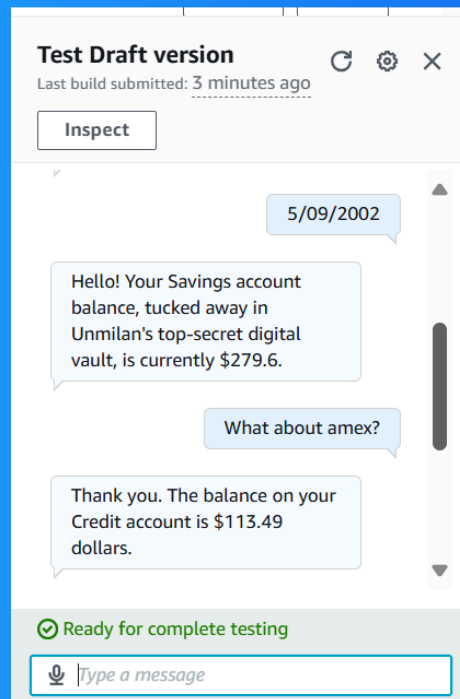


Save User Info with your Chatbot



Unmilan Mukherjee



Introducing Today's Project!

What is Amazon Lex?

Amazon Lex is a chatbot service provided by AWS that allows us to create custom chatbots with intents and different conversation routes. This is useful as this can be very easily connected to other AWS services like Lambda for more functionality.

How I used Amazon Lex in this project

I used Amazon Lex today to set our user's data in some context so that the bot does not keep asking the user for data that was already in our chat's context.

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

I did not expect how we can "link" our different intents using this context option and also set the duration of how long this context is set to tweak our bot based on our needs.

This project took me...

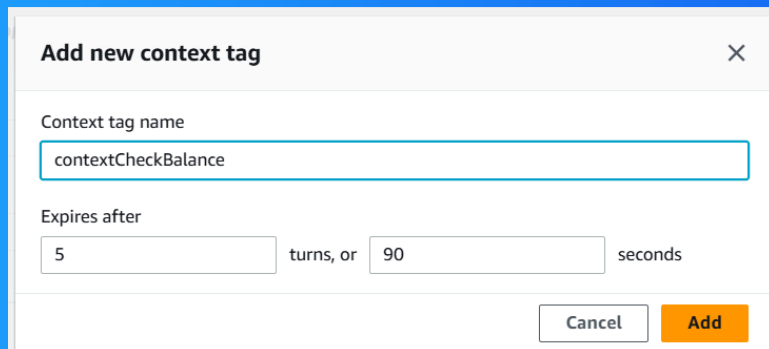
This project took me 60 minutes to do.

Context Tags

Context tags are used to store certain data from our chatbot's conversation temporarily. This prevents the chatbot from asking the user for certain data if it was already mentioned earlier in the conversation.

There are two types of context tags: 1) Output Context Tag which tells the chatbot to remember context after the intent is finished; 2) Input Context Tag which tells the chatbot to check for context before fulfilling the intent.

I created a context tag called: contextCheckBalance. This context tag was created in the intent CheckBalance. This tag stores information about the user's birthday from the chat's context.

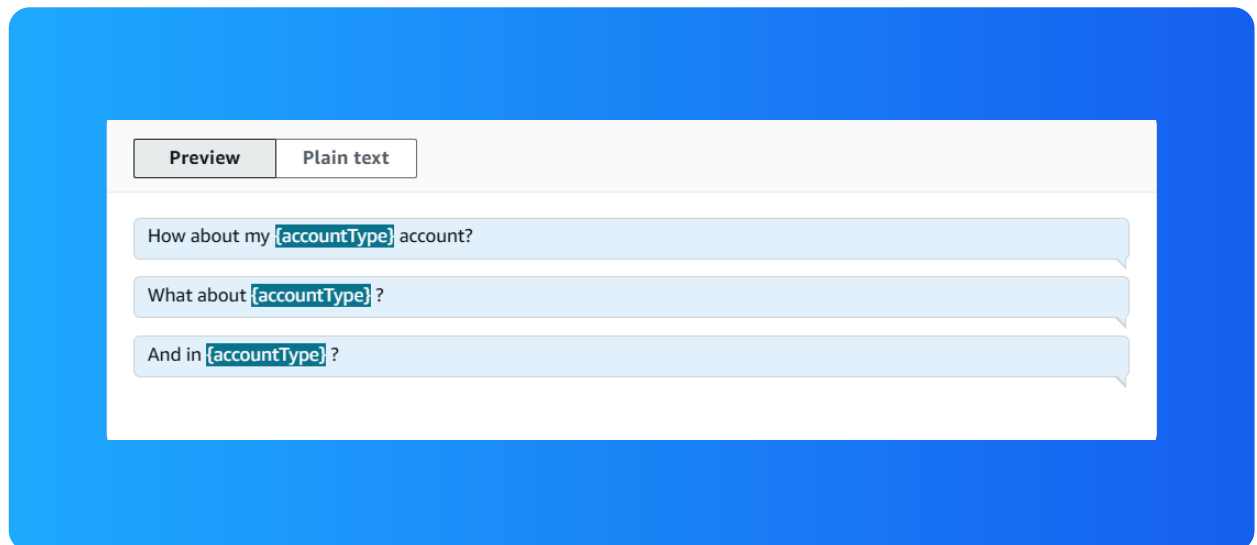


The image shows a dialog box titled "Add new context tag" with a close button (X) in the top right corner. Inside the dialog, there is a text input field labeled "Context tag name" containing the text "contextCheckBalance". Below this, there is a section labeled "Expires after" with two input fields: the first contains the number "5" and is followed by the text "turns, or", and the second contains the number "90" and is followed by the text "seconds". At the bottom right of the dialog, there are two buttons: a "Cancel" button and an "Add" button.

FollowUpCheckBalance

I created a new intent called FollowUpCheckBalance. The purpose of this intent is to handle follow up questions like "What about this account?".

This intent is connected to the previous intent I made, CheckBalance, because we are setting up a context carry over by setting the default value of our dateOfBirth value of our FollowupCheckBalance to the value set in CheckBalance intent.



Input Context Tag

I created an input context, contextCheckBalance that connects the DOB value in our CheckBalance Intent to our FollowupCheckBalance intent. We can do this in the Context section of our intents.

Slot: dateOfBirth Info

Dialog code hook Info ☒ Active
You can enable Lambda functions to validate user input.

► Lambda dialog code hook
Invoke Lambda function: No

▼ Default values - optional

#contextCheckBalance.dateOfBirth

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

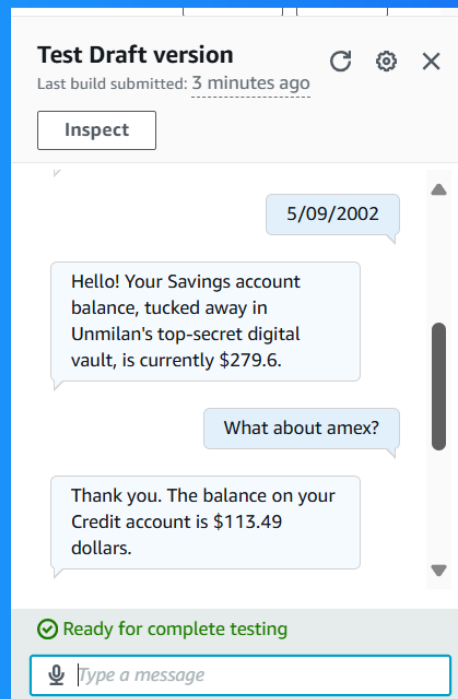
San Diego, #ContextTag.SlotName, [SessionAttributeName] **Add default value**

Cancel **Update slot**

The final result!

To see the context tags and followup intent in action, I first asked it for the balance in my account and provided it my DOB. After that I asked it for a followup by asking "What about amex". This triggered my FollowupCheckBalance intent.

If I had gone straight to trying to trigger FollowUpCheckBalance without setting up any context, then the chatbot would trigger a default error message and tell us to ask again.

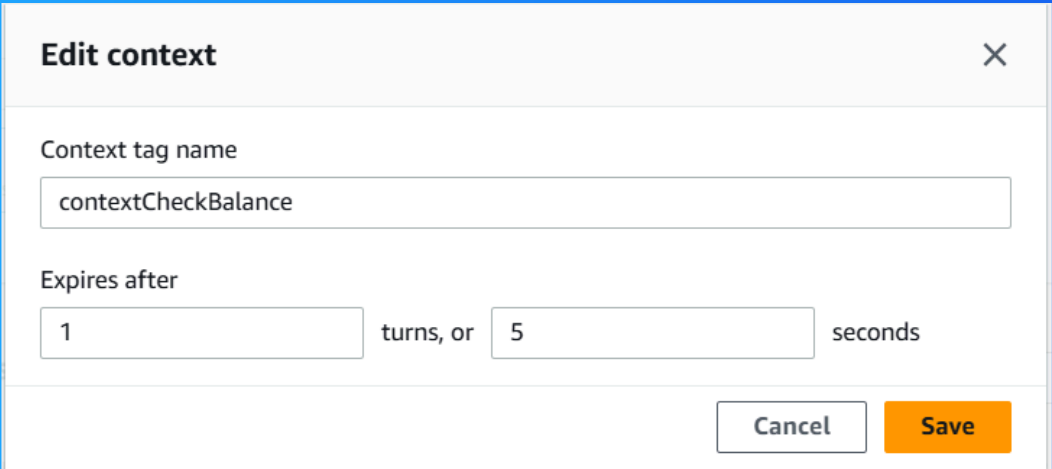


Managing context expiry

An extension for this project is to manage contextCheckBalance's context expiry, which means the deletion of the context in our conversation after a set number of turns or time. By default, expiry is in 5 turns or 90 seconds.

I updated my bot's context expiry to 1 turn or 5 seconds. What this means for my end users is that it is almost like there is no context being saved at all. The bot will forget our user's context after just 1 message or 5 seconds.

A long context expiry context window would be helpful when we are expecting to have a long conversation with our user. A shorter context expiry window would be helpful when the user's data is sensitive and there are data privacy concerns.



The image shows a screenshot of a web application interface with a blue background. In the center, there is a white dialog box titled "Edit context" with a close button (X) in the top right corner. Inside the dialog box, there is a section labeled "Context tag name" with a text input field containing the value "contextCheckBalance". Below this, there is a section labeled "Expires after" with two input fields: the first contains the number "1" and is followed by the text "turns, or", and the second contains the number "5" and is followed by the text "seconds". At the bottom right of the dialog box, there are two buttons: a "Cancel" button and a "Save" button (which is highlighted in orange).