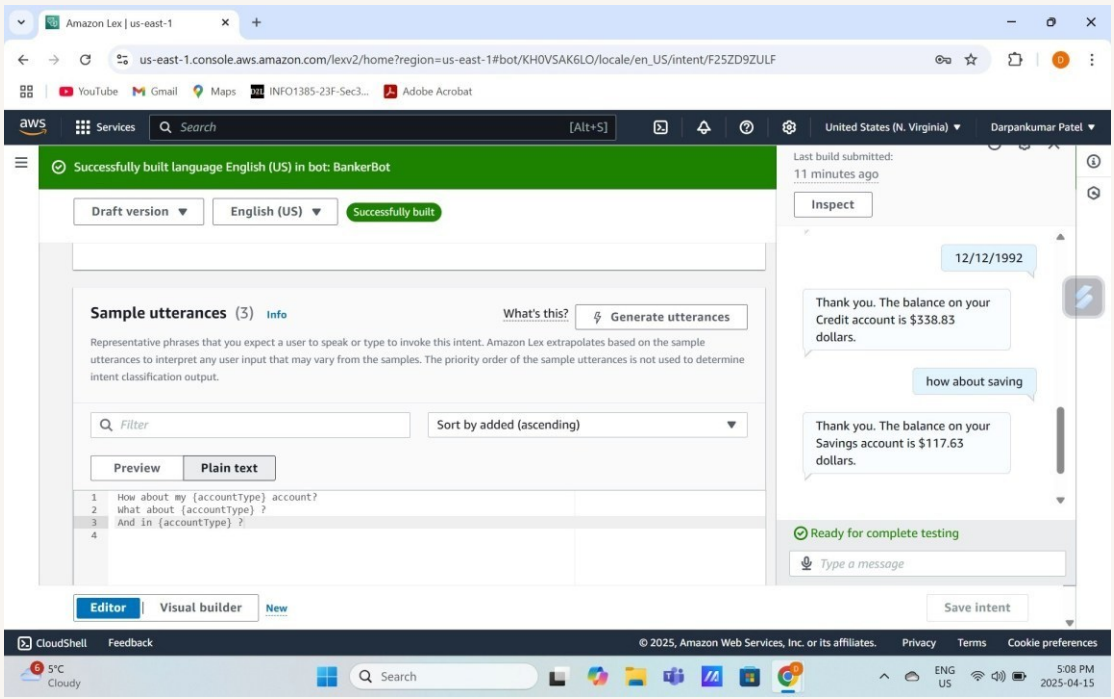
Save User Info with a Lex Chatbot



Introducing Today's Project!

What is Amazon Lex?

Amazon Lex is a service for building chatbots using voice and text. It uses AI to understand language, allowing smart, natural conversations. It's useful since it integrates with AWS, scales well, and doesn’t require managing servers.

How I used Amazon Lex in this project

In today’s project, I used Amazon Lex to build a chatbot that responds to banking queries. I created intents like `CheckBalance` and `FollowupCheckBalance`, added utterances, linked it to Lambda, and used context tags to manage the conversation flow.

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

One thing I didn’t expect was the importance of context tags in handling follow-up questions. I didn’t realize how crucial they were for carrying over information like the user’s date of birth from one intent to another, ensuring a smooth interaction

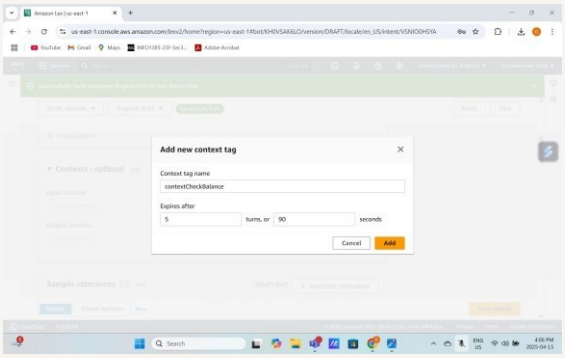
This project took me...

This project took about 2-3 weeks to complete, including setup, configuring intents, creating Lambda functions, testing, and debugging. The learning curve was steep, but it was a valuable experience integrating Amazon Lex with Lambda.

Context Tags

Context tags are markers in Amazon Lex that help chatbots remember and reuse information across a conversation. They reduce the need for users to repeat details by storing data between intents using input and output context tags.
There are two types of context tags: output and input. Output tags let the bot remember info after an intent finish. Input tags check if certain info already exists before starting an intent, helping avoid asking repeat questions.

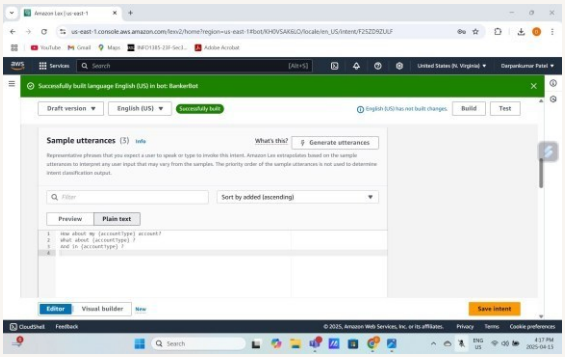
I created a context tag called contextCheckBalance. This context tag was created in the CheckBalance intent. This tag stores information about the user's account type so it can be reused in follow-up intents without asking again.



FollowUpCheckBalance

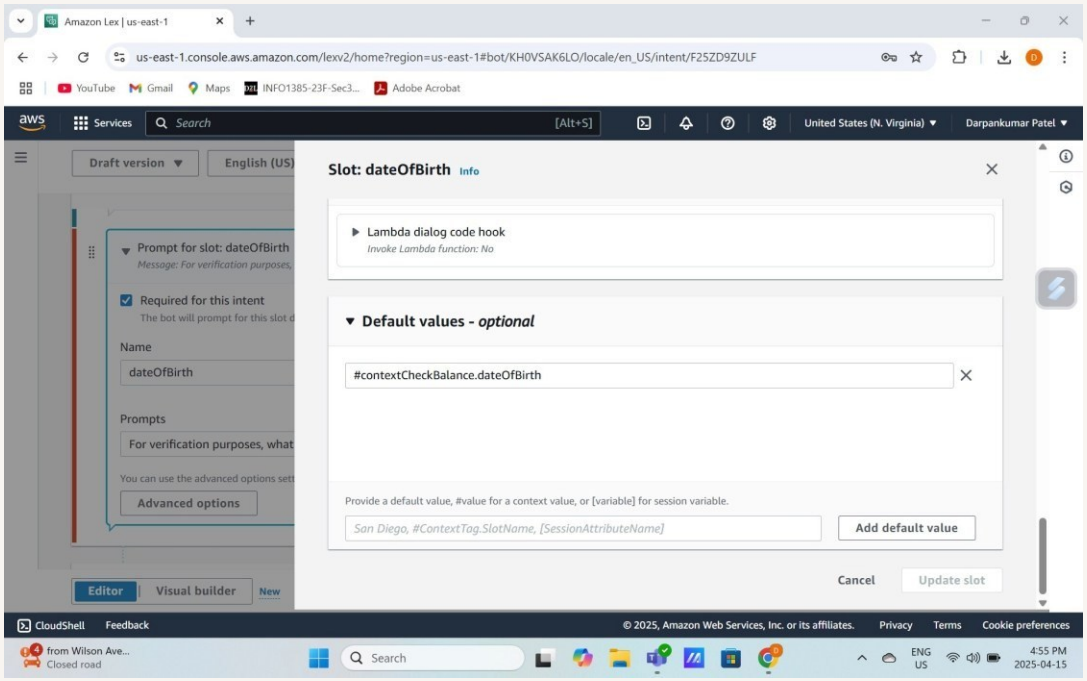
I created a new intent called FollowupCheckBalance to handle follow-up questions about other account balances. It reuses the saved birthday info with context tags, so users don’t have to repeat it.

This intent is connected to the previous intent I made, CheckBalance, because it uses the same user's date of birth stored as an output context. This lets FollowupCheckBalance handle follow-up questions without asking the user for the same info again



Input Context Tag

I created an input context, contextCheckBalance, that connects to the output context from CheckBalance. It allows FollowupCheckBalance to reuse the user's date of birth without asking again, making the conversation smoother and more natural.



The final result!

To see the context tags and followup intent in action, I first asked my chatbot for my account balance to trigger the CheckBalance intent. Then I said, “What about my other account?” to trigger FollowupCheckBalance using the saved date of birth.

If I had tried to trigger FollowupCheckBalance without setting any context, the chatbot wouldn’t know my date of birth and would throw an error or ask for it— since the input context from CheckBalance wasn’t set yet.