SAPIENZA
UNIVERSITÀ DI ROMA

Food Ordering chatbot using Amazon Lex

Prepared By:
Waddah Alhajar – 2049298
Alhajar.2049298@studenti.uniroma1.it

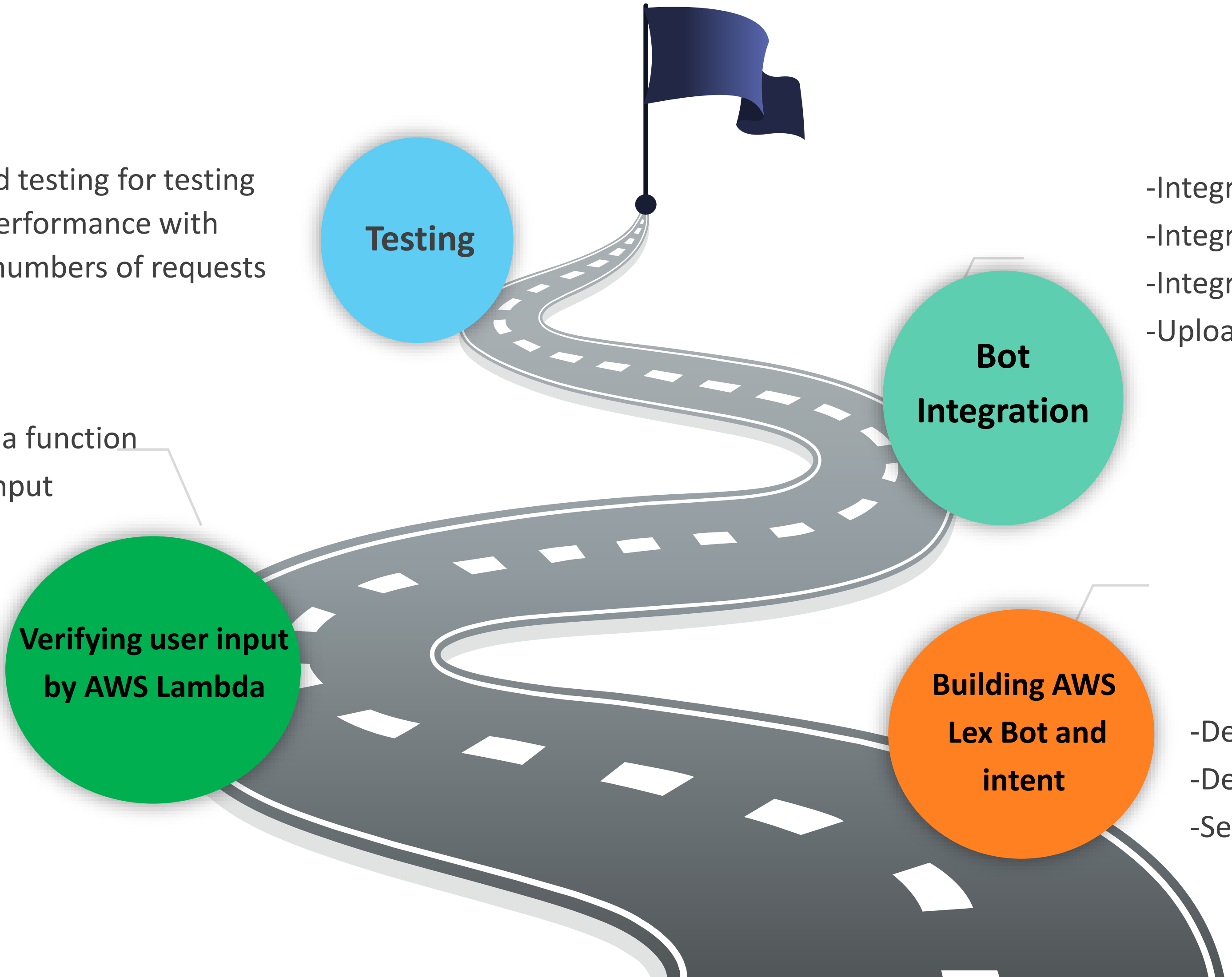
Overview

The project aims to develop a conversational chatbot that facilitates the process of ordering burgers. The chatbot utilizes Amazon Lex, a service for building conversational interfaces, to handle natural language interactions with users.

It guides users through a series of prompts to gather all the necessary information for placing an order. The chatbot incorporates validation logic using Lambda function to ensure that the provided inputs are correct and complete.

The chatbot is integrated into a static website hosted on Amazon S3 service and it is connected to a Telegram bot using third party service called Kommunicate.

Approach


Building AWS Lex Bot and intent
Verifying user input by AWS Lambda
Bot Integration
Testing

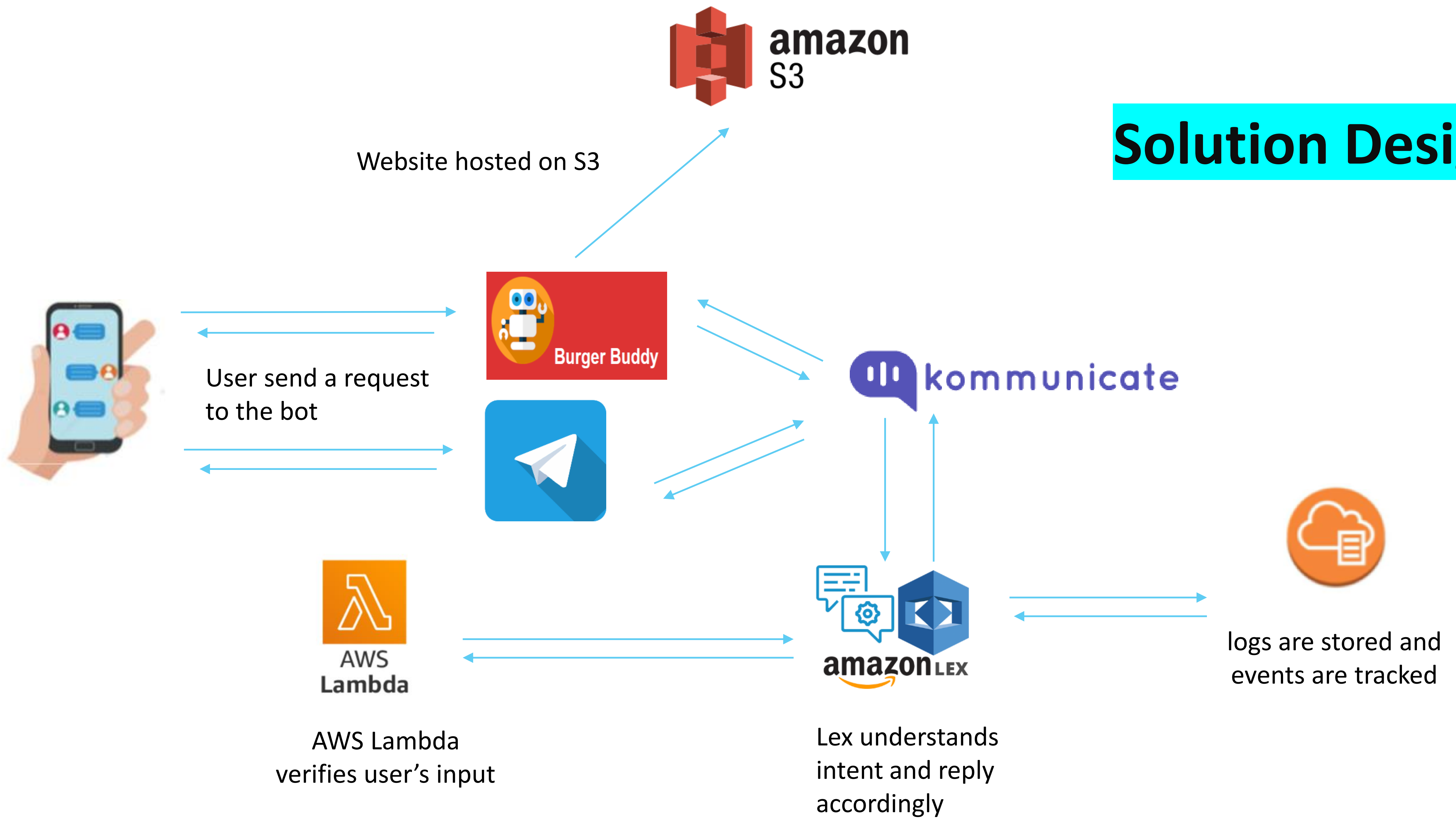
-Defining intents
-Defining slots
-Settings configuration

Building a Lambda function to verify input

-Integrating the Bot with Kommunicate
-Integrate Kommunicate with website
-Integrate with Telegram
-Upload website on S3

Crowd testing for testing bot performance with high numbers of requests

Solution Design

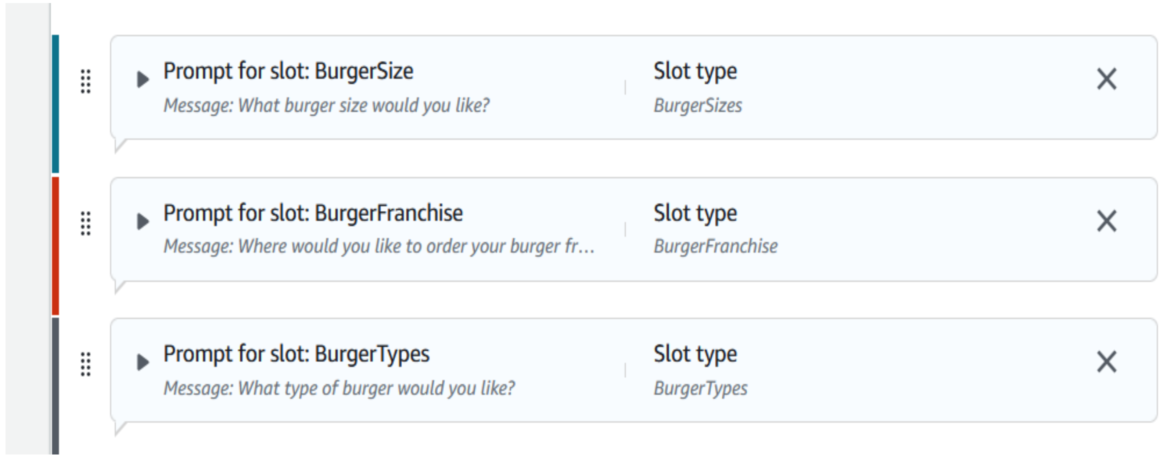

amazon S3
Website hosted on S3
Burger Buddy
User send a request to the bot
kommunicate
AWS Lambda
AWS Lambda verifies user’s input
amazon LEX
Lex understands intent and reply accordingly
logs are stored and events are tracked

Implementation and Deployment

Lex Chatbot : We start by defining the burger order intent, then defining the slots and make the customized slot types which are values that Amazon Lex uses to train the machine learning model to recognize values for a slot, and we provide sample utterances that capture the conversation flow and user interactions. Then we configure fulfilment actions. In our case we have 3 slots, burger type, burger size, and the restaurant. On successful fulfilment it will send a confirmation message and in case of failure it will send a custom message as well.

Lambda Function for Input Verification: we created a Lambda function that serves as the fulfilment code hook for the Lex chatbot. This Lambda function will receive the user input, validate it based on our defined logic, and respond accordingly. It will ensure that the input provided by the user meets the expected criteria before proceeding with further actions
1- The function receives the event and context parameters, representing the input event and runtime information
2- It retrieves the necessary information from the event, including the bot name, intent, and slots
3- Then calling the validate order function, passing the slots as a parameter. This function performs validation checks on the order details.
4- If the invocationSource in the event is 'DialogCodeHook', the function checks if the order validation result is not valid. If it's not valid, the function constructs a response to elicit the invalid slot and provides a message explaining the error. If it's valid, the function constructs a response to delegate the dialog to the next fulfilment code hook.
5-If the invocationSource is 'FulfillmentCodeHook', meaning the fulfillment code hook is invoked, the function constructs a response with a dialog action of type 'Close'. It marks the intent as fulfilled and includes a plain text message indicating that the order has been placed.
6-The constructed response is printed and returned as the output of the Lambda function.

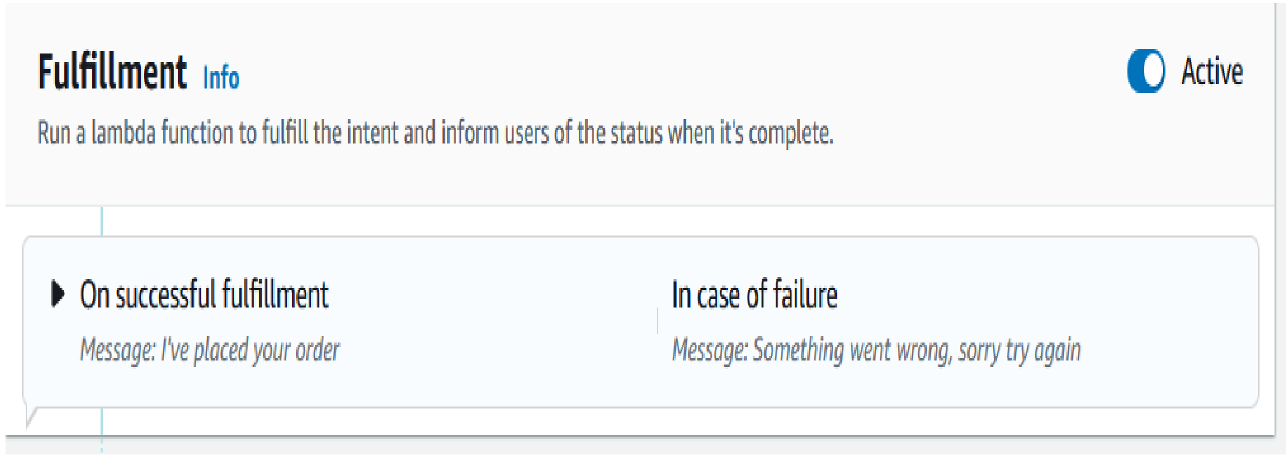
Some Bot Configuration screenshots


Prompt for slot: BurgerSize
Message: What burger size would you like?
Slot type
BurgerSizes

Prompt for slot: BurgerFranchise
Message: Where would you like to order your burger fr...
Slot type
BurgerFranchise

Prompt for slot: BurgerTypes
Message: What type of burger would you like?
Slot type
BurgerTypes

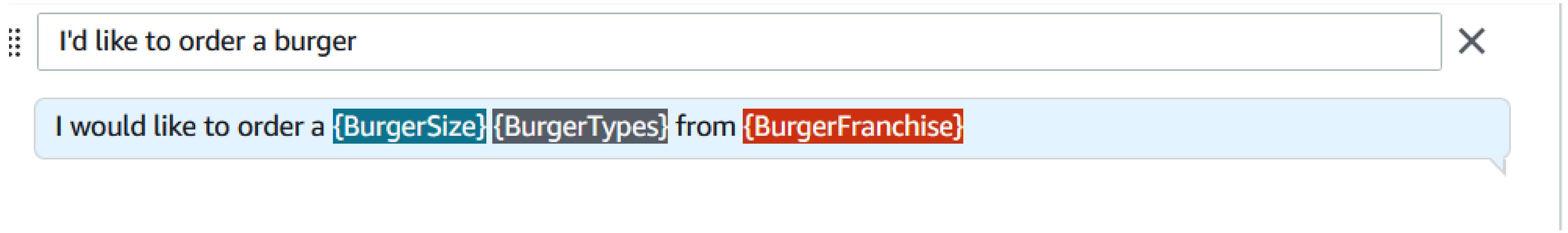
Slots Prompts


Fulfillment Info
Active
Run a lambda function to fulfill the intent and inform users of the status when it's complete.

On successful fulfillment
Message: I've placed your order

In case of failure
Message: Something went wrong, sorry try again

Fulfillment settings


I'd like to order a burger

I would like to order a {BurgerSize} {BurgerTypes} from {BurgerFranchise}

Sample utterances

Implementation and Deployment

Kommunicate Integration: we set up an account on the Kommunicate platform (www.kommunicate.io) and create a bot to connect with your Lex chatbot. Then enter the API key and the security credentials of our AWS user. And the bot name, region, and Alias name. This integration will enable communication between the Lex chatbot and the Kommunicate platform, allowing users to interact with the chatbot through various channels, including the website and Telegram.

Website Deployment on Amazon S3: After developing the basic static website using Html and CSS, we upload the website files, to the S3 bucket which is created and configured for static website hosting.

Website Integration with Kommunicate: Integrating Kommunicate into our website by adding the Kommunicate chat widget code snippet to the index page. This will embed the chat widget on our website allowing users to interact with the Lex chatbot directly.

Telegram Bot Integration with Kommunicate: we create a Telegram bot using bots father of telegram then configuring the access parameters in Kommunicate's dashboard. This integration will enable users to communicate with the Lex chatbot through the Telegram messaging platform.

Experimental Design

To test the performance of the Lambda function and Lex bot, we designed the following experiments taking into account the following constraints:

- Concurrency Limit: Lambda provides a total concurrency limit of 1,000 requests at a time for free, we can not exceed this limit.
- Lex Free Tier Limit: Lex free tier allows up to 10,000 requests per month.
- Order Fulfillment: Each order fulfillment process may take approximately 7 messages to complete.
- Crowd Testing: We opted for crowd testing with a group of friends. This approach provides more realistic user interactions and allows testing in diverse scenarios. We did two rounds of crowd testing and I performed a warm up for the system before that.

- Lex Test Workbench: This second experiment is to double check the correct output of the bot in different scenarios using Lex Test Workbench. It helped us verify the bot's responses and behavior.

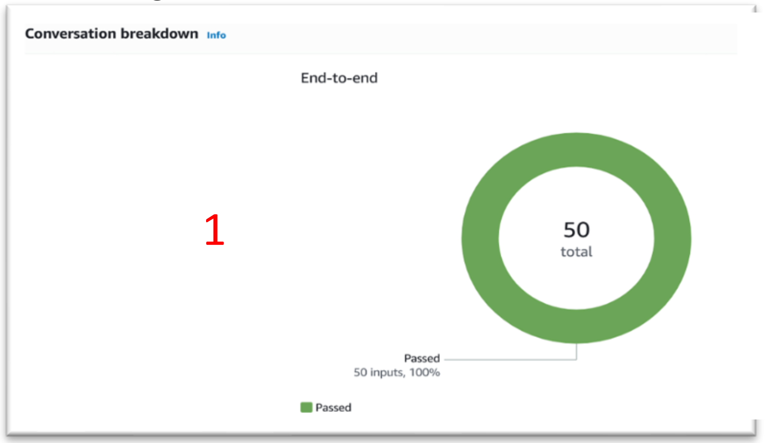
- During crowd testing, we observed that we were able to generate a max of 623 requests per one minute and 2.57K events for the Lambda function in five minutes.

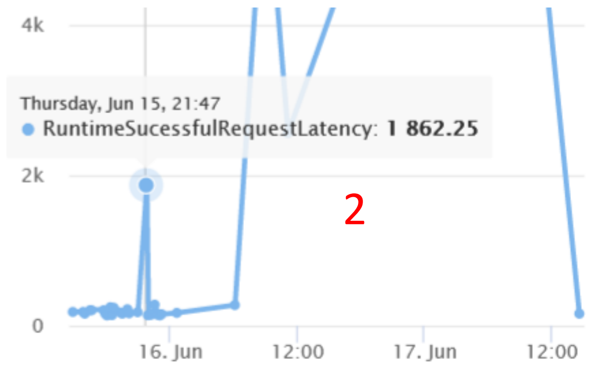
- Performance Analysis: After conducting the tests, we analyzed the CloudWatch logs to evaluate the performance of the Lambda function and Lex bot. We examined factors such as response time and error rate to assess their efficiency and identify any areas for improvement.

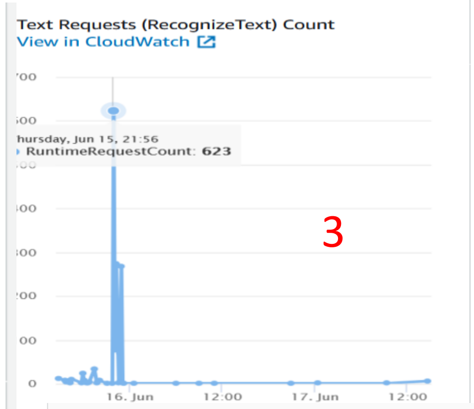
- By combining crowd testing with Lex Test Workbench and analyzing the performance metrics from CloudWatch logs, we aimed to obtain valuable insights into the scalability, reliability, and overall performance of our system.

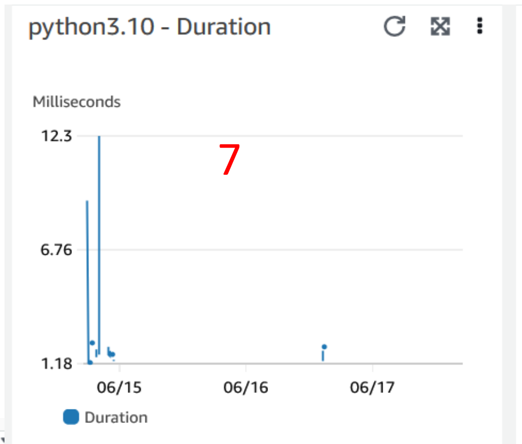
Experimental Results

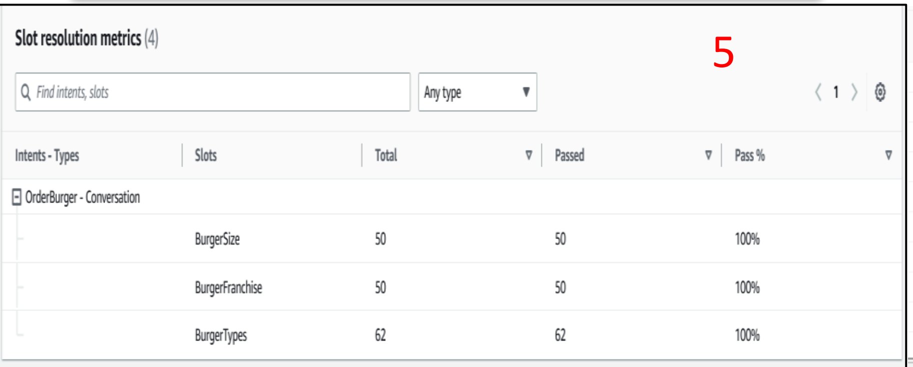
1- Lex Test Workbench Success: We achieved a remarkable 100% success rate in the Lex Test Workbench. Our bot flawlessly handled a rigorous 50 conversations test set encompassing all intents [1] [5] [6], demonstrating its robustness and accuracy in understanding user queries.
2- Swift warm up: needed time to reach optimal performance was 1.862 second [2], which considered rapid initialization.
3- Optimal Latency during Crowd Testing: the latency never exceeded a commendable 150 milliseconds [2], providing users with a remarkable experience.
4- High Request Generation: Leveraging the power of crowd testing, we generated a substantial volume of 623 requests within a single minute [3]. This extensive load testing allowed us to assess the system's scalability and resilience under high-demand scenarios.
5- Lambda Function Performance: Our Lambda function proved its mettle by flawlessly handling an impressive 2.57K requests within a five-minute timeframe [4]. Remarkably, this increased workload had no discernible impact on the runtime latency [2], while the highest duration of executing the Lambda function was 12.3 milliseconds [7], showcasing the stability and efficiency of our architecture.







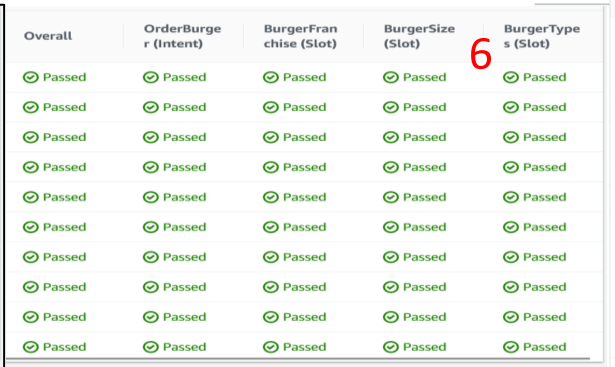



Slot resolution metrics (4)

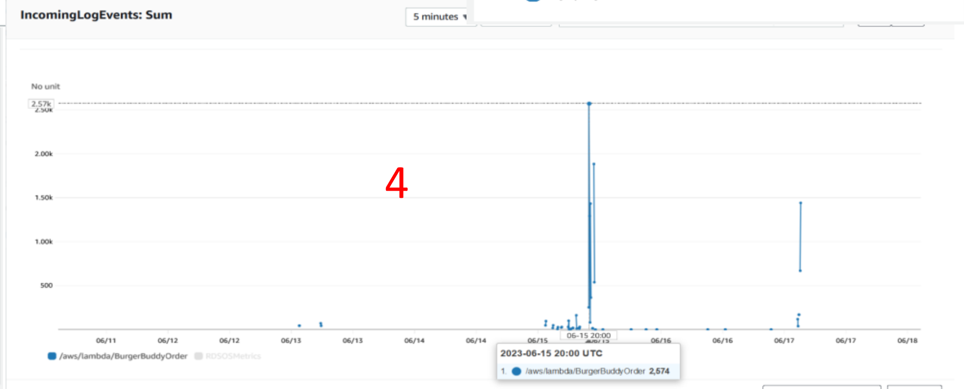
5

Find intents, slots | Any type | 1

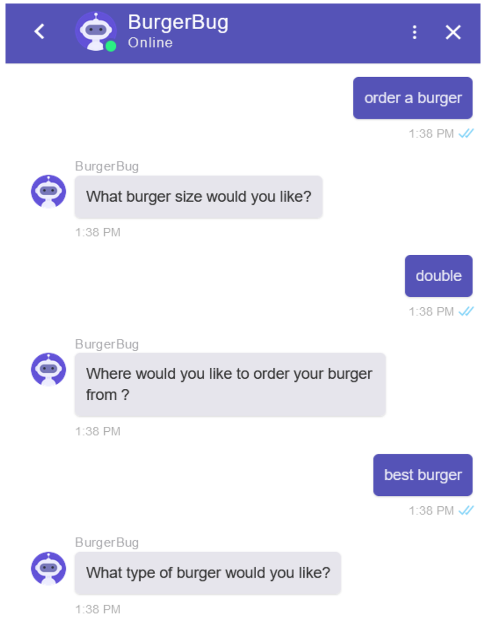
Intents - Types	Slots	Total	Passed	Pass %
OrderBurger - Conversation				
	BurgerSize	50	50	100%
	BurgerFranchise	50	50	100%
	BurgerTypes	62	62	100%


6

Overall	OrderBurger (Intent)	BurgerFranchise (Slot)	BurgerSize (Slot)	BurgerTypes (Slot)
Passed	Passed	Passed	Passed	Passed
Passed	Passed	Passed	Passed	Passed
Passed	Passed	Passed	Passed	Passed
Passed	Passed	Passed	Passed	Passed
Passed	Passed	Passed	Passed	Passed
Passed	Passed	Passed	Passed	Passed
Passed	Passed	Passed	Passed	Passed
Passed	Passed	Passed	Passed	Passed
Passed	Passed	Passed	Passed	Passed
Passed	Passed	Passed	Passed	Passed



Sample Outputs


BurgerBug
Online

order a burger
1:38 PM

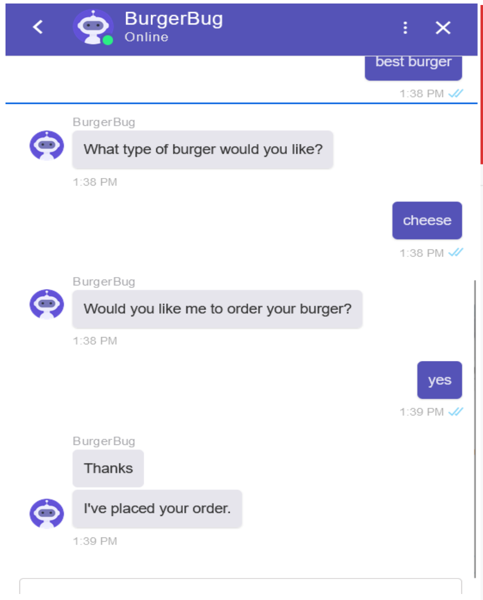
BurgerBug
What burger size would you like?
1:38 PM

double
1:38 PM

BurgerBug
Where would you like to order your burger from ?
1:38 PM

best burger
1:38 PM

BurgerBug
What type of burger would you like?
1:38 PM


BurgerBug
Online

best burger
1:38 PM

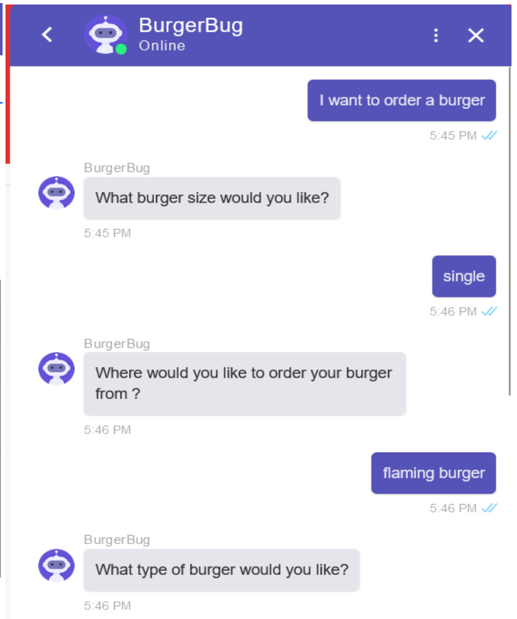
BurgerBug
What type of burger would you like?
1:38 PM

cheese
1:38 PM

BurgerBug
Would you like me to order your burger?
1:38 PM

yes
1:39 PM

BurgerBug
Thanks
I've placed your order.
1:39 PM


BurgerBug
Online

I want to order a burger
5:45 PM

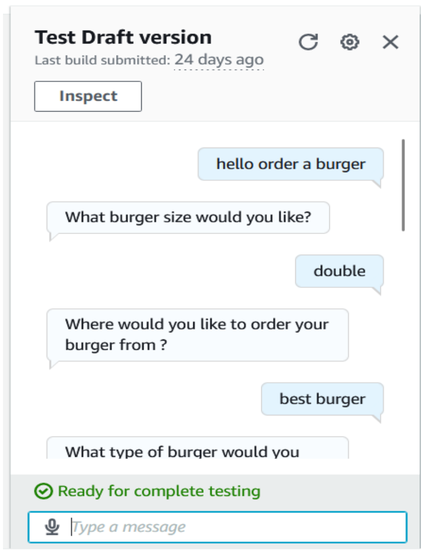
BurgerBug
What burger size would you like?
5:45 PM

single
5:46 PM

BurgerBug
Where would you like to order your burger from ?
5:46 PM

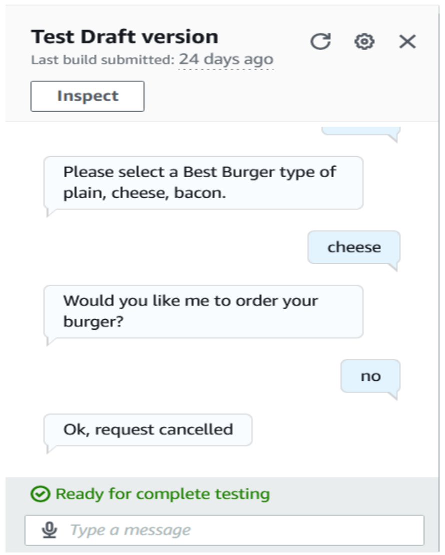
flaming burger
5:46 PM

BurgerBug
What type of burger would you like?
5:46 PM


Test Draft version
Last build submitted: 24 days ago
Inspect

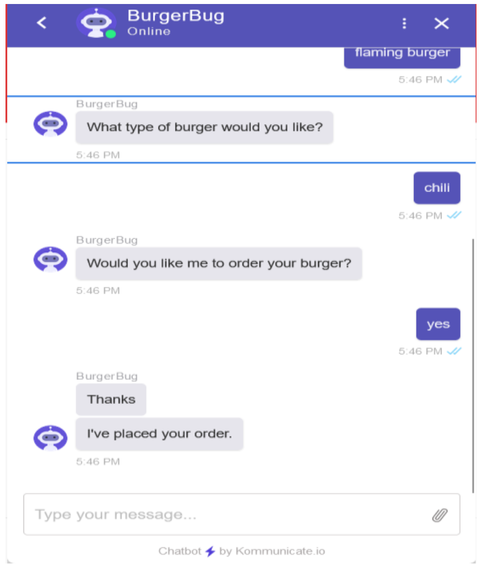
hello order a burger
What burger size would you like?
double
Where would you like to order your burger from ?
best burger
What type of burger would you

Ready for complete testing
Type a message


Test Draft version
Last build submitted: 24 days ago
Inspect

Please select a Best Burger type of plain, cheese, bacon.
cheese
Would you like me to order your burger?
no
Ok, request cancelled

Ready for complete testing
Type a message


BurgerBug
Online

flaming burger
5:46 PM

BurgerBug
What type of burger would you like?
5:46 PM

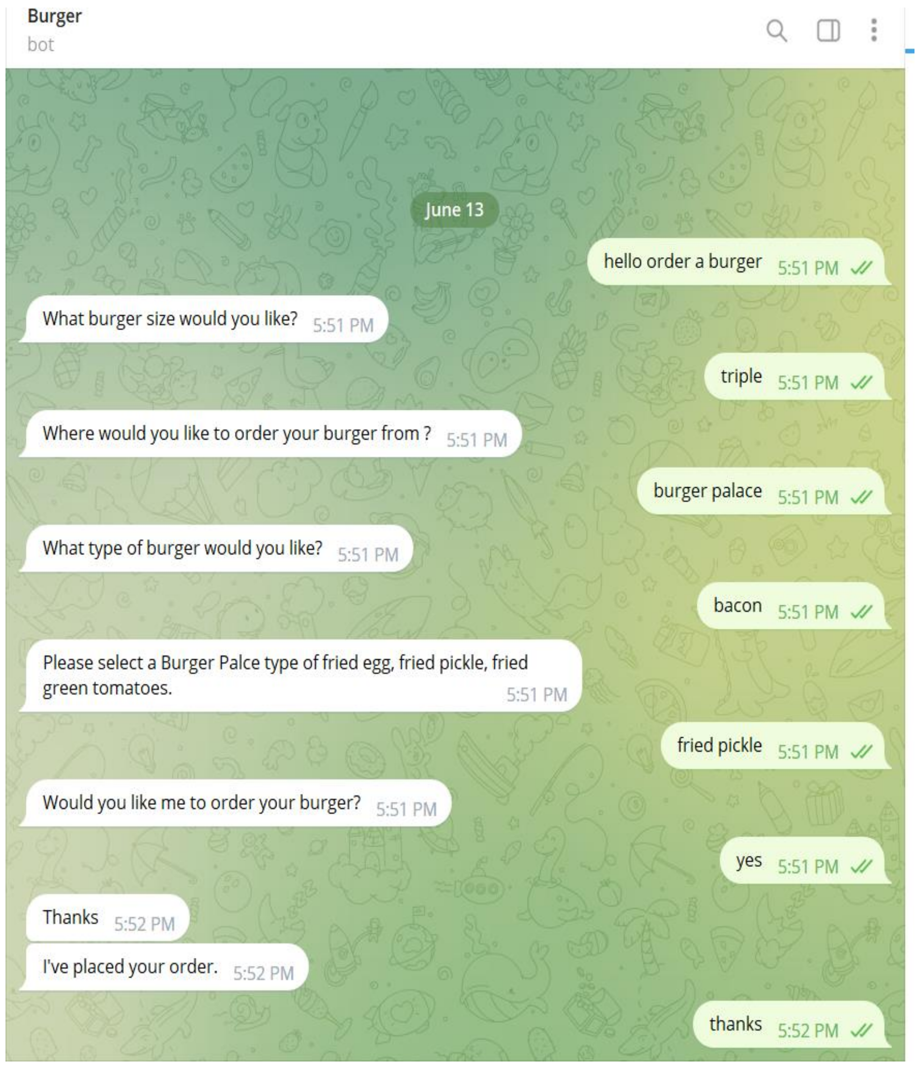
chili
5:46 PM

BurgerBug
Would you like me to order your burger?
5:46 PM

yes
5:46 PM

BurgerBug
Thanks
I've placed your order.
5:46 PM

Type your message...
Chatbot ⚡ by Kommunicate.io


Burger
bot

June 13

hello order a burger 5:51 PM
What burger size would you like? 5:51 PM
triple 5:51 PM
Where would you like to order your burger from ? 5:51 PM
burger palace 5:51 PM
What type of burger would you like? 5:51 PM
bacon 5:51 PM
Please select a Burger Palce type of fried egg, fried pickle, fried green tomatoes. 5:51 PM
fried pickle 5:51 PM
Would you like me to order your burger? 5:51 PM
yes 5:51 PM
Thanks 5:52 PM
I've placed your order. 5:52 PM
thanks 5:52 PM

Sapienza
Università di Roma

Thank You!