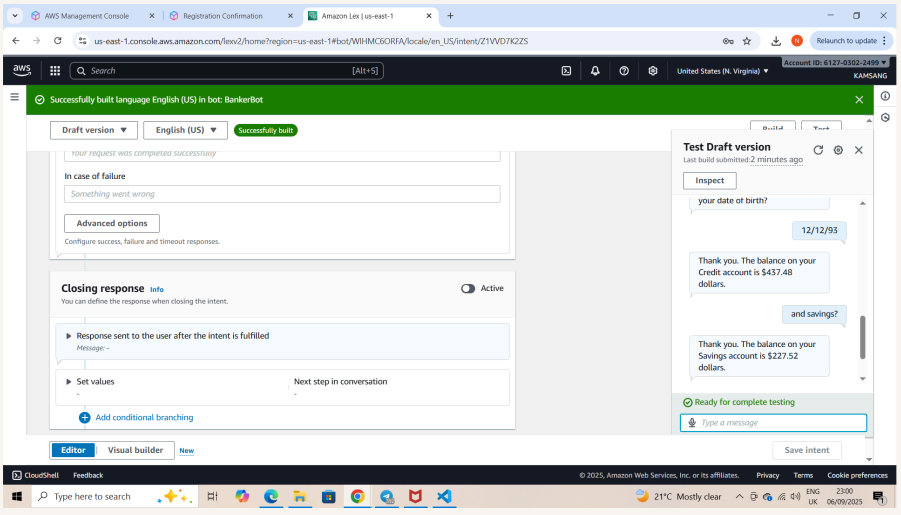
nextwork.org

Save User Info with a Lex Chatbot

Nchindo Boris



Nchindo Boris
NextWork Student

Introducing Today's Project!

What is Amazon Lex?

Amazon Lex is a service for building chatbots and voice assistants using natural language understanding and speech recognition. It's useful because it simplifies bot development, handles scaling, integrates with AWS, and allows seamless interactions with users in real-time.

How I used Amazon Lex in this project

In today's project, I used Amazon Lex to set up context carryover, which is a technique that helps my chatbot remember things it's learnt about the user (like their birthday) from one intent and share them with other intents!

One thing I didn't expect in this project was...

One thing I didn't expect in this project was how important this task will be to make the bot feel natural by maintaining a smooth conversation

This project took me...

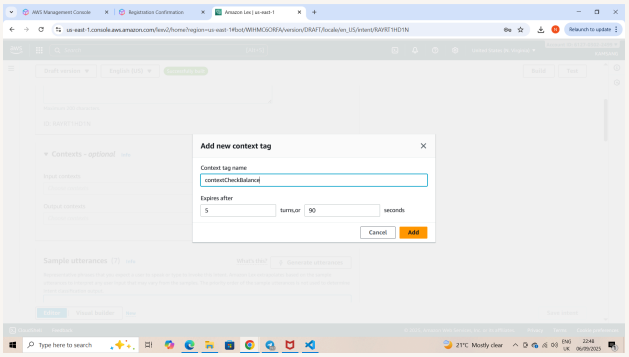
This project took me 40 minutes to complete.

Context Tags

Context tags are short descriptive keywords used to store and check for specific information across different parts of a conversation. They help save the user from having to repeat certain information

There are two types of context tags: 1. Output context tag: This tells the chatbot to remember certain details after an intent is finished, so other parts of the conversation can use this stored information later. For example, the account type from BalanceCheck could be saved and reused 2. Input context tag: This checks if specific details are already available before an intent activates. For example, FollowupCheckBalance will check if this conversation already has the user's date of birth saved somewhere, so it won't need to ask for that information again.

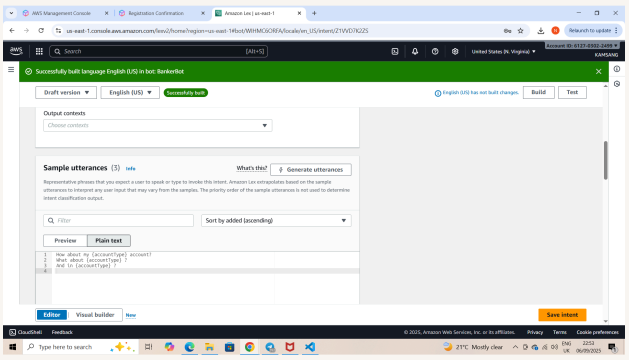
I created a context tag called contextCheckBalance in the CheckBalance intent. This tag helps store the user's date of birth so the chatbot can remember it for future balance checks.


Add new context tag
Context tag name
contextCheckBalance
Expires after
5
turns,or
90
seconds
Cancel
Add

FollowUpCheckBalance

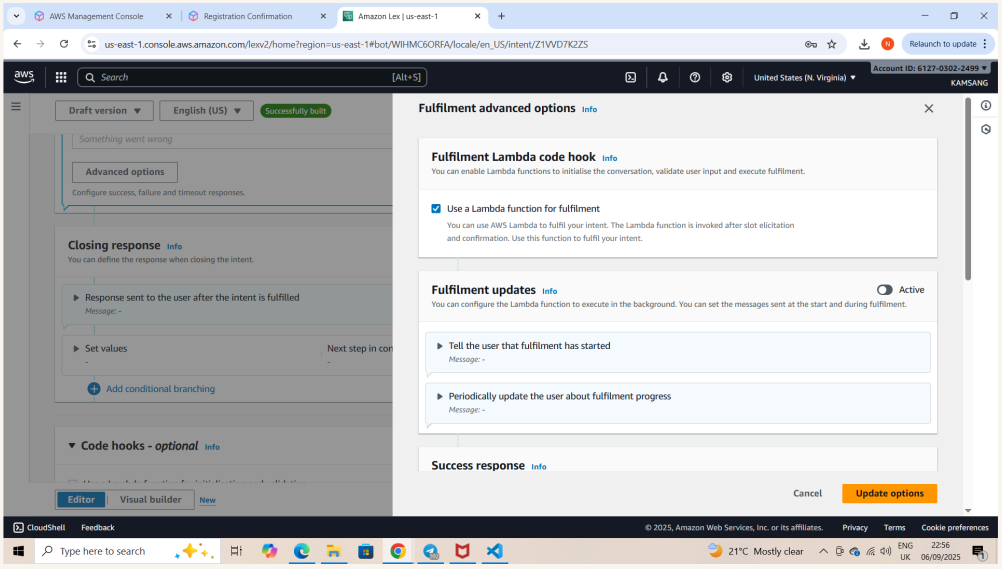
I created a new intent called FollowupCheckBalance. The purpose of this intent is to let the chatbot handle follow-up questions when a user wants to check the balance of another account. It helps keep the conversation going smoothly without asking for the same info again

This intent is connected through context and shared data to the previous intent I made, CheckBalance, because when a user checks their balance using CheckBalance, the bot saves their date of birth. FollowupCheckBalance uses that saved info and the input context to continue the conversation without asking the same questions again. It helps the chatbot feel more natural and efficient.



Input Context Tag

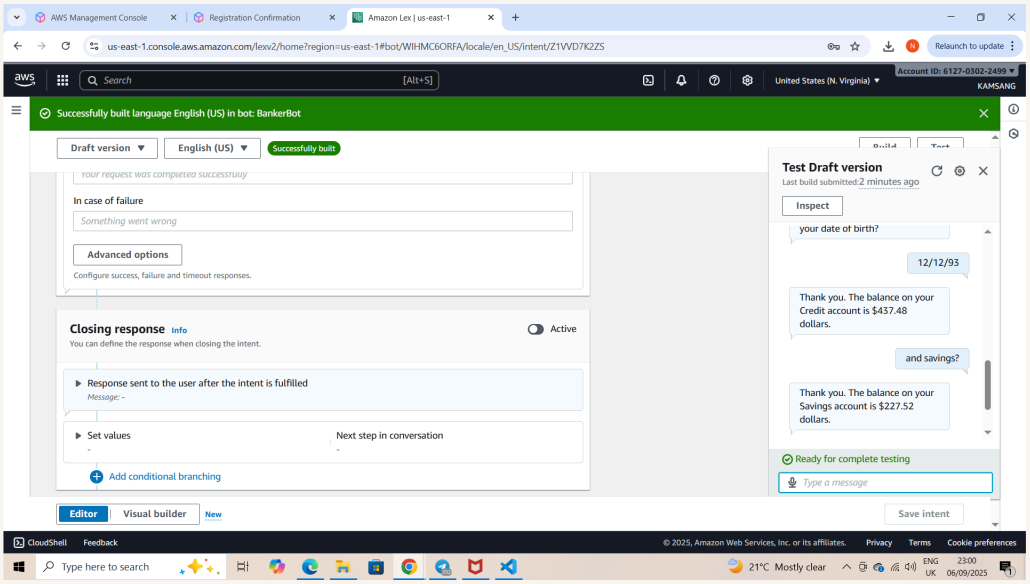
I created contextCheckBalance, that is set as my output context to the CheckBalance intent. This means when the user finishes that step, the chatbot remembers the context and passes it along to the next intent, like FollowupCheckBalance. It's how the bot knows the conversation should keep going without starting over.



The final result!

To see the context tags and followup intent in action, I asked the question “and savings?” These follow-up questions helped trigger the FollowupCheckBalance intent since the context was already set from the previous interaction.

If I had gone straight to trying to trigger FollowUpCheckBalance without setting up any context, the chatbot won’t have the info it needs, like the user’s date of birth. That means it’ll ask the user to enter it again, which kind of defeats the purpose of making the conversation smoother. Context helps the bot pick up right where it left off.

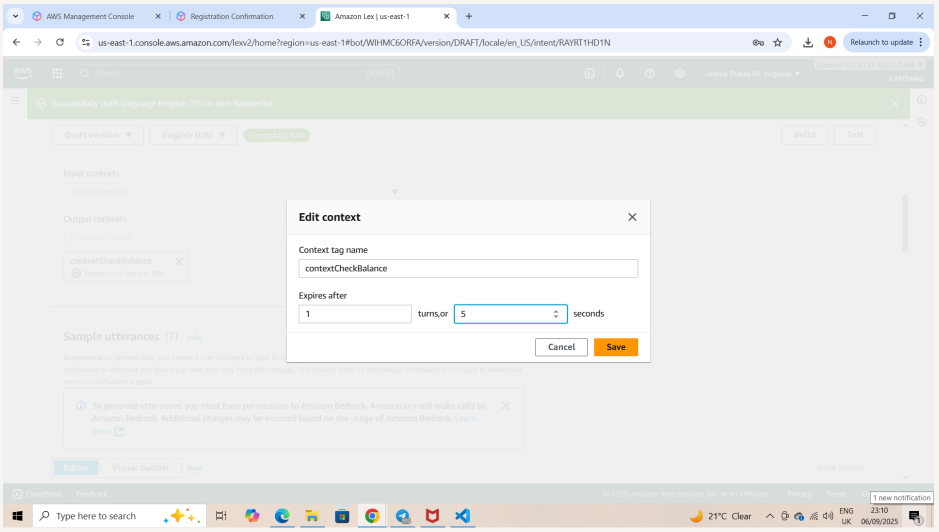


Managing context expiry

An extension for this project is to manage contextCheckBalance's context expiry, which means the chatbot forgets the context after a certain amount of time or after a set number of turns in the conversation. It's like setting an expiration date on what the bot remembers, so it doesn't hold onto user info forever. This helps keep conversations fresh and protects user privacy. By default, expiry is in 5 turns and 90 seconds.

I updated my bot's context expiry to 1 turn and 5 seconds significantly reducing the bot memory. What this means for my end users is that, the bot will forget context from your first balance check in just a shorter time resulting to FallbackIntent again

A long context expiry window would be helpful when you expect the user to have long conversations and run through multiple intents in the same session. Ideally they wouldn't have to share the same information again and again! A shorter context expiry window would be helpful when information is sensitive and you'd want to protect the user from attackers getting access to chat windows with lots of active contexts.


Edit context
Context tag name
contextCheckBalance
Expires after
1
turns,or
5
seconds
Cancel
Save

The place to learn & showcase your skills

Check out nextwork.org for more projects