



Cloud Computing Project

“Amazon Lex Chatbot for Personal Banking”

Under the guidance of:

Dr. Aditi Sharma

Team Members:

Pratham Dave - 21070122121

Sangeet Agrawal - 21070122140

Pranay Bisht - 21070122117

Samyak Mishra - 21070122139

Department of CSE - Div B (2021-25)

Abstract

This report outlines the process of building a customer service chatbot for a fictitious FinTech company using Amazon Lex and AWS Lambda. The objective is to develop an automated solution that can handle common customer inquiries, such as checking account balances and retrieving transaction details. The project utilizes Amazon Web Services (AWS) tools, particularly Amazon Lex for creating conversational interfaces and AWS Lambda for backend logic. By defining specific intents and slot types in Amazon Lex and integrating AWS Lambda to handle the backend processing, a seamless interaction workflow is created. This report breaks down the steps to build the chatbot and connect it with backend logic to provide a comprehensive customer service experience.

Methodology

The methodology for developing this chatbot involves two primary sections: creating the conversational interface using Amazon Lex and implementing the backend processing with AWS Lambda.

1. Building the Amazon Lex Chatbot

a. **Creating the Lex Bot:**

Start by accessing the AWS Management Console and navigating to the Amazon Lex service. Set up a new bot named “PersonalBanker,” ensuring to configure session timeouts, output voice settings, and other necessary parameters.

b. **Defining Intents:**

An intent defines an action the user wants to perform. Create an intent called “GetBalanceCheck” to allow users to check their bank balances. Add sample utterances that represent how users might phrase their requests, such as “Check my bank balance,” “How much money is in my account,” and “How much money do I have.”

c. **Creating Slot Types:**

Slots are used to capture additional information from the user that is required to fulfill the intent. Define slots such as “AccountType” to capture the type of account the user is asking about (e.g., Savings or Current) and “PinNumber” to capture the user’s four-digit PIN for authentication.

d. Configuring Slot Prompts:

Set up prompts for each slot to request information from users. For example, for the “AccountType” slot, the prompt could be, “What type of account do you want to check (Current or Savings)?” Similarly, for the “PinNumber” slot, the prompt might be, “What is your PIN number for your {AccountType} account?”

e. Testing and Enhancing the Bot:

Use the testing interface provided in the Amazon Lex console to check the bot’s performance. Make sure it recognizes user intents correctly, processes slot information accurately, and handles different scenarios. Add response cards to simplify user choices for specific questions and incorporate error handling to guide users if they provide incorrect inputs.

2. Implementing AWS Lambda Functions for Backend Logic

a. Creating the Lambda Function:

Click on the AWS Lambda service present in the AWS Console. Create a new Lambda function named “myPersonalBanker” using the Node.js runtime. This function will handle the backend processing and provide responses to user queries from Amazon Lex.

b. Writing the Backend Logic:

Develop the backend logic in JavaScript within the Lambda function. This code identifies the intent received from Lex (such as ‘GetBalanceCheck’), retrieves the slot values (e.g., “AccountType” and “PinNumber”), and generates a response. For simplicity, a static array is used to simulate data retrieval, but in a real-world application, the function would access a database.

c. Adding Error Handling and User Feedback:

To make the chatbot more user-friendly, add logic to the Lambda function to handle incorrect inputs, such as an invalid PIN. Modify the function to reset the slot and prompt the user again until a correct PIN is provided, ensuring a smooth user experience.

d. Integrating Lambda with Amazon Lex:

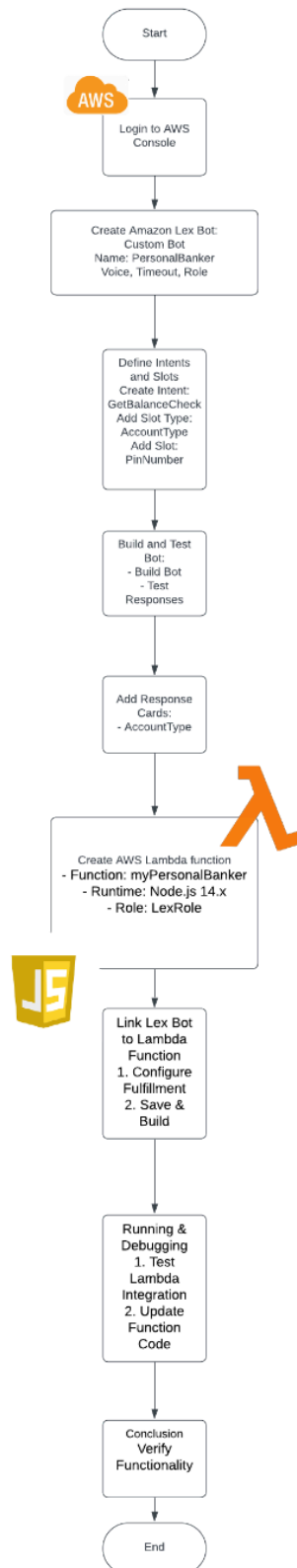
Integrate the Lambda function with the Amazon Lex bot by configuring the “Fulfillment” settings in Lex to use the Lambda function. This allows Lex to

invoke the Lambda function whenever it needs to fulfill an intent based on user input, enabling dynamic and context-aware responses.

e. Testing the Integrated System:

After linking the Lambda function to Amazon Lex, perform additional tests to validate the integrated system. Ensure that the chatbot invokes the Lambda function correctly, processes the information accurately, and returns relevant responses to users.

Flow Diagram



Screenshots of Workflow

Phase 1: Utilization of Amazon Lex Service

- Creation of Lex Chatbot:

Create your bot

Amazon Lex enables any developer to build conversational chatbots quickly and easily. With Amazon Lex, no deep learning expertise is necessary—you just specify the basic conversational flow directly from the console, and then Amazon Lex manages the dialogue and dynamically adjusts the response. To get started, you can choose one of the sample bots provided below or build a new custom bot from scratch.

CREATE YOUR OWN

TRY A SAMPLE

Custom bot

BookTrip

OrderFlowers

ScheduleAppointment

Bot name

Language

Output voice

Session timeout min

Sentiment analysis ☐ Yes ☒ No

<https://us-east-1.console.aws.amazon.com/console/home?region=us-east-1> © 2024, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Output voice

Session timeout min

Sentiment analysis ☐ Yes ☒ No

IAM role [AWSServiceRoleForLexBots](#)
Automatically created on your behalf

COPPA Please indicate if your use of this bot is subject to the [Children's Online Privacy Protection Act \(COPPA\)](#). [Learn more](#)
☐ Yes ☒ No

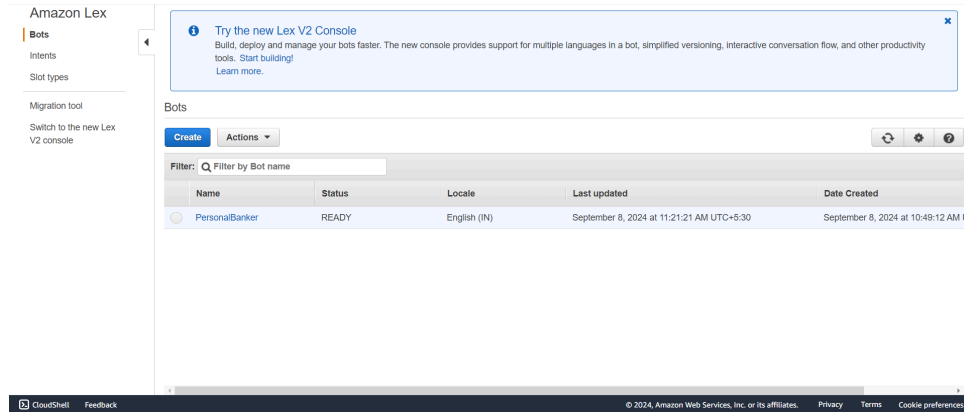
Confidence score threshold

Tags

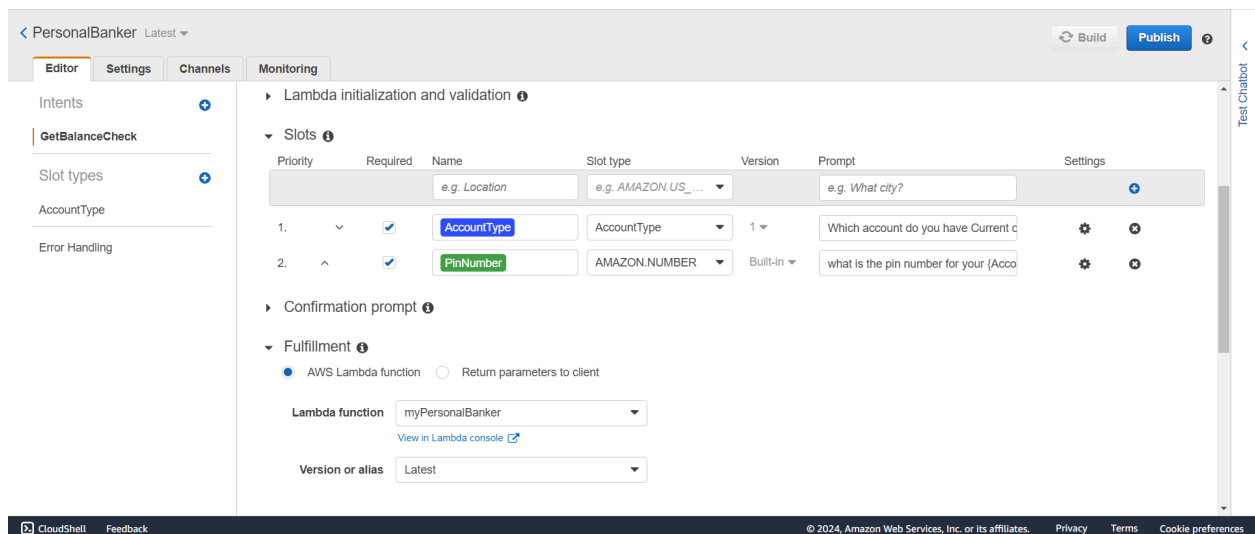
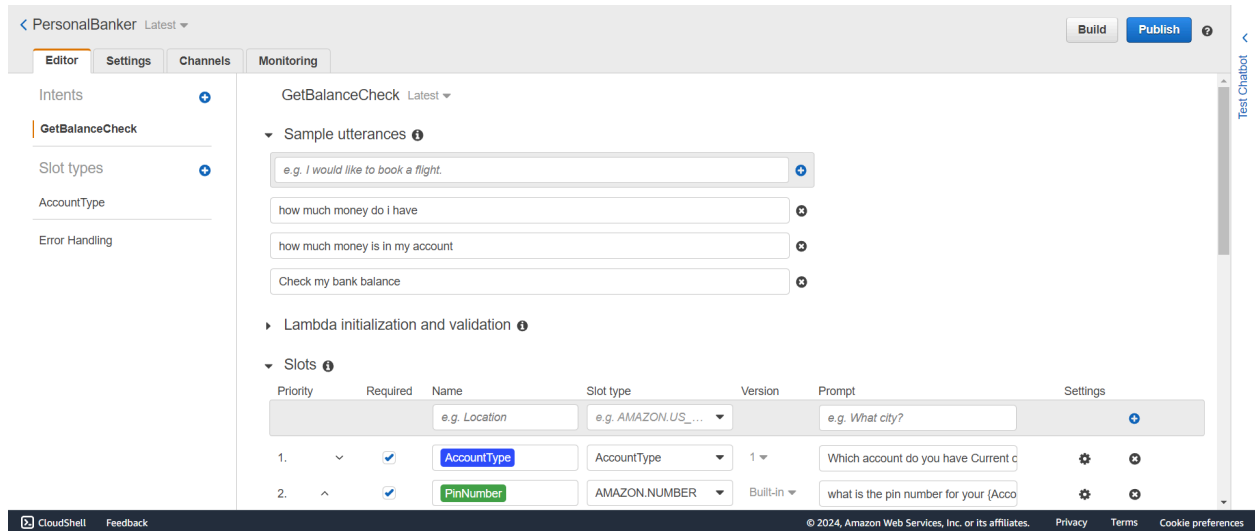
[Cancel](#) [Create](#)

<https://us-east-1.console.aws.amazon.com/console/home?region=us-east-1> © 2024, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

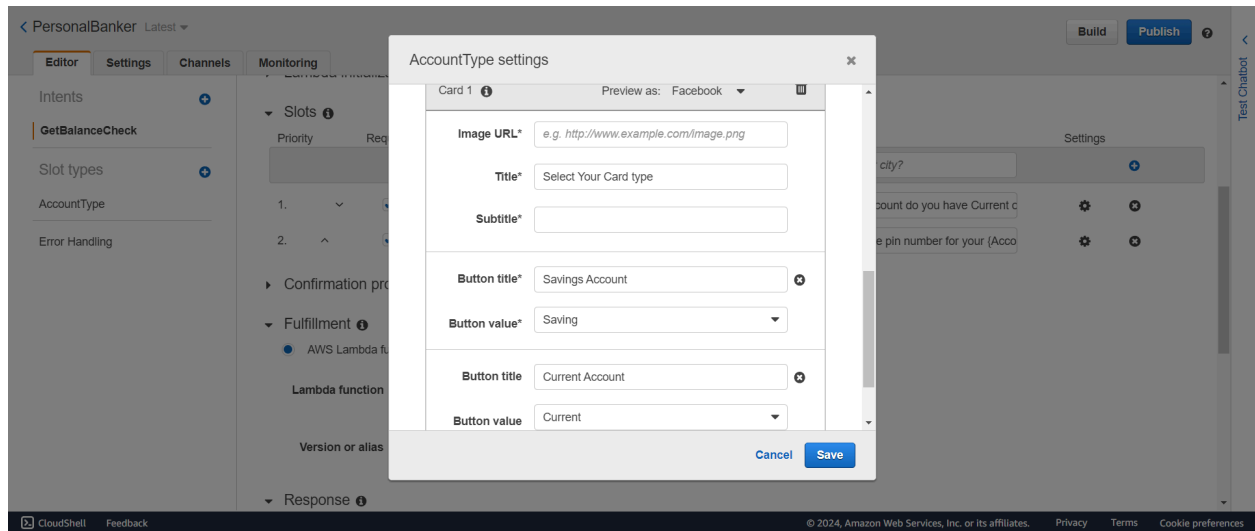
- Configuring the Lex Chatbot on V1 Console:



- Editing the Sample Utterances & Slot Types in Lex Chatbot:



- Adding the Buttons in Slot Intents:



Phase 2: Utilization of Lambda Function Service by AWS

- Creation of myPersonalBanker Lambda Function:

Lambda

Dashboard

Applications

Functions

▼ Additional resources

Code signing configurations

Event source mappings

Layers

Replicas

▼ Related AWS resources

Step Functions state machines

Lambda > Functions > Create function

Create function

Choose one of the following options to create your function.

☒ Author from scratch

☐ Use a blueprint

☐ Container image

Basic information

Function name

myPersonalbanker

Runtime

Node.js 20.x

Architecture

x86_64

Lambda

Dashboard

Applications

Functions

▼ Additional resources

Code signing configurations

Event source mappings

Layers

Replicas

▼ Related AWS resources

Step Functions state machines

Permissions

By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ Change default execution role

Execution role

☒ Create a new role with basic Lambda permissions

☐ Use an existing role

☐ Create a new role from AWS policy templates

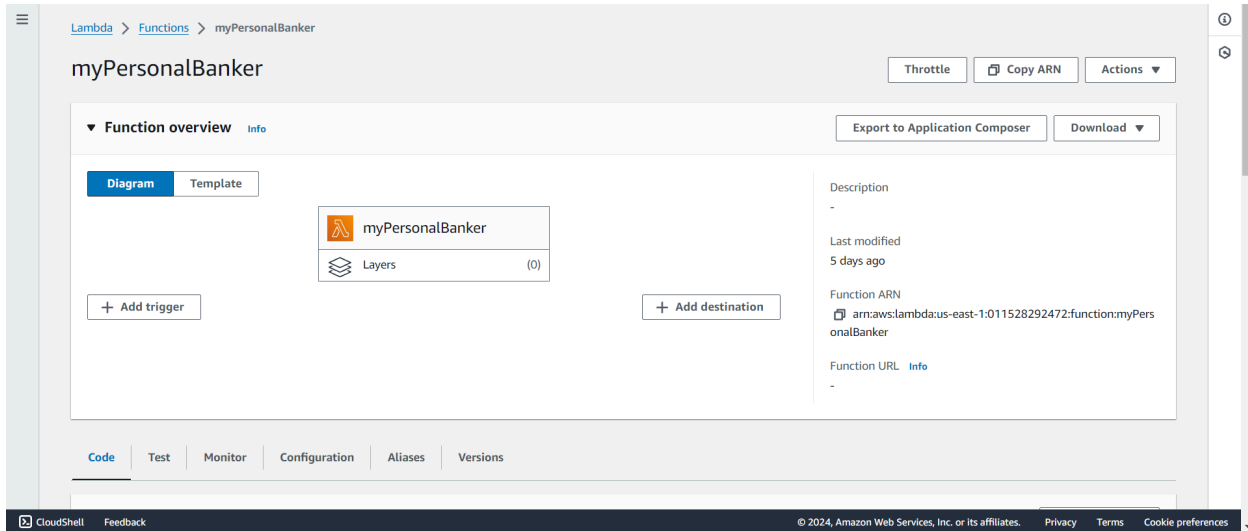
Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.

Lambda will create an execution role named myPersonalbanker-role-f7ce9ifh, with permission to upload logs to Amazon CloudWatch Logs.

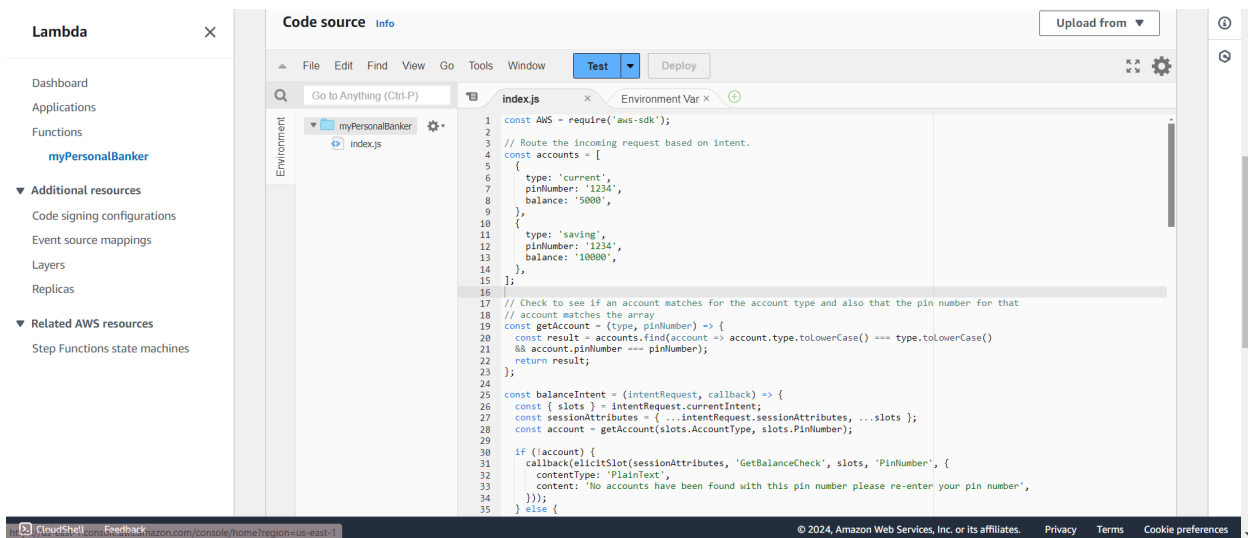
► Advanced settings

Cancel

Create function



- Adding the JavaScript Code for fetching the Account details (Type, Pin & Balance). This helps in linking the “Lambda Function Service” with “Amazon Lex Service”.



- Selection of Runtime Settings (Runtime, Handler, Architecture)

Lambda

Dashboard

Applications

Functions

myPersonalBanker

▼ Additional resources

Code signing configurations

Event source mappings

Layers

Replicas

▼ Related AWS resources

Step Functions state machines

62 63 };

16:1 JavaScript Spaces: 2

Code properties Info

Package size

1.4 kB

SHA256 hash

vdGokH4+oinBCagHV+vMt1R1qAESdwzhNhA7JFVjWU=

Last modified

5 days ago

Runtime settings Info

Runtime

Node.js 16.x

Handler Info

index.handler

Architecture Info

x86_64

► Runtime management configuration

Layers Info

Edit

Add a layer

Merge order

Name

Layer version

Compatible runtimes

Compatible architectures

Version ARN

There is no data to display.

CloudShell Feedback

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Phase 3: Building the Intents & Testing the Chatbot

PersonalBanker Latest

Build Publish

Editor Settings Channels Monitoring

Intents

GetBalanceCheck

Slot types

AccountType

Error Handling

GetBalanceCheck Latest

Sample utterances

Slot types

Lambda initialization and validation

Slots

Test bot (Latest) Ready. Build complete.

Check my bank balance

Which account do you have Current or saving?

Select Your Card type

Savings Account

Current Account

what is the pin number for your Saving?

Clear chat history

Chat with your bot...

Inspect response

Dialog State: ElicitSlot

Show

CloudShell Feedback

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

PersonalBanker Latest

Build Publish

Editor Settings Channels Monitoring

Intents

GetBalanceCheck

Slot types

AccountType

Error Handling

GetBalanceCheck Latest

Sample utterances

Slot types

Lambda initialization and validation

Slots

Test bot (Latest) Ready. Build complete.

Current Account

what is the pin number for your Saving?

3456

No accounts have been found with this pin number please re-enter your pin number

1234

You have ₹10000 in your account

Clear chat history

Chat with your bot...

Inspect response

Dialog State: Fulfilled

Show

CloudShell Feedback

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Future Scope

Amazon Connect Integration: Develop a virtual call center by connecting the chatbot with Amazon Connect, incorporating text-to-voice and multilingual support for enhanced customer interactions.

Improved Call Management: Utilize Amazon S3 for efficient storage and retention of customer data, and leverage Amazon Connect's analytics tools to monitor and improve call efficiency.

Advanced Voice and Chat Capabilities: Enhance text-to-speech functionalities for natural voice interactions and expand support to include chat for better customer engagement.

Natural Language Understanding: Integrate Amazon Lex to improve context-aware responses, enabling more intuitive and seamless customer service experiences.

Conclusion

This project demonstrates the process of creating a chatbot using Amazon Lex and integrating it with AWS Lambda for backend logic. The chatbot can handle typical customer service tasks, like checking account balances, by leveraging natural language processing capabilities and serverless computing. By combining Amazon Lex's powerful conversational interface with the dynamic logic of AWS Lambda, the solution provides a scalable, efficient, and user-friendly customer service experience. Proper cleanup of all AWS resources created during the exercise is necessary to avoid incurring additional costs.