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.

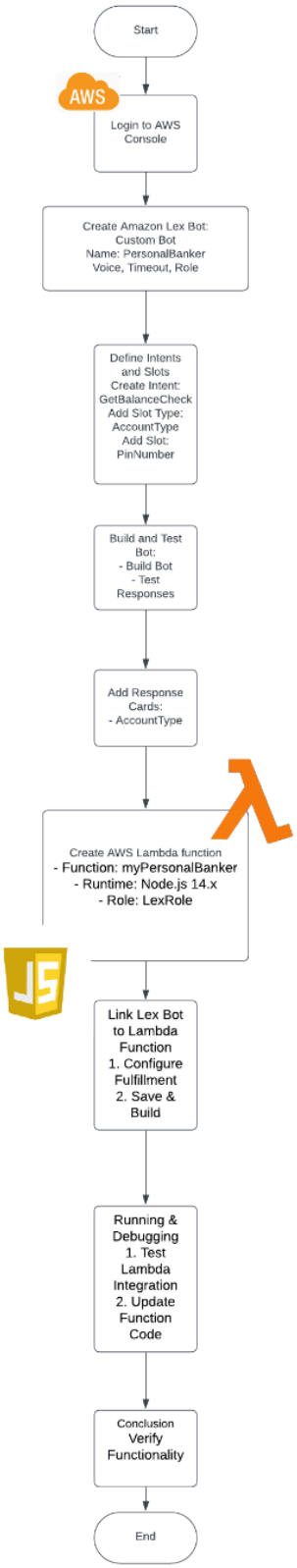
c. **Adding Error Handling and User Feedback:**
   To make the chatbot more user-friendly, add logic to the Lambda function to handle incorrect inputs, such as an invalid PIN. Modify the function to reset the slot and prompt the user again until a correct PIN is provided, ensuring a smooth user experience.

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.

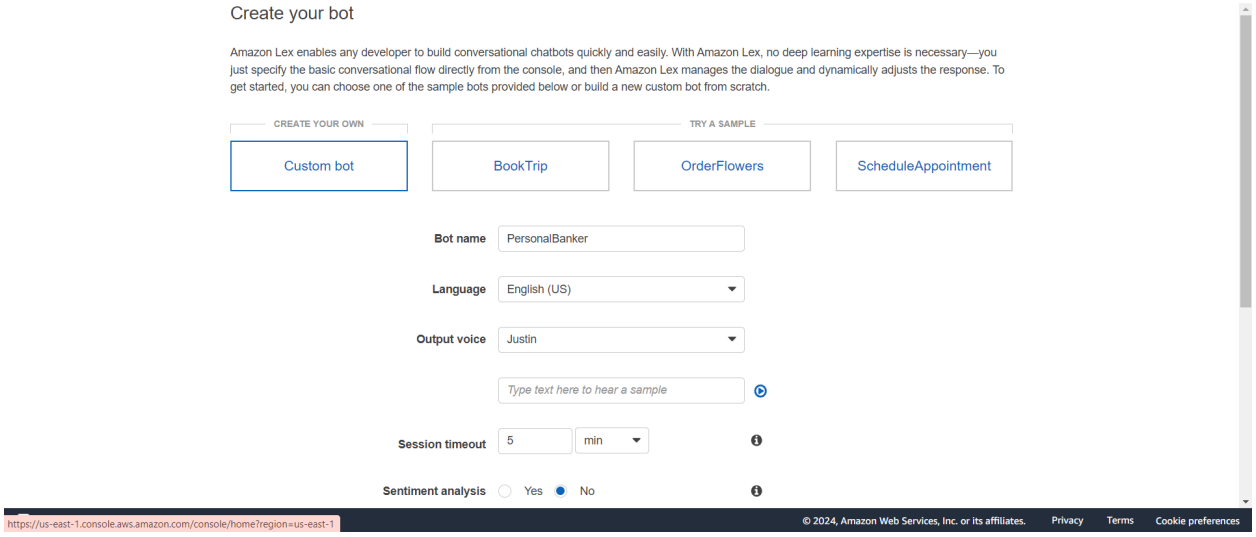
Flow Diagram

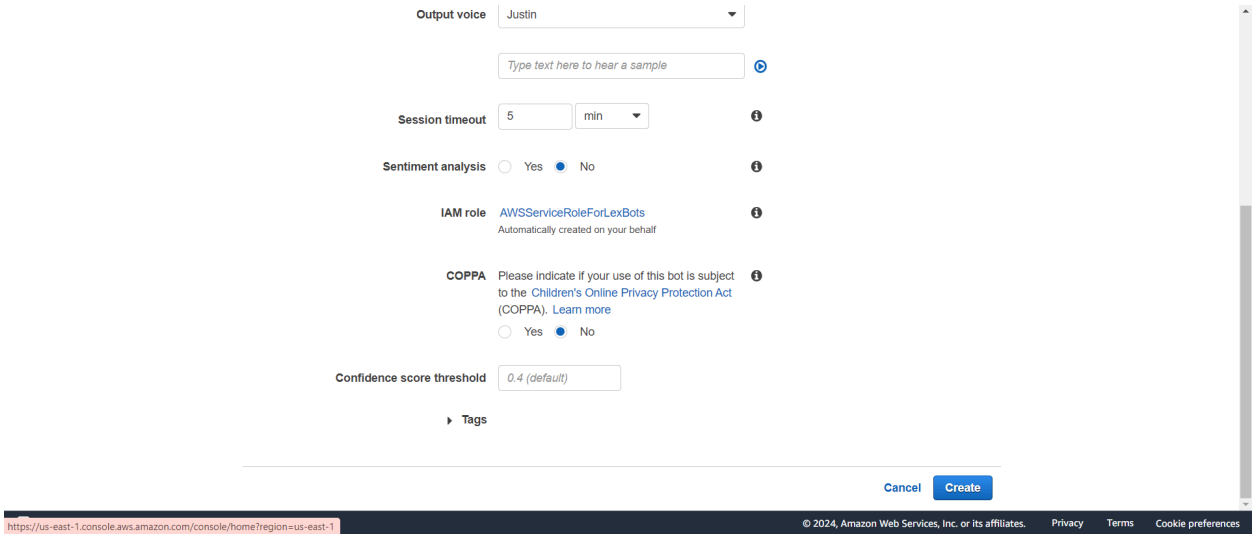


Screenshots of Workflow

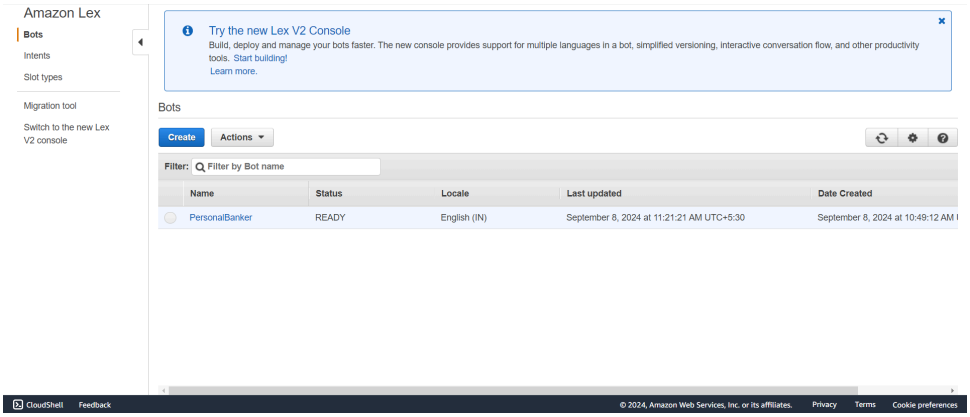
Phase 1: Utilization of Amazon Lex Service

- Creation of Lex Chatbot:

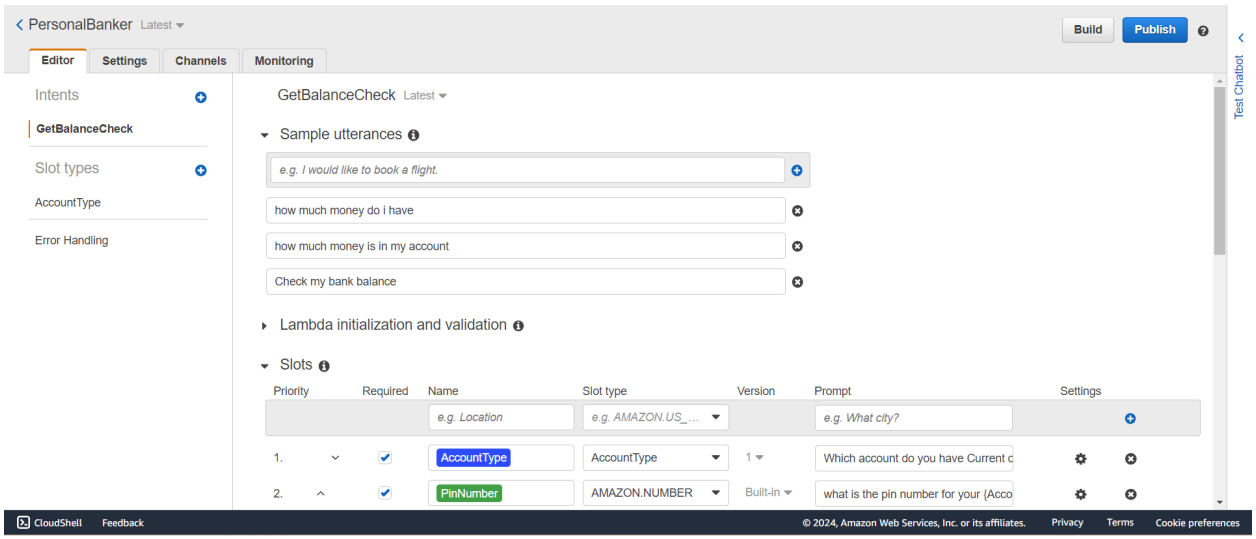


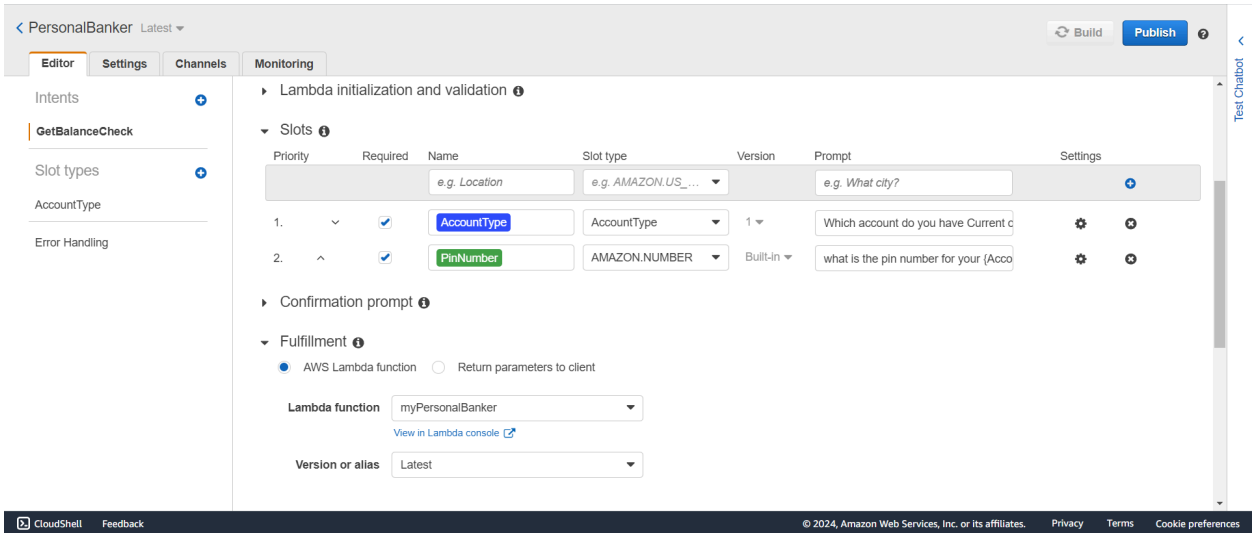


- Configuring the Lex Chatbot on V1 Console:

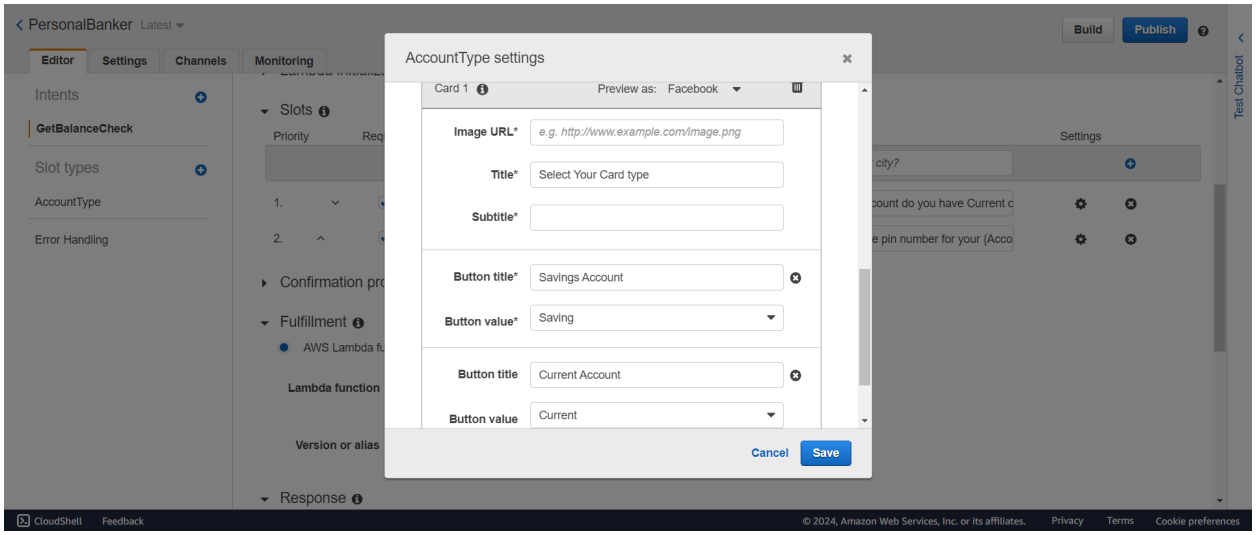


- Editing the Sample Utterances & Slot Types in Lex Chatbot:



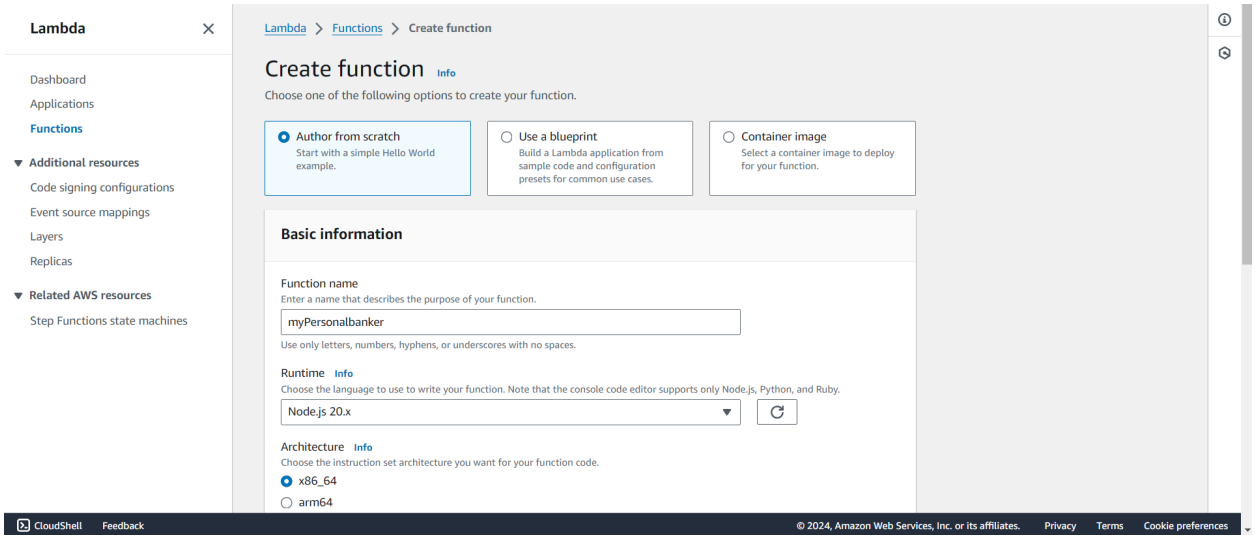


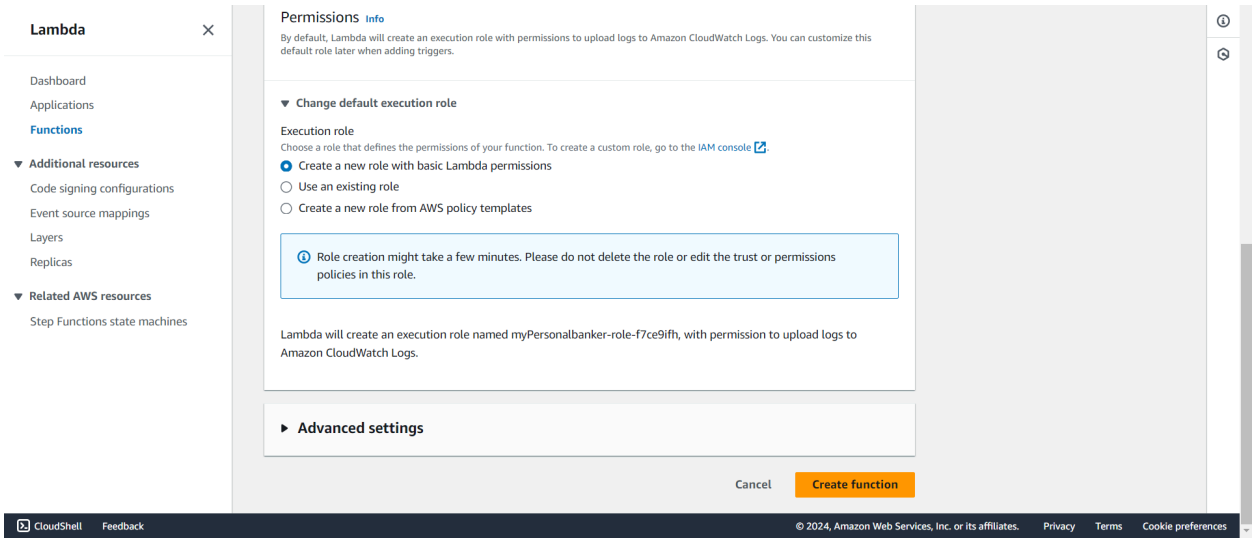
- Adding the Buttons in Slot Intents:

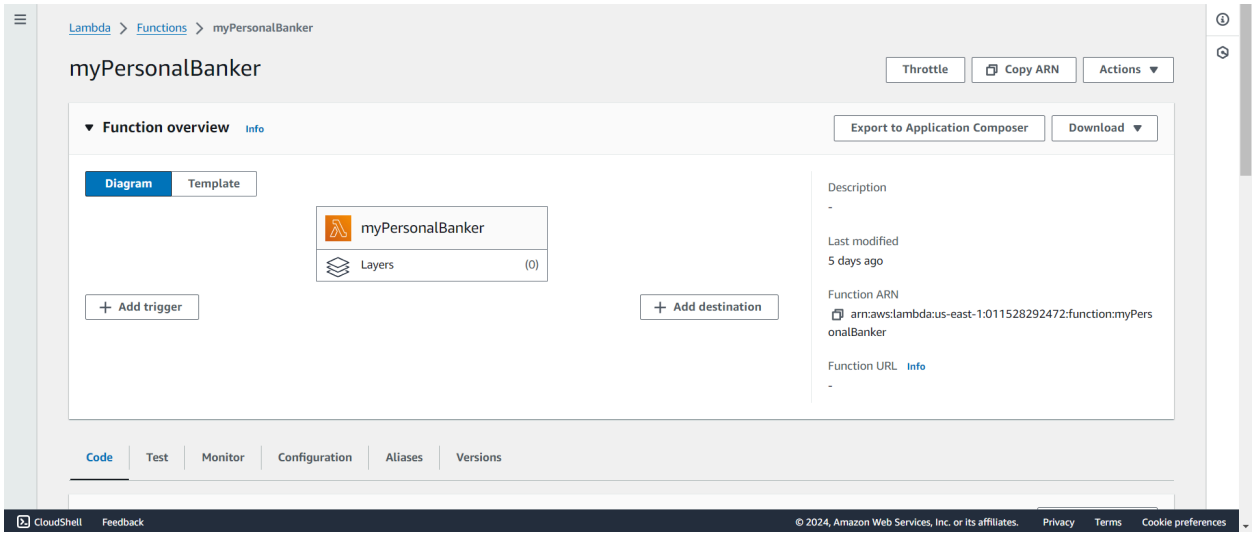


Phase 2: Utilization of Lambda Function Service by AWS

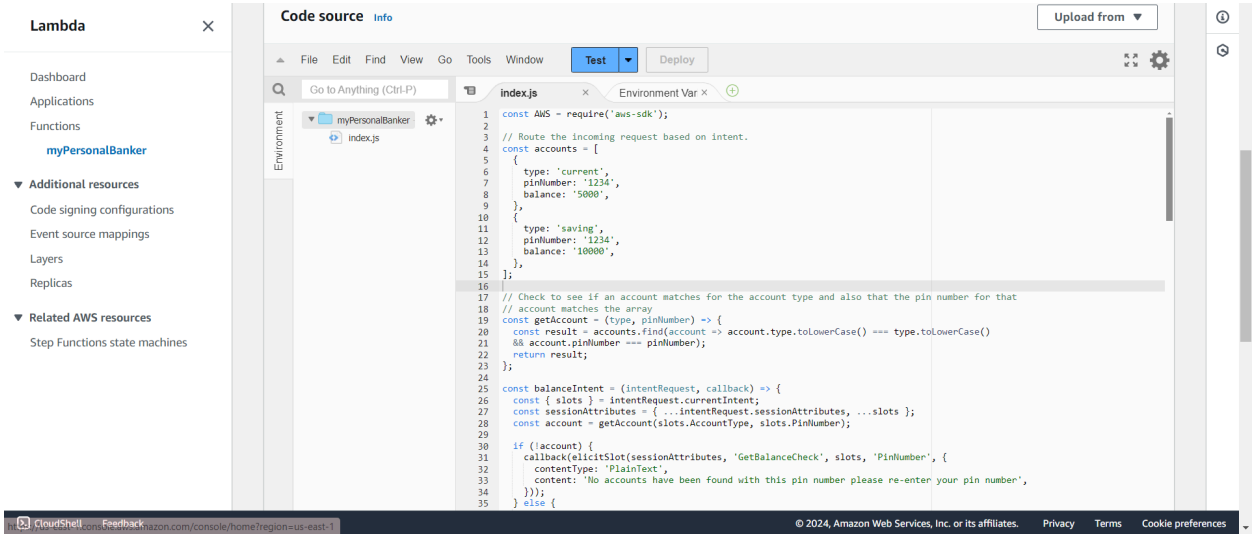
- Creation of myPersonalBanker Lambda Function:



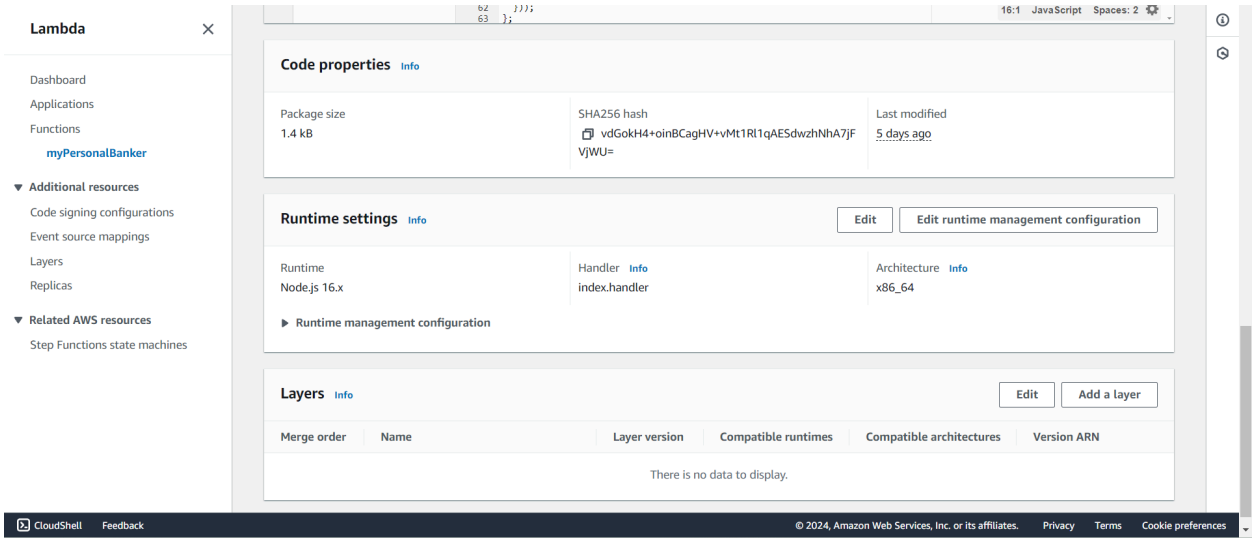




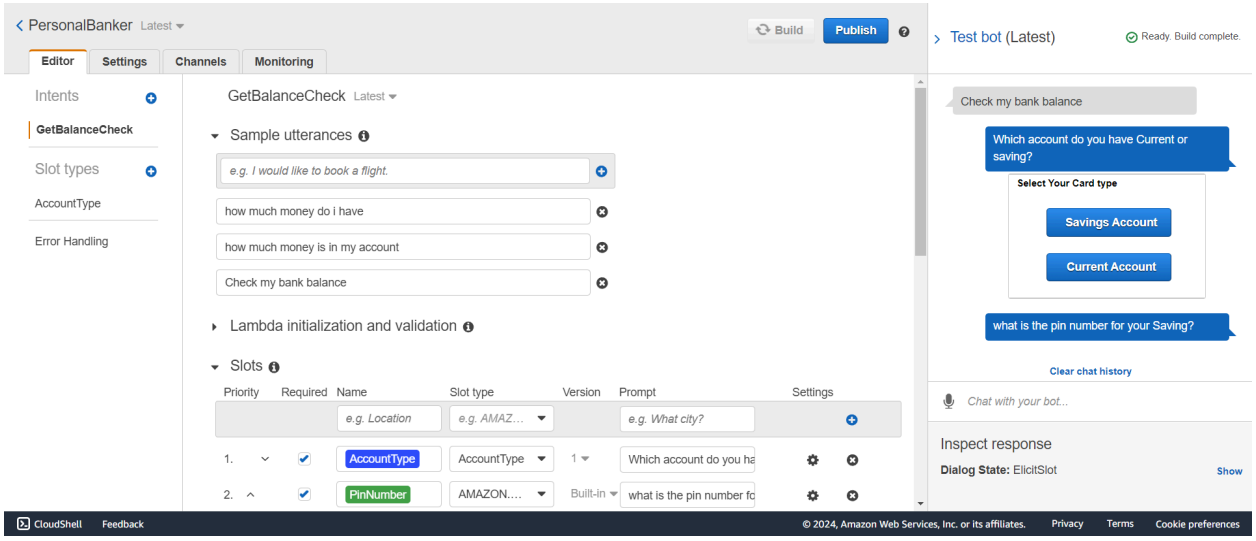
- Adding the JavaScript Code for fetching the Account details (Type, Pin & Balance). This helps in linking the “Lambda Function Service” with “Amazon Lex Service”.

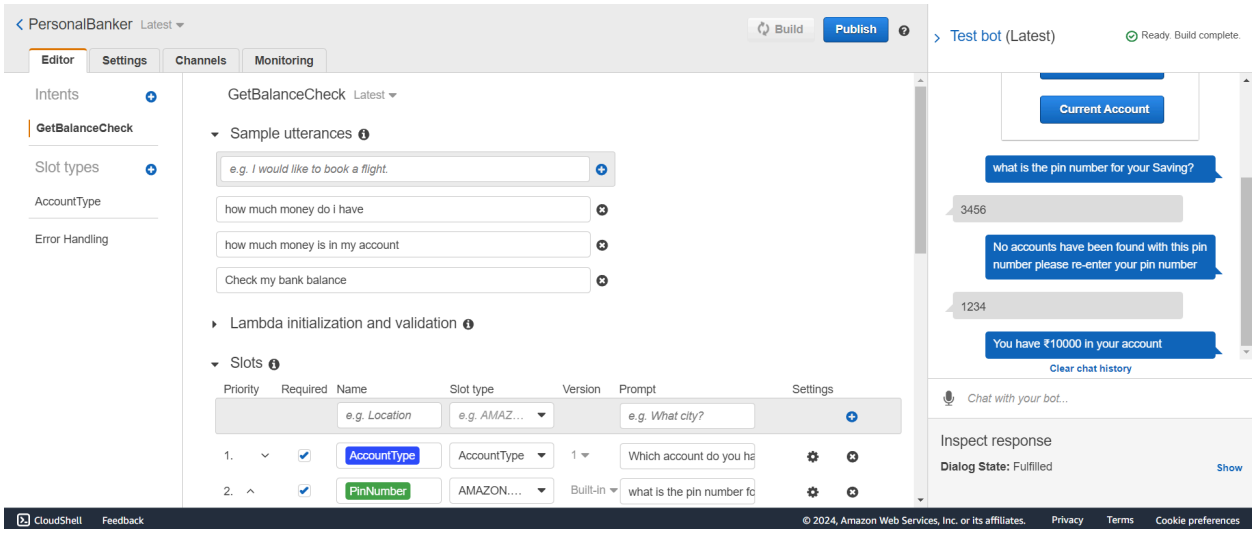


- Selection of Runtime Settings (Runtime, Handler, Architecture)



Phase 3: Building the Intents & Testing the Chatbot





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.