizes and fills in the account type from the user's input.

05 I also learned how to add and use another slot for the user's date of birth (dateOfBirth). This helps with verifying the user's identity before providing sensitive information like account balances. By incorporating the AMAZON.Date slot type for dateOfBirth, the chatbot can ask for and validate the user's birth date seamlessly, ensuring an extra layer of security in the interaction. I have the same interaction when I call my banker, that's amazing!

Part 3 : How I connected my chatbot with AWS Lambda!



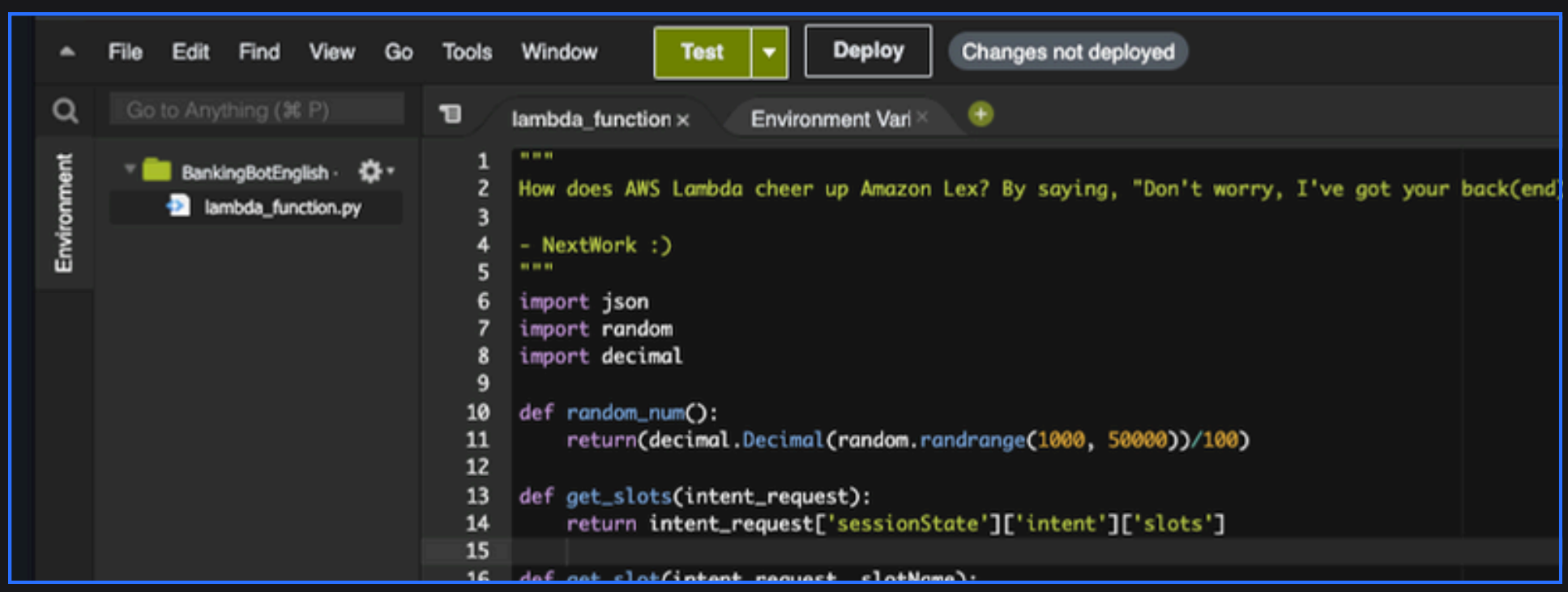
Using AWS Lamba

AWS Lambda :

- A Lambda function's execution role is an AWS Identity and Access Management (IAM) role that grants the function permission to access AWS services and resources. This is sometimes called Functions-as-a-Service (FaaS).
- AWS Lambda runs code without managing servers, scaling automatically.
- Useful for serverless applications, automation, and data processing.

In this project :

- Created a Lambda function to handle CheckBalance for BankerBot.
- BankingBotEnglish NextWork.py generates and returns a random bank balance.
- This function adds dynamic data retrieval that the chatbot can't perform on its own.



Connecting Lambda with Lex

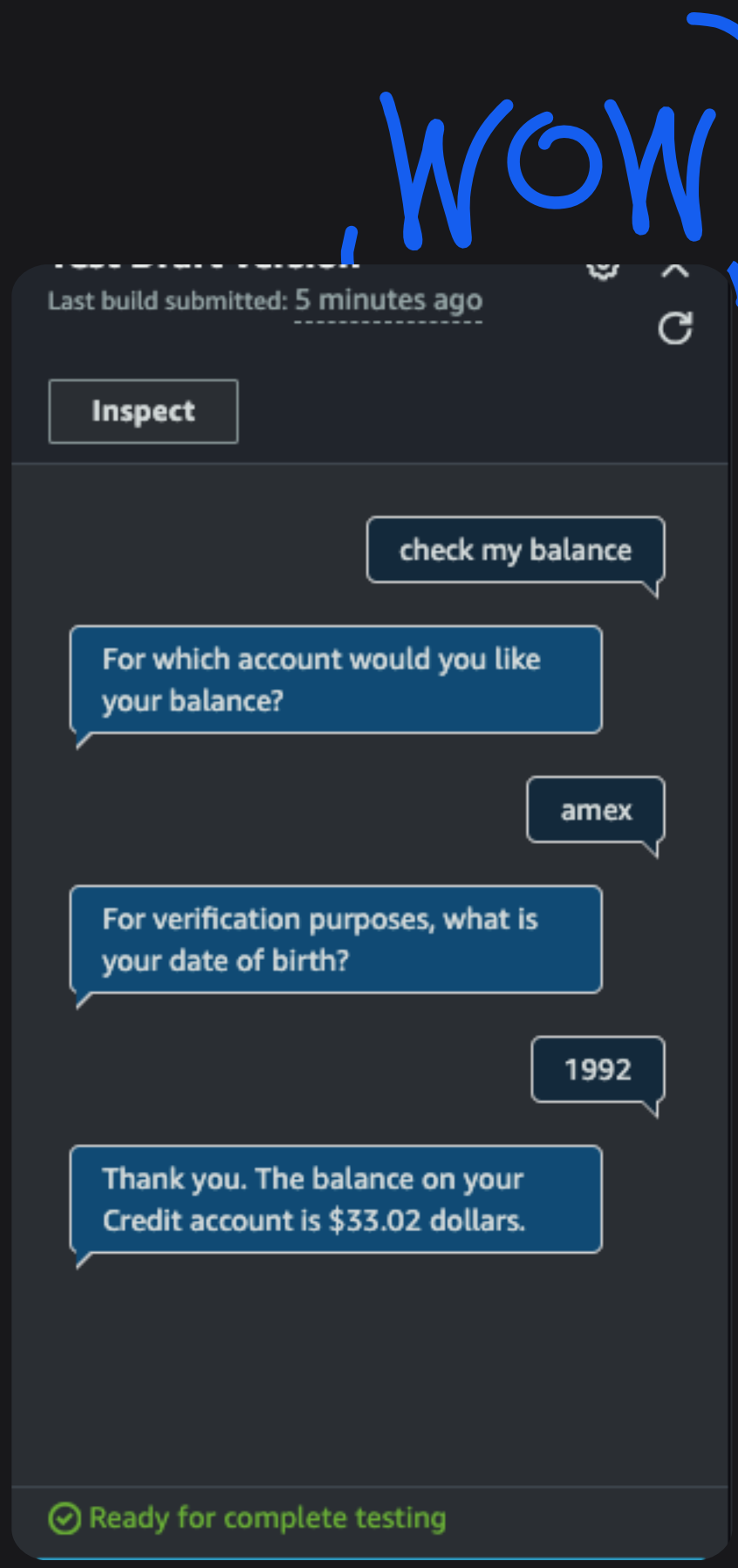
Step 1 To connect Lambda with my chatbot alias, I selected the Lambda function BankingBotEnglish in the Lex console under the TestBotAlias settings and saved the configuration.

Step 2 Another intent setting to configure is code hooks.

A code hook is a mechanism that allows the integration of custom logic or external services into a chatbot's workflow, enabling more sophisticated interactions.

In this project, I had to use code hooks because they facilitate the integration of the AWS Lambda function BankingBotEnglish with the CheckBalance intent in my Lex chatbot. This integration is necessary to fetch and return dynamic data, such as random bank balance figures, which the chatbot cannot generate on its own.

After connecting Lambda with my Lex bot and enabling the code hook, my chatbot could successfully retrieve and provide users with random bank balance figures in response to inquiries about their account balances.



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

My Key Learnings

01 AWS Lambda is a service by Amazon that lets you run code without managing servers. It scales automatically and is used when you need to process data, connect to databases, or perform calculations based on user input.

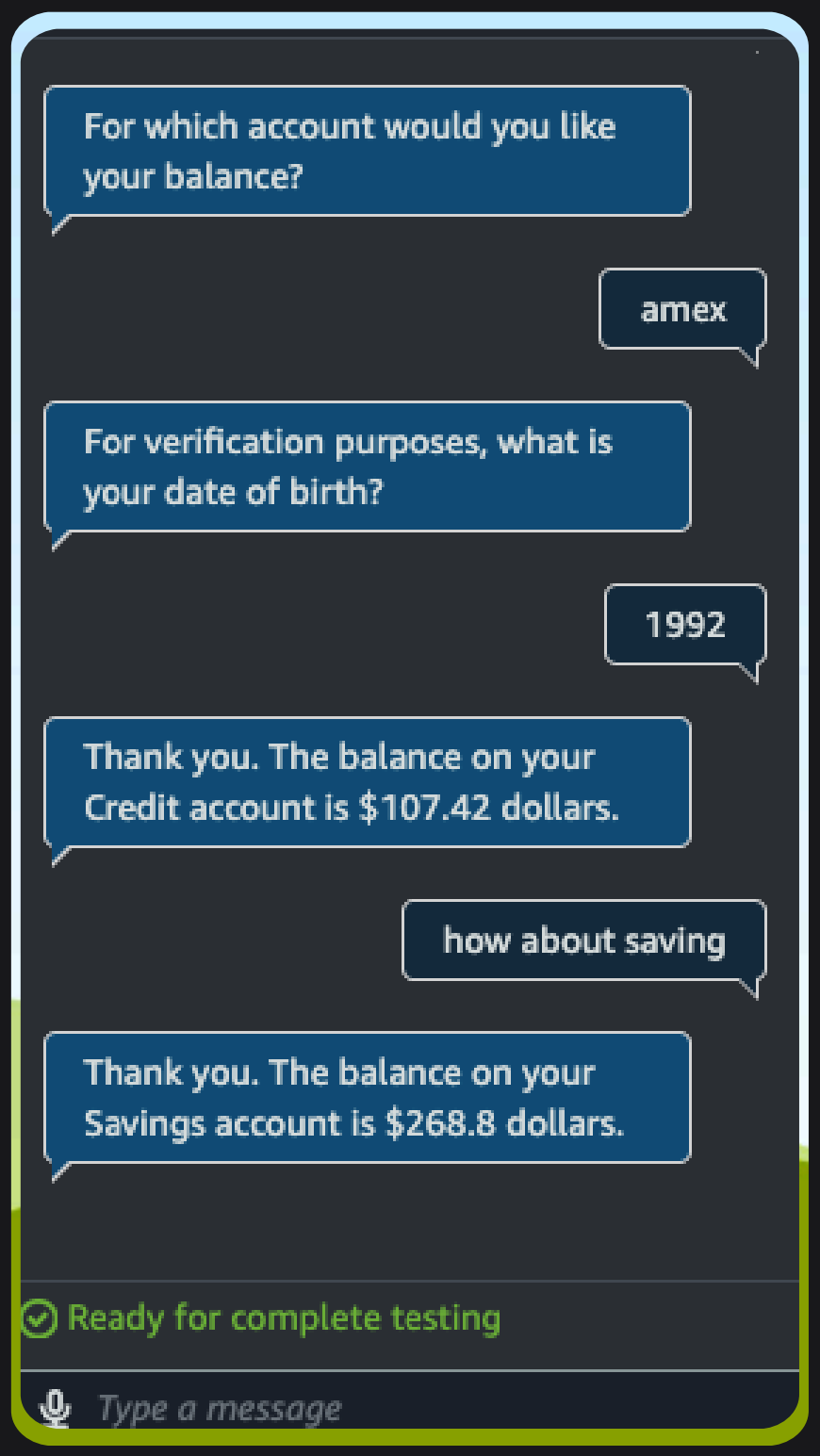
02 You connect Amazon Lex with AWS Lambda when your chatbot needs to perform tasks beyond simple responses, such as accessing databases, performing calculations, or integrating with external APIs.

03 Set up a Lambda function in AWS Lambda console with a python code, In Amazon Lex console, configure your bot to use this Lambda function for specific intents

04 Understanding how to integrate AWS services like Amazon Lex and AWS Lambda has shown me the power of serverless architecture in building scalable and efficient applications.

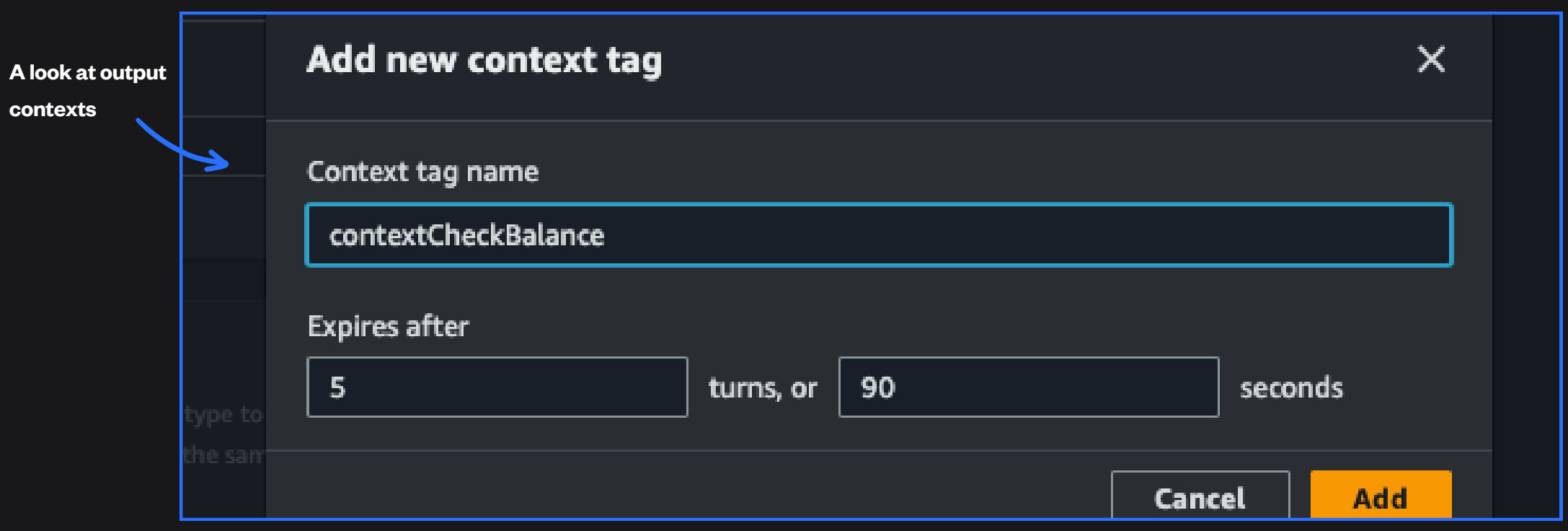
Part 4: How I built a chatbot with Amazon Lex

that can remember user info!



Context Tags

- Context tags in Amazon Lex manage information throughout a conversation, improving user experience by avoiding repetitive questions.
- There are two types of context tags :
- Output context tags: Store information after an intent is fulfilled, enabling other intents to access it later. For instance, saving account type details after a balance check.
- Input context tags: Check for specific information before activating an intent, ensuring that already provided details, like a user's birthday, aren't requested again.
- I created an output context tag called "contextCheckBalance" in the CheckBalance intent.



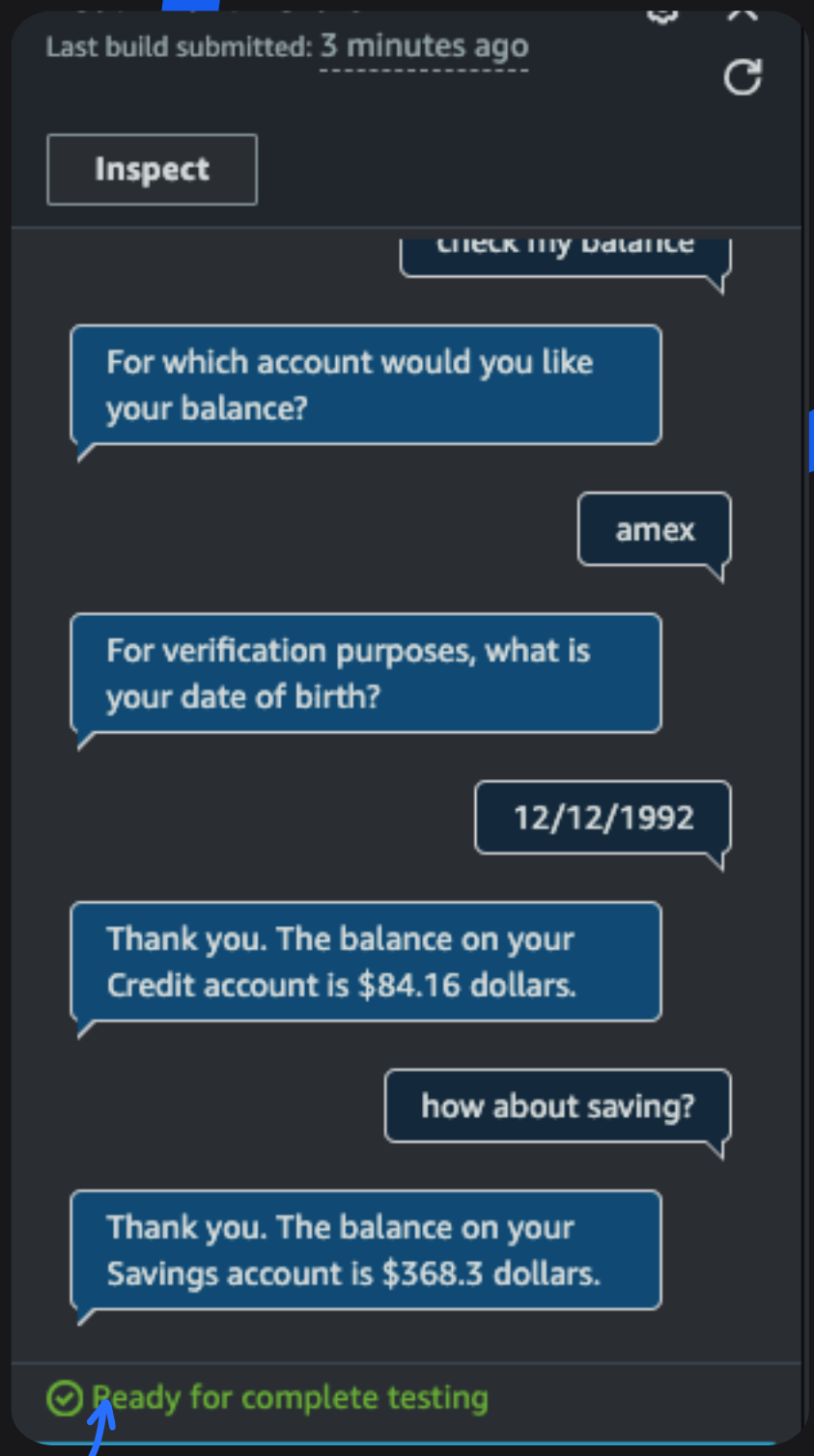
A Follow-Up Intent

- I created a new intent called FollowupCheckBalance. The purpose of this intent is to allow users to request a follow-up account balance check without re-entering previously provided information, such as their date of birth.
- This intent is related to the previous intent I made, CheckBalance, because FollowupCheckBalance is triggered after CheckBalance when the user wants to check another account balance. It utilizes the context stored from CheckBalance to avoid asking for the date of birth again.
- I created an input context, contextCheckBalance, that is connected to the output context tag from CheckBalance, allowing it to access stored information, such as the date of birth, ensuring a seamless user experience without redundant prompts.

A look at input contexts

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 started by asking the bot to "check my balance", "amex" and provided my birthday when prompted.
- Then, I said "how about saving" to trigger the follow-up intent for checking another account without needing to repeat my birthday.
- If I had gone straight to trying to trigger FollowUpCheckBalance without setting up any context...The bot wouldn't answer directly to "how about saving" because it wouldn't have my birthday information.
- It might ask for clarification or request to start over with "check balance" again.



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

My Key Learnings

01

Context tags are used in Amazon Lex to store and retrieve specific information across different conversation parts. They help users avoid repeating information by remembering details from previous interactions.

02

Input context tags check for existing information before an intent, while output context tags store information for later use in the conversation.

03

I created the input context in FollowupCheckBalance by specifying 'contextCheckBalance' as the input context tag, ensuring the intent retrieves previously stored user details, such as the date of birth.

04

FollowupCheckBalance requires the presence of the 'contextCheckBalance' output context from a previous CheckBalance intent to retrieve stored user details such as the date of birth, thus it cannot function successfully without prior context

05

I've learned the importance of designing conversational flows that anticipate user needs and preferences, ensuring a smoother and more personalized interaction with the chatbot.

Final thoughts...

- This project took me approximately 2 hours to complete. Writing documentation took me an additional 1 hour.

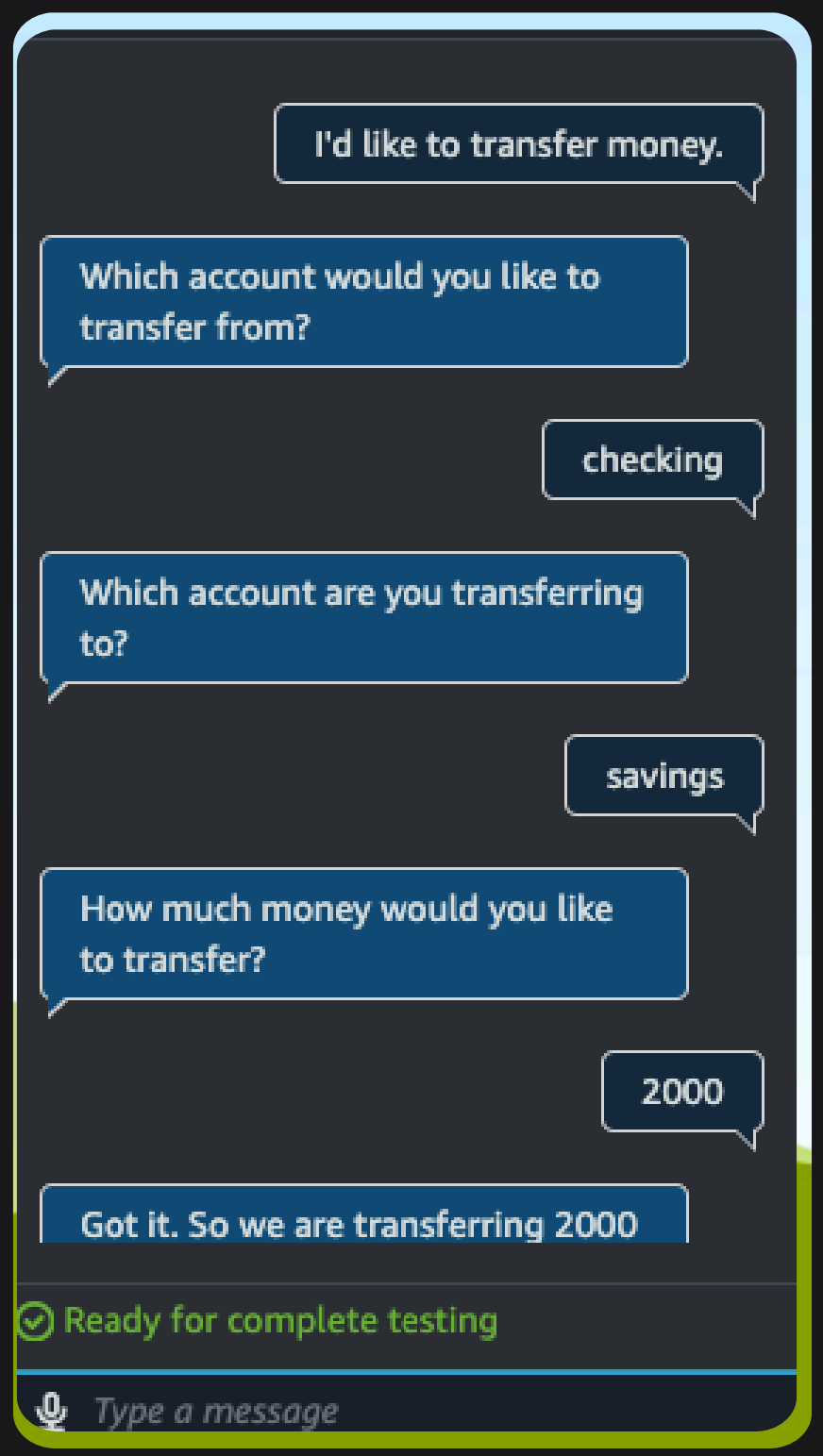
- Delete EVERYTHING at the end! Let's keep this project free :)

- One thing I didn't expect was...

- In the next phase of this project, I'm excited to level up my Lex bot one more time by creating an intent that can help users transfer funds between accounts. I'm also using AWS CloudFormation to recreate my bot in seconds!

Part 5 : How I built a chatbot with Amazon Lex

with multiple slots! 🎰



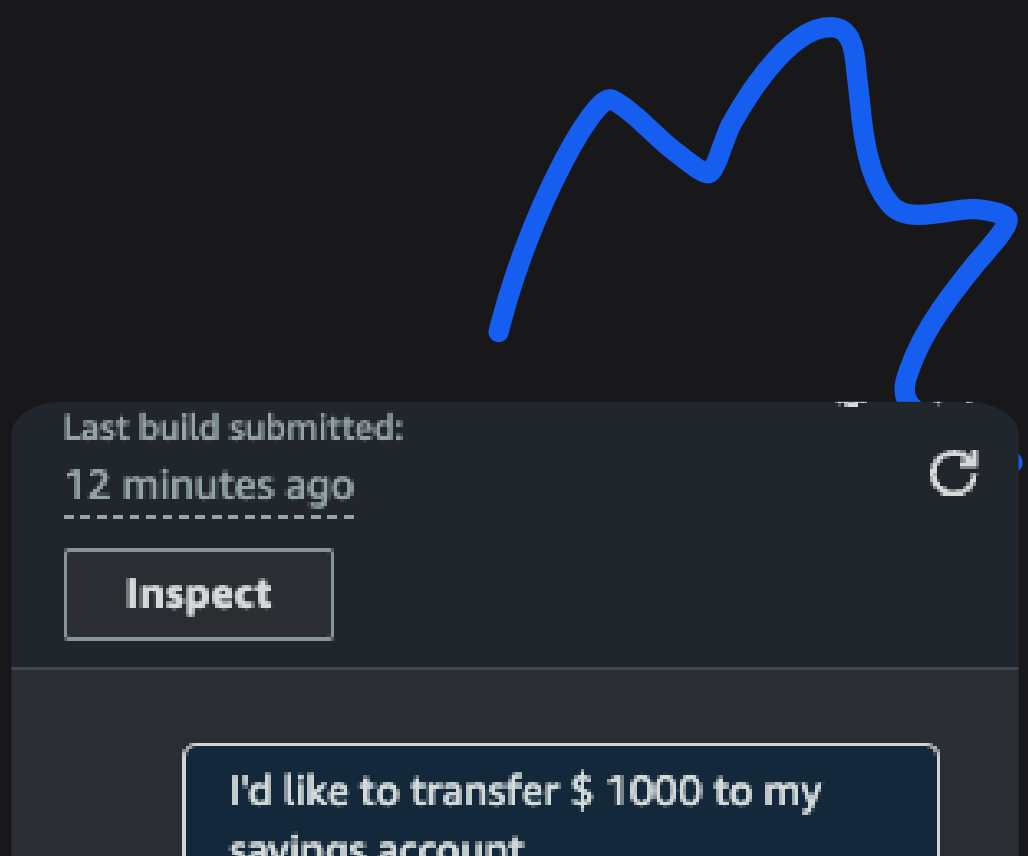
More slots!

- Slots in Amazon Lex serve as placeholders for specific user-provided information essential for executing actions or fulfilling requests within a chatbot interaction. For example, in BankerBot, the accountType slot categorizes different types of bank accounts such as Checking, Savings, or Credit.

- The final intent, TransferFunds, enables users to initiate and complete transfers of funds between specified accounts. This intent facilitates seamless transactions within the BankerBot, enhancing user convenience and banking functionality.

- In the TransferFunds intent, I utilized the accountType slot type twice : once for sourceAccountType and again for targetAccountType. This repetition ensures that the bot accurately identifies both the source from which funds will be transferred and the destination account.

- I also learnt how to create confirmation prompts. Confirmation prompts in TransferFunds are designed to verify transaction details before finalizing the transfer. They repeat back the specified transferAmount, sourceAccountType, and targetAccountType to the user, confirming the intended action. For instance, "Got it. So we are transferring {transferAmount} from {sourceAccountType} to {targetAccountType}. Can I go ahead with the transfer?" This ensures clarity and reduces the risk of errors in financial transactions.



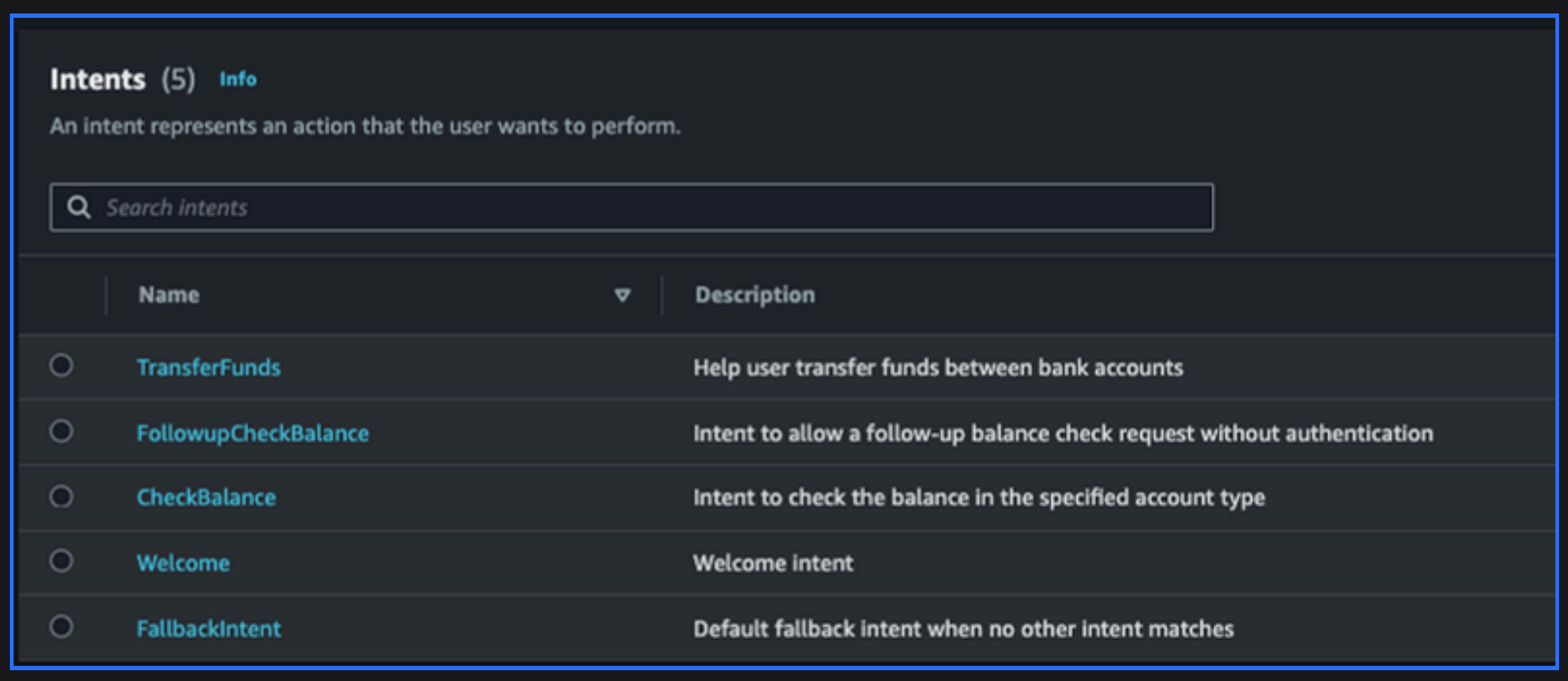
A conversation demonstrating the two slots and the confirmation prompts in action!

A LITTLE EXTRA...

Deploying with CloudFormation

- AWS CloudFormation is a service that automates the provisioning and management of your AWS infrastructure.
- As an extension to this project, I learned how to deploy the entire BankerBot using a single CloudFormation stack.
- Doing this took me less time and effort compared to manually creating each resource.
- Something I learned from deploying with CloudFormation was the importance of infrastructure as code. It allows for a repeatable and consistent way to build and deploy my chatbot environment, but the YAML File isn't easy to write if you're not a programmer.

CloudFront deployed this for me!



My Key Learnings

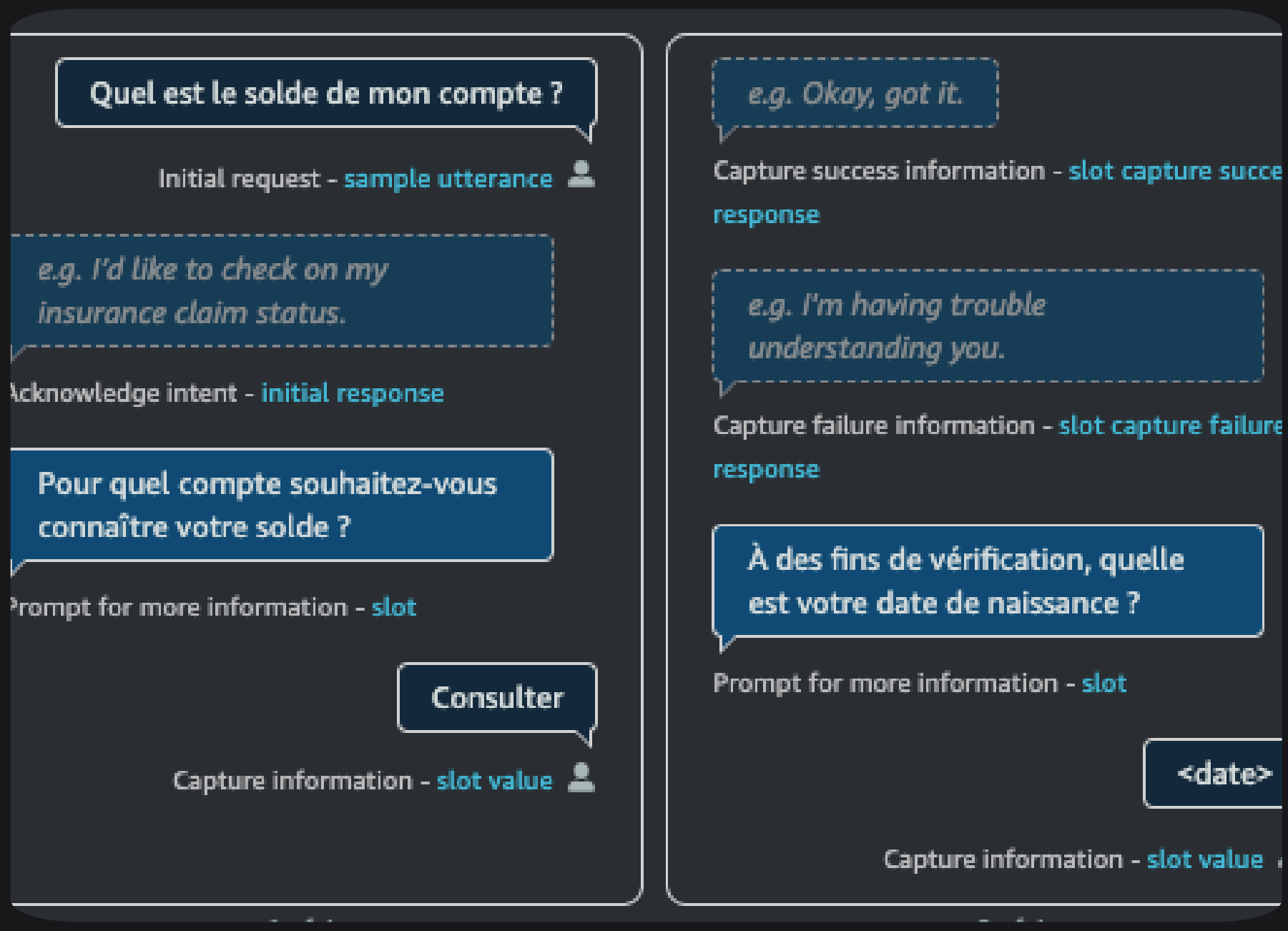
01 I used the accountType slot type twice to specify both the source and target accounts for transfers, ensuring clear distinctions for each step of the transaction.

02 Confirmation prompts are messages that verify user intent by restating information for confirmation, helping prevent errors. They ensure that the bot proceeds with accurate actions based on user input.

03 CloudFormation is an AWS service that automates the deployment of resources using templates. It created the entire BankerBot infrastructure, including intents, slots, and other configurations.

04 I learned the importance of using clear, distinct slot names and how structured automation can enhance deployment efficiency and consistency.

How I built a French chatbot with Amazon Lex



Step #1: Set up your Lex chatbot

Creation method

- **Create a blank bot** — Create a basic bot with no preconfigured languages, intents, and slot types.
- **Start with an example** — An example bot has preconfigured languages, intents, and slot types. You can change these settings.
- **Start with transcripts** — Automatically generate intents from conversation transcripts that you upload. Only English (US) language is available when starting with a transcript.

Bot configuration

Bot name

BankerBot

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

Description - *optional*

This description appears on bot list page. It can help you identify the purpose of your bot.

Un bot qui aide les clients à verifier leurs comptes et faire des transferts .

▼ Language: French (CA)

Select language

French (CA)

Description - *optional*

Maximum 200 characters.

Voice interaction

The text-to-speech voice that your bot uses to interact with users.

Gabrielle

Voice sample

Bonjour, je m'appelle Gabrielle. Comment puis-je vous aider?

Play

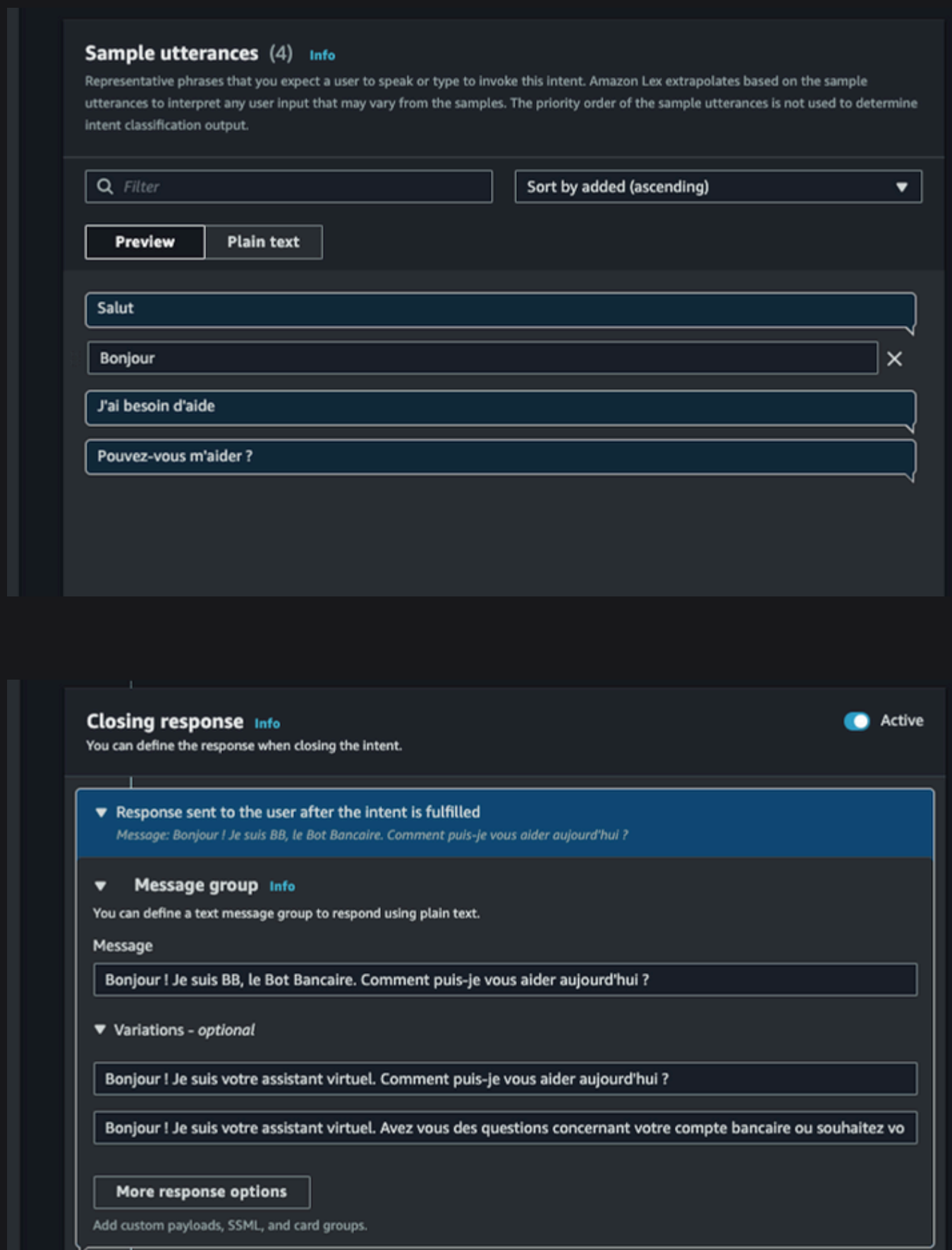
Intent classification confidence score threshold

0.40

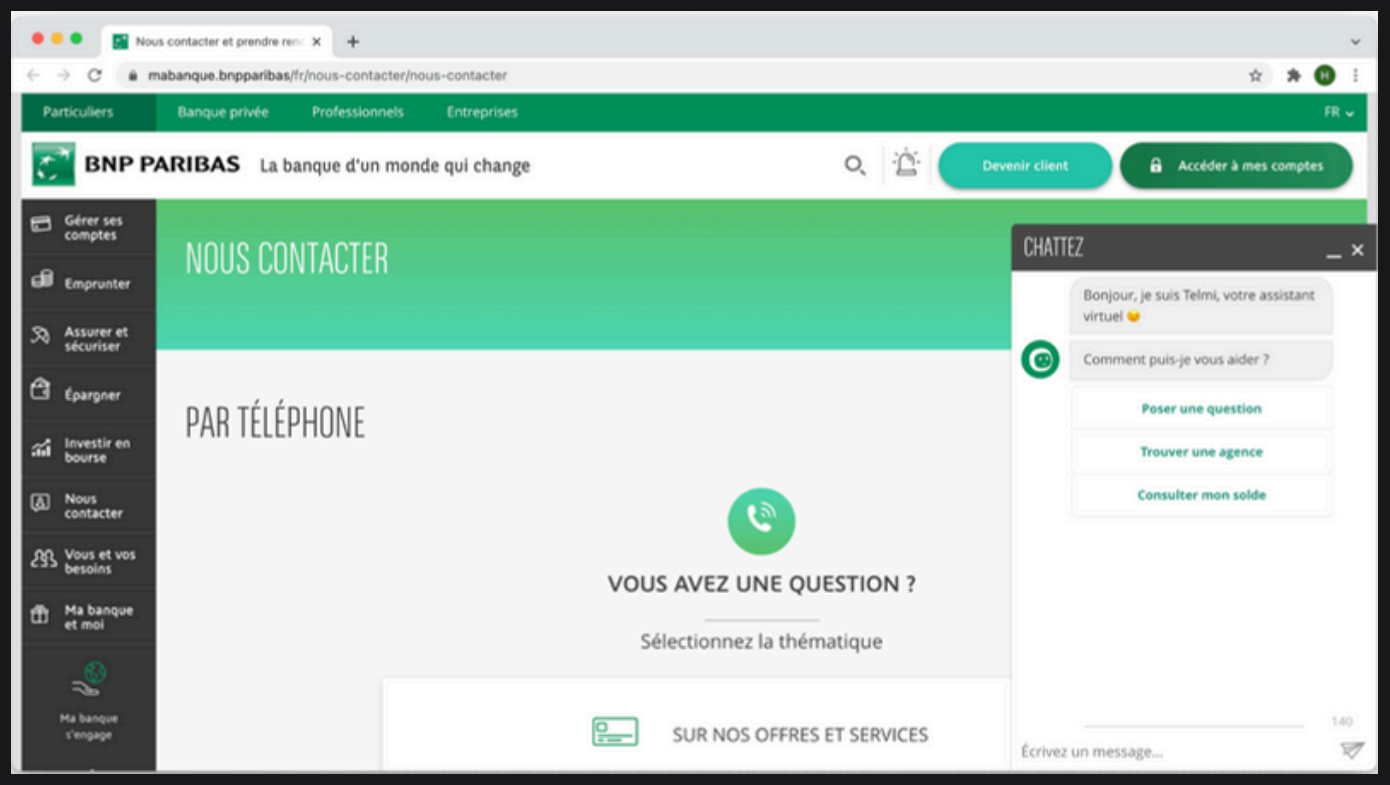
Min: 0.00, max: 1.00.

If French, only one voice is available ...

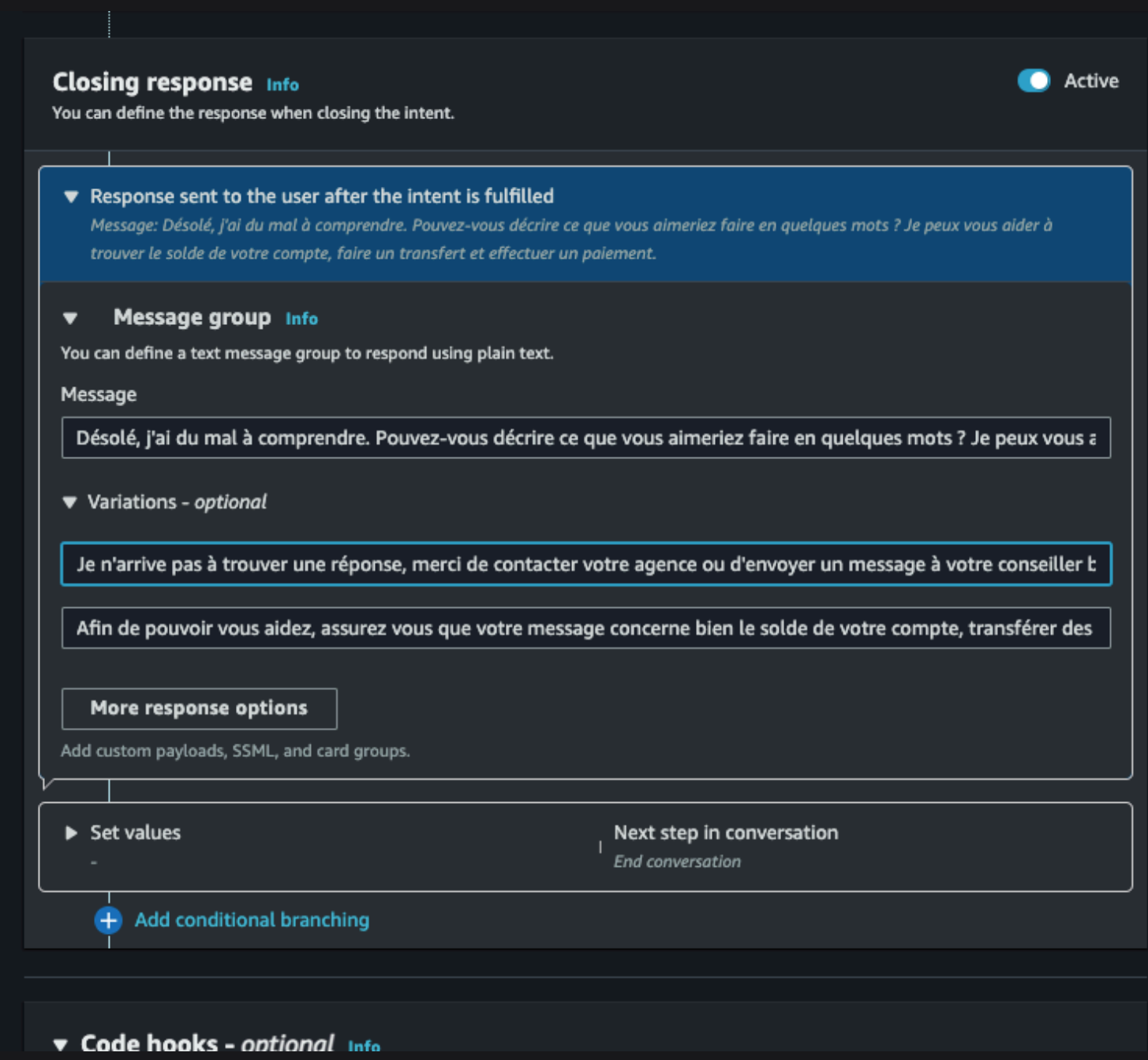
💬 Step #2: Create your first intent



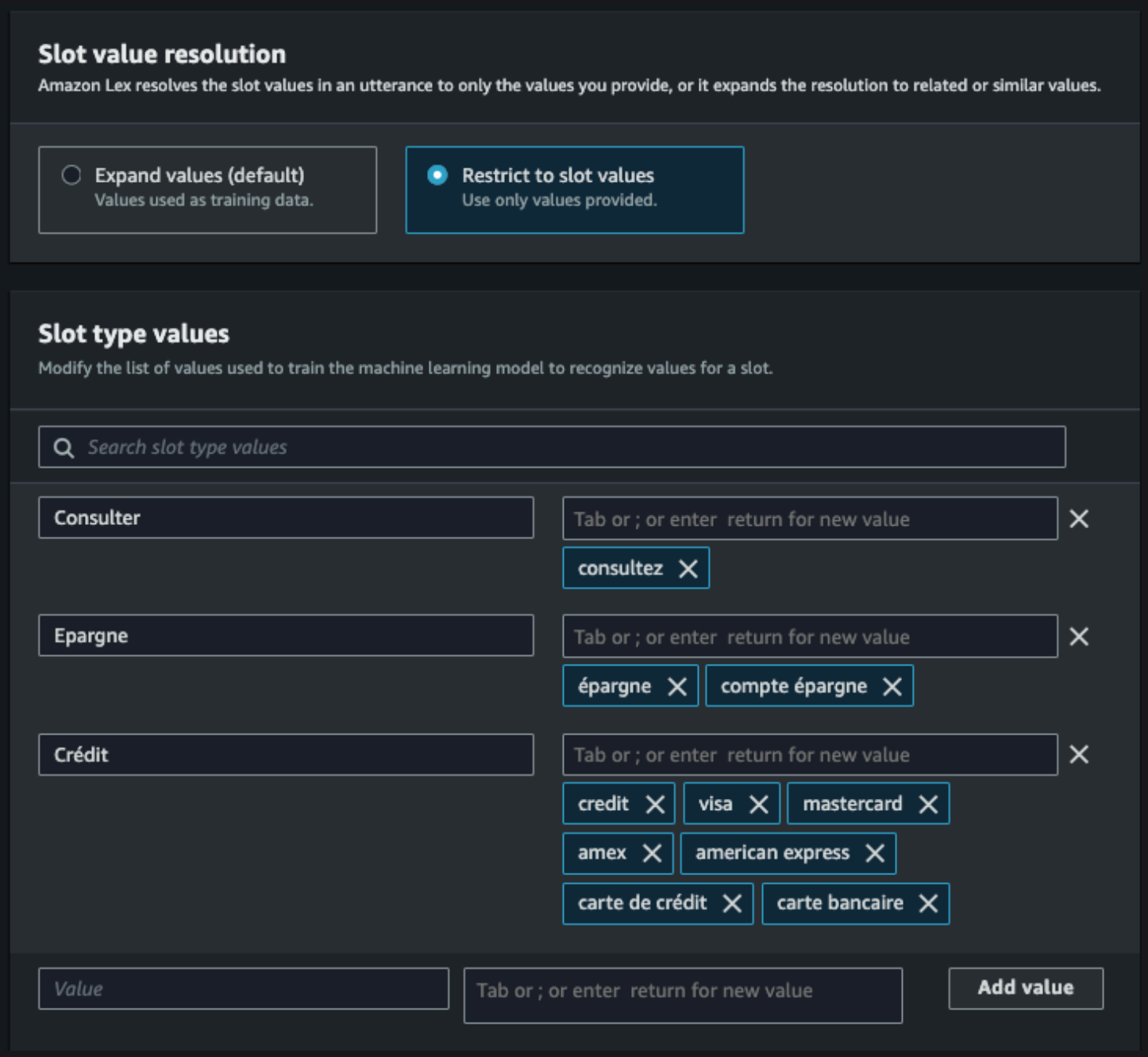
To be honest, my bank has a mobile application, but I never use the chatbot available at the official webpage ...



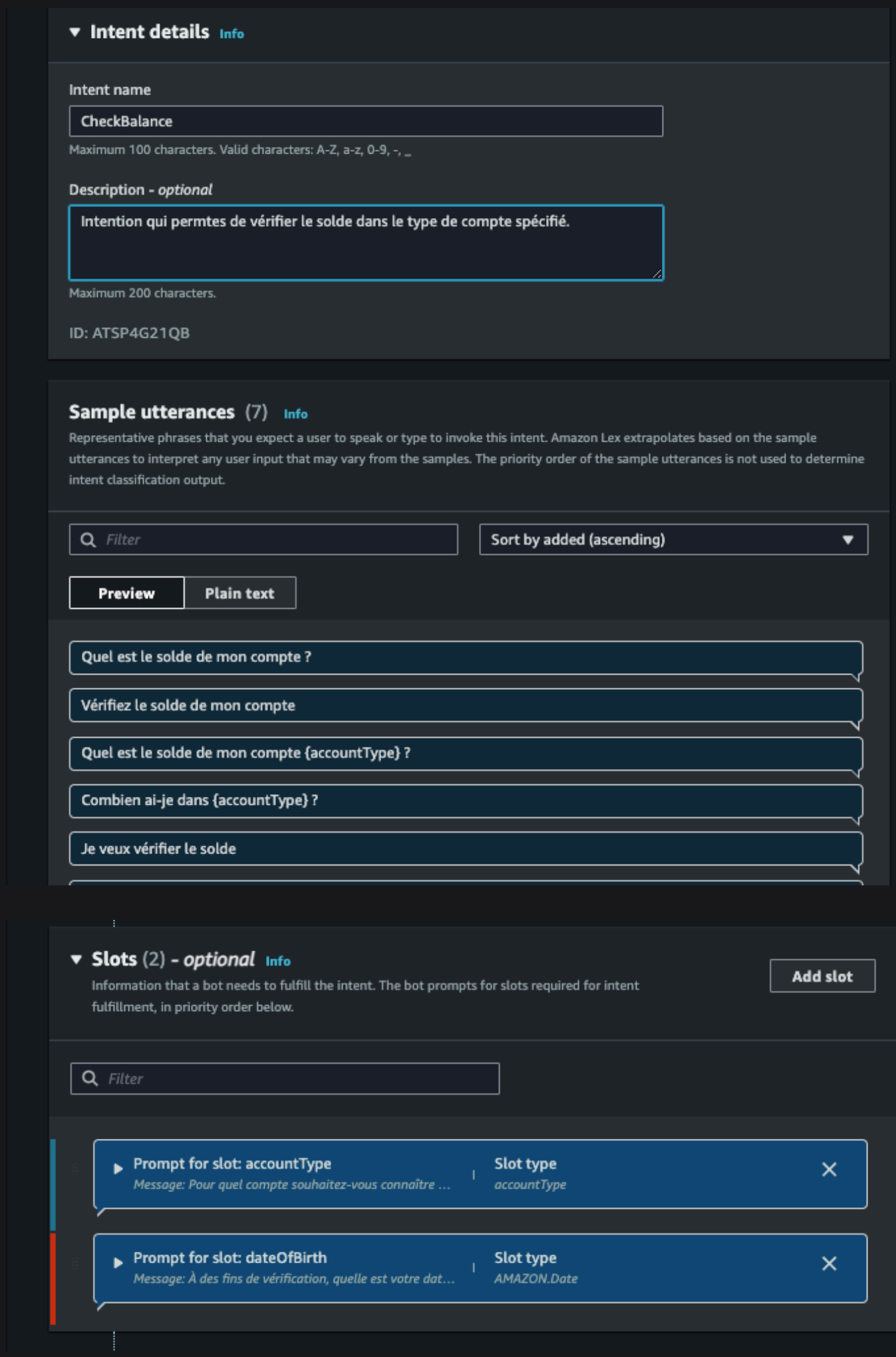
✋ Step #3: Manage FallbackIntent



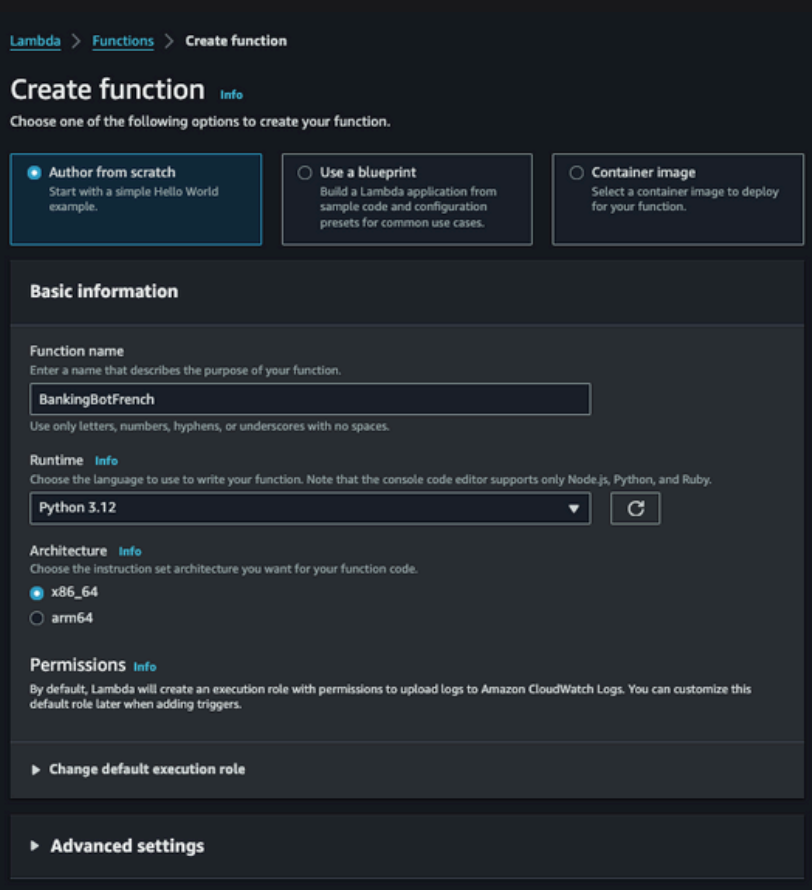
📋 Step #4: Create a custom slot for account types



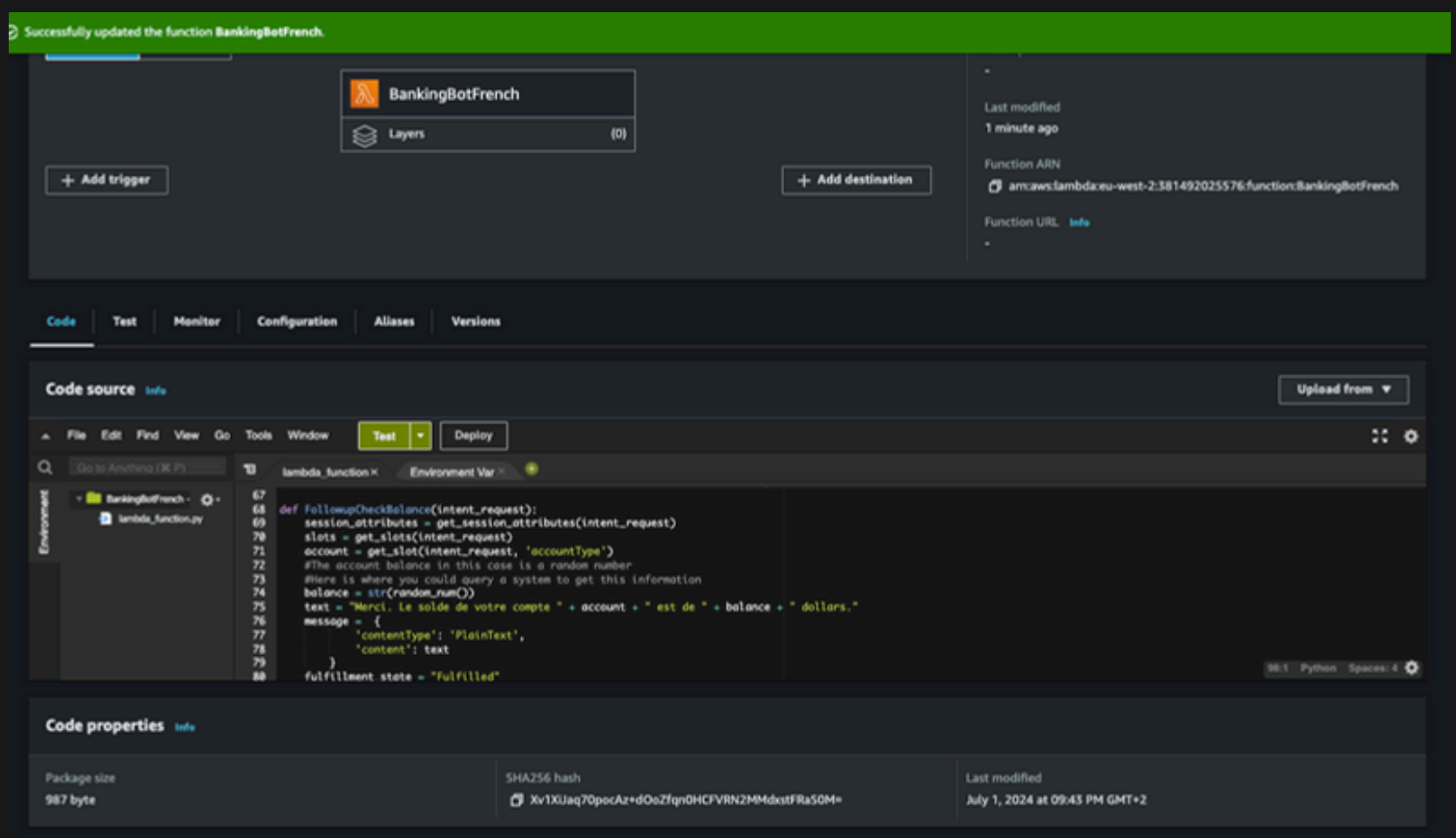
💰 Step #5: Create the CheckBalance intent



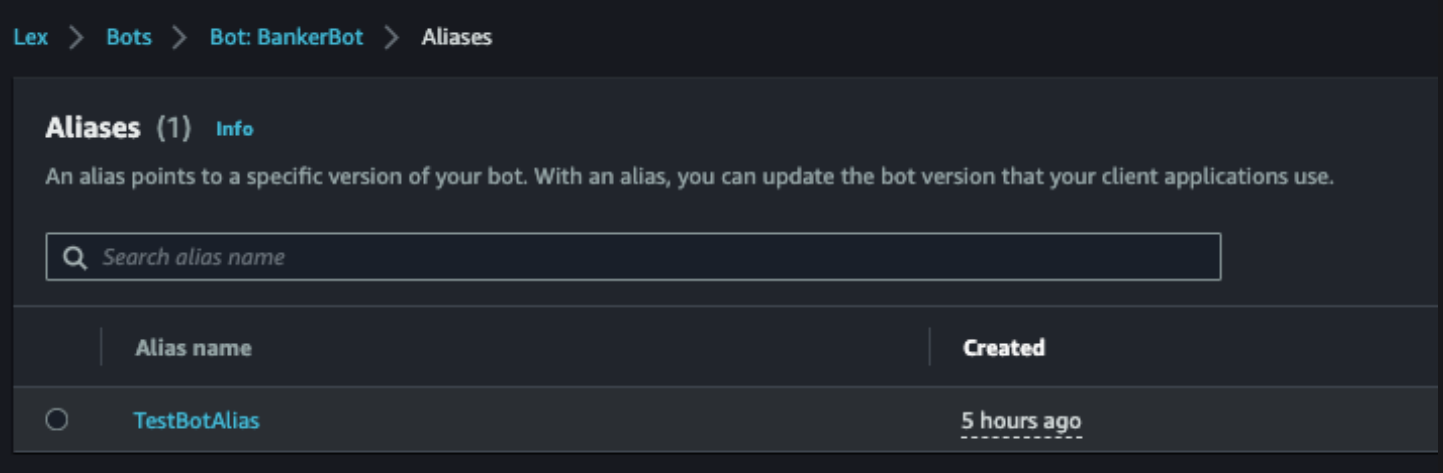
Step #6 : Create Your AWS Lambda function

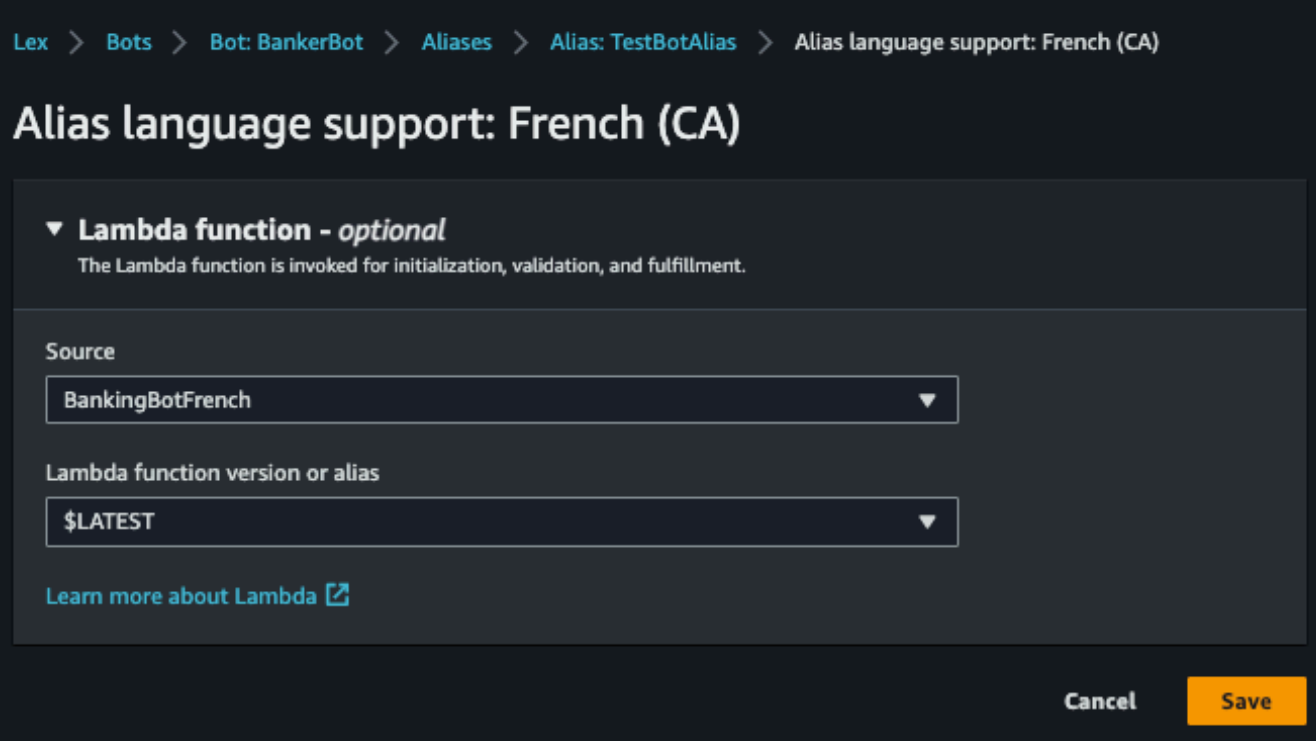


The code is the exact same but I have change some sentences in French



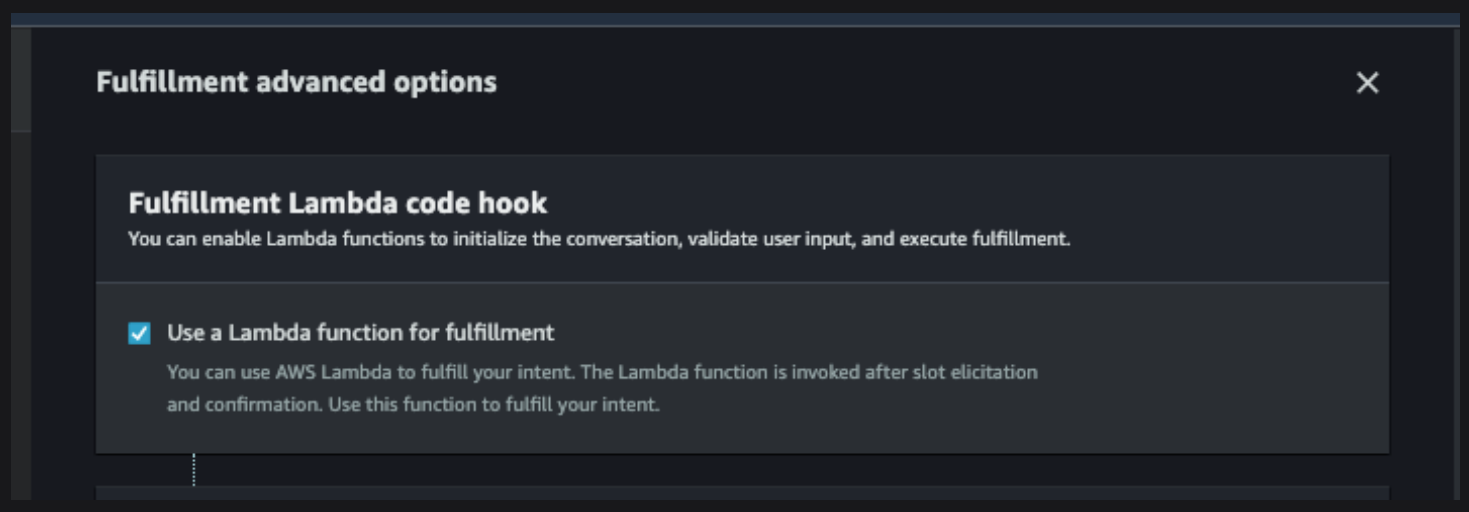
🔌 Step #7: Connect AWS Lambda with Amazon Lex



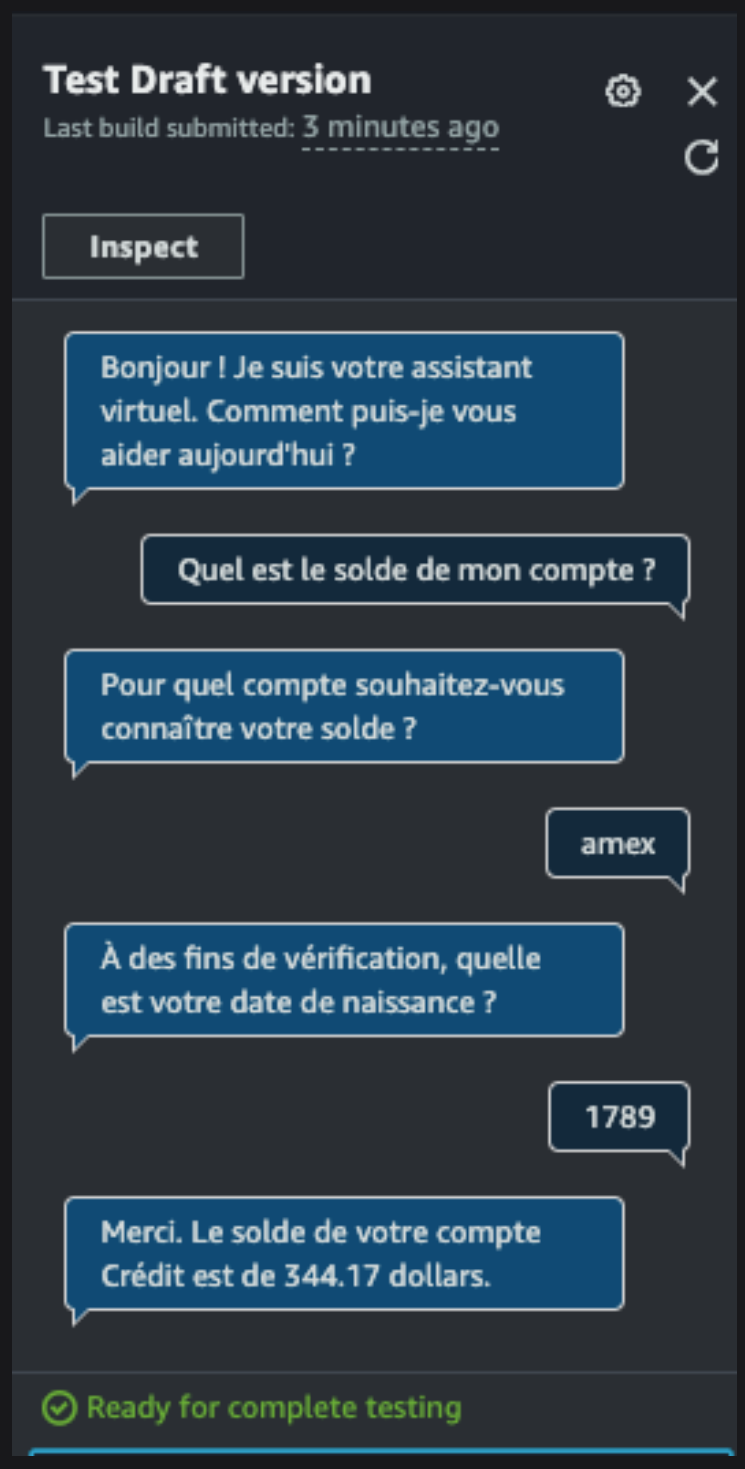


🤝 Step #8: Connect your CheckBalance intent with your Lambda function

Navigate to your **CheckBalance** intent.
Scroll down to your **Fulfilment** panel.



🇫🇷 The French bot is now able to return (random) bank balance figures!



🤝 Step #9 Create the new TransferFunds intent

Intent name

TransferFunds

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

Description - *optional*

Aider les clients à effectuer des transferts entre deux comptes bancaires

Maximum 200 characters.

ID: ZPOSVOFPHJ

Sample utterances (7)

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.

Filter

Sort by added (ascending)

Preview | Plain text

- Puis-je effectuer un virement ?
- Je veux transférer des fonds
- Je veux faire un virement
- J'aimerais transférer {transferAmount} de {sourceAccountType} à {targetAccountType}
- Puis-je transférer {transferAmount} vers mon {targetAccountType}
- Pouvez-vous m'aider à faire un virement ?
- J'ai besoin de faire un virement

Slots (3) - *optional*

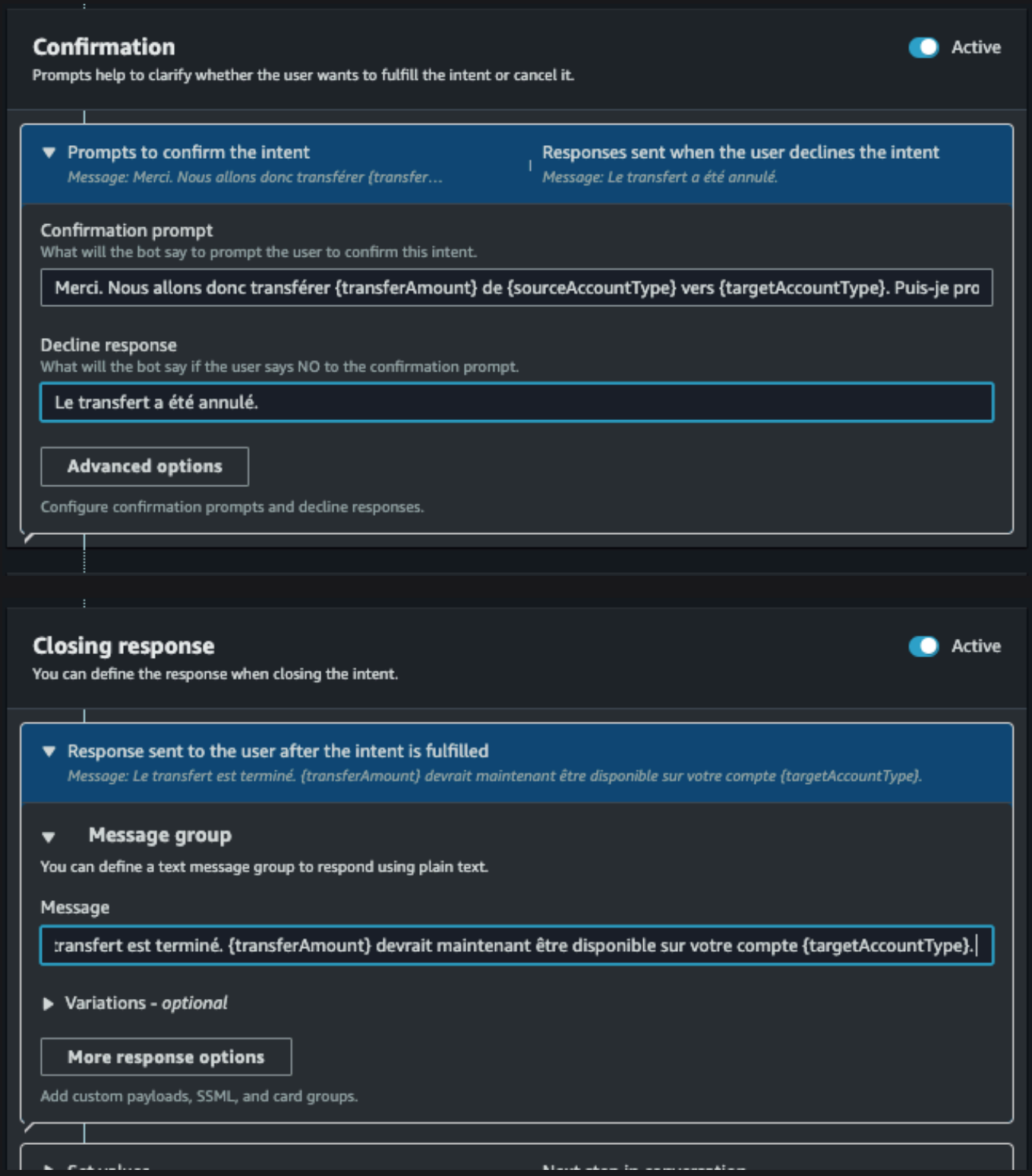
Information that a bot needs to fulfill the intent. The bot prompts for slots required for intent fulfillment, in priority order below.

Add slot

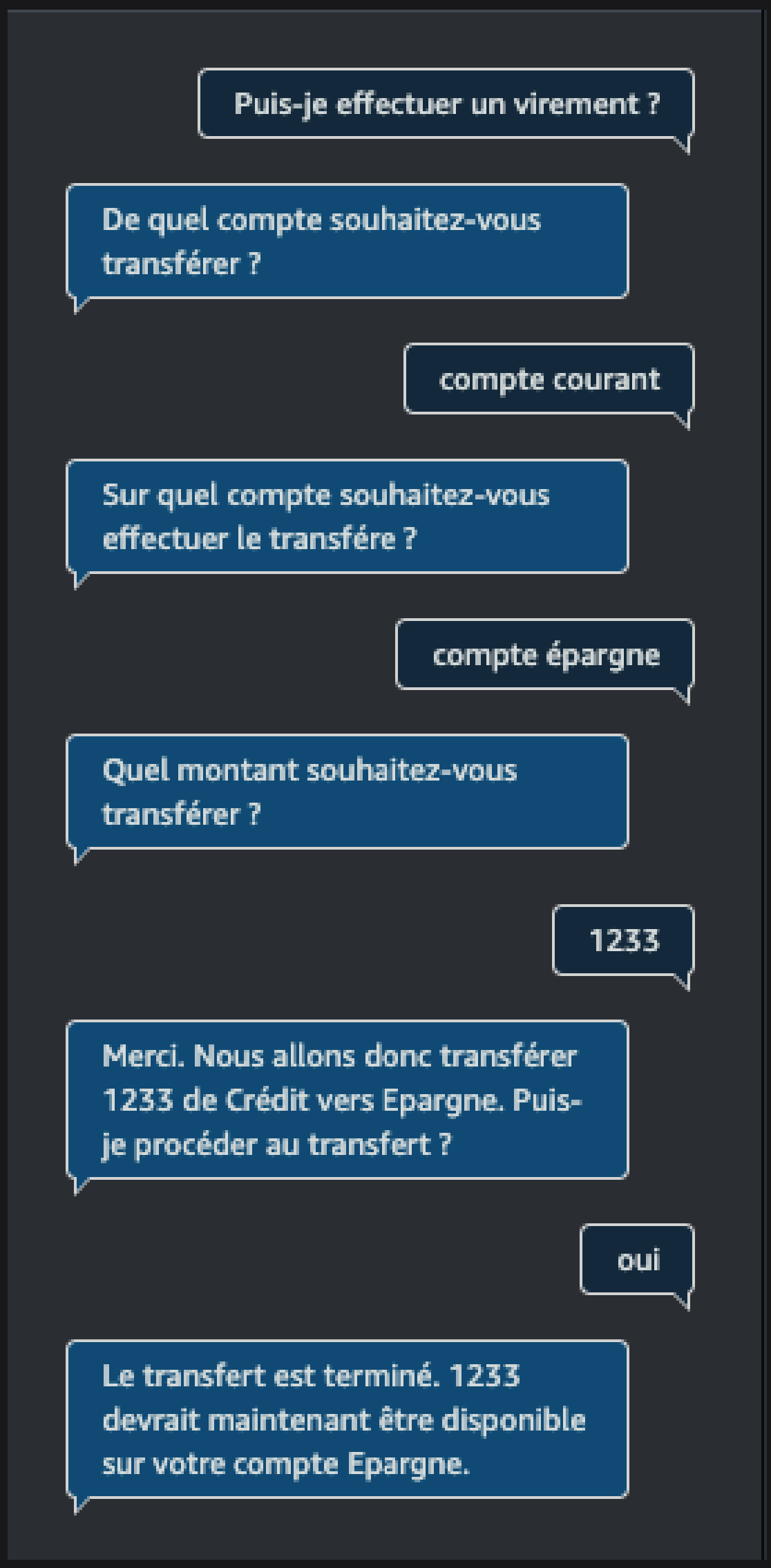
Filter

Prompt for slot: sourceAccountType *Message: De quel compte souhaitez-vous transférer ?*	**Slot type** *accountType*	✕
Prompt for slot: targetAccountType *Message: Sur quel compte souhaitez-vous effectuer le …*	**Slot type** *accountType*	✕
Prompt for slot: transferAmount *Message: Quel montant souhaitez-vous transférer ?*	**Slot type** *AMAZON.Number*	✕

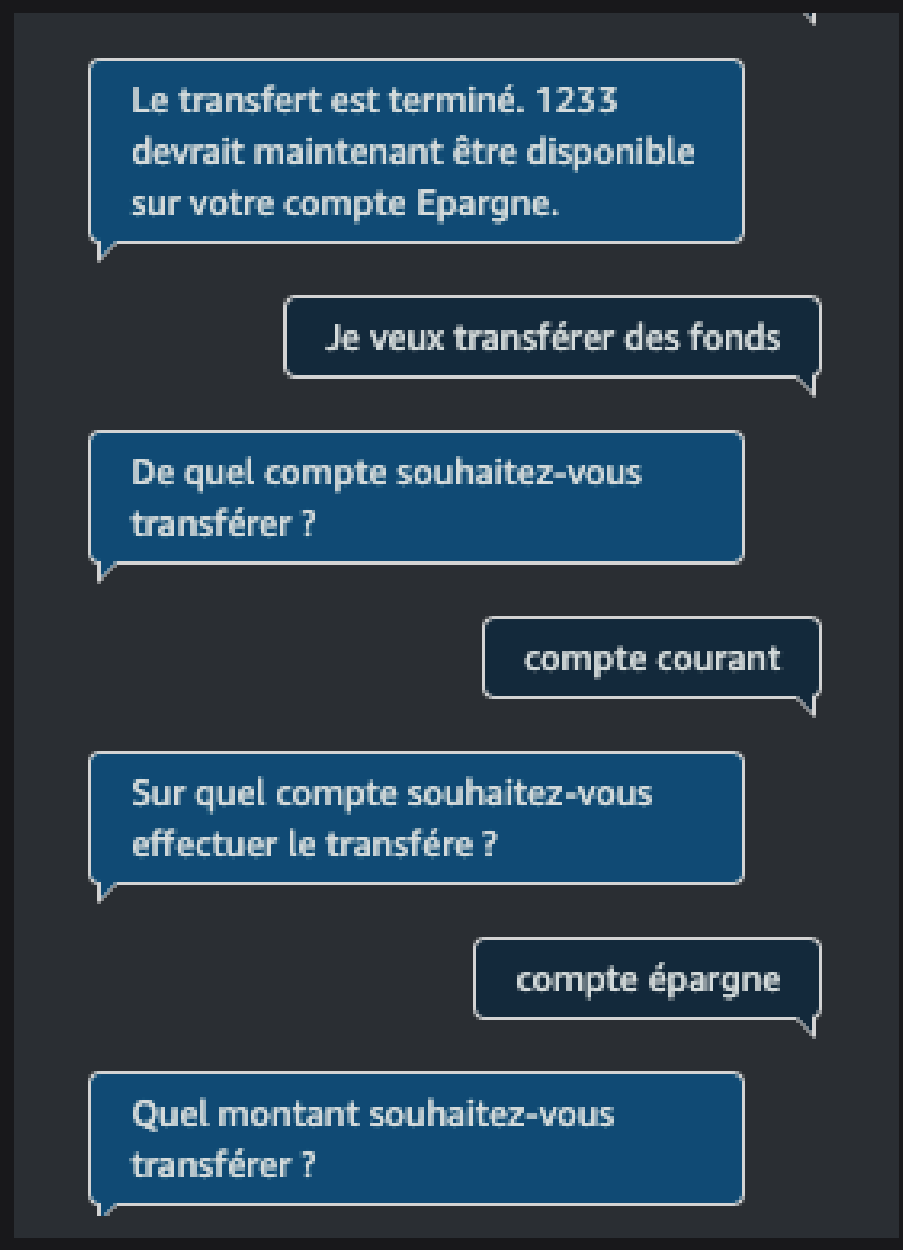
🤝 Step #9 Create the new TransferFunds intent



🤝 Step #9 Create the new TransferFunds intent

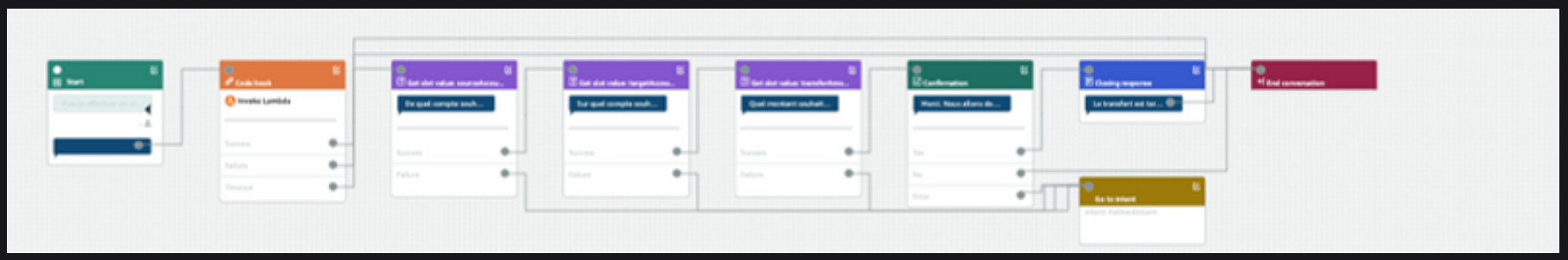


This time, when we ask for a new transfer, the bot won't ask to the user the birthday :



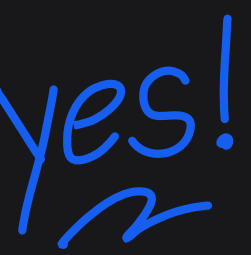
Final thought

Next time, I will try the visual builder



Find this helpful?

Like this post



Leave a comment

Save for later

Let’s connect!

pssst... if you want to get this free project guide and documentation template, **check out NextWork!**