

CUSTOMER SERVICE CHATBOT USING AWS LAMBDA AND AWS LEX

A PROJECT REPORT

Submitted by

NATCHATHRA MANSHAKUTTY (RA2412033010030)

GANDAM SAI RAM (RA2412033010034)

JOAN RESHMI LAWRENCE (RA2412033010036)

Under the Guidance of

Dr. Annapurani Panaiyappan K

(Professor, Department of Networking and Communications)

In partial fulfillment of the requirements for the degree of

MASTERS OF TECHNOLOGY

CLOUD COMPUTING



DEPARTMENT OF NETWORKING AND COMMUNICATIONS

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR - 603 203

DEC 2024



SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

SRM INSTITUTE SCIENCE AND TECHNOLOGY

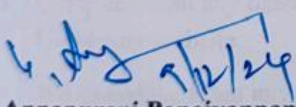
KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

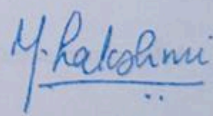
This is to certify that the research report entitled “**Customer service Chatbot using AWS LEX and AWS Lambda**” has been carried out by **Natchathra Manshakutty, Joan Reshmi Lawrance, and Gandam Sairam** under my supervision and guidance.

This work is submitted in partial fulfilment of the department for the **Master of Technology in Cloud Computing, Department of Networking and Communication, at SRM Institute of Science and Technology, Kattankulathur**, during the academic year 2024.

The research embodies the candidate’s original work , except where due references are made, and has not been previously submitted for the award of any degree or diploma


Dr. Annapurani Panaiyappan . K
(Supervisor)




Dr. M. Lakshmi
(Head of the Department)

Dr.M.LAKSHMI, B.E., M.E., Ph.D.,
Professor & Head
Dept. of Networking and Communications
School of Computing
College of Engineering and Technology
SRM Institute of Science and Technology
SRM Nagar, Kattankulathur - 603 203,
Chengalpattu Dist, Tamil Nadu, India.



SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

DEPARTMENT OF NETWORKING AND COMMUNICATIONS
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
OWN WORK DECLARATION FORM

DEGREE COURSE: M-TECH - CLOUD COMPUTING

STUDENT NAMES: Natchathra Manshakutty, Joan Reshmi Lawrance, Gandam Sairam

REGISTRATION NO.: RA2412033010030, RA2412033010034, RA2412033010036

TITLE OF WORK: - Customer service Chatbot using AWS LEX and AWS Lambda

We hereby certify that this assessment complies with the University's Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines. We confirm that all the work contained in this assessment is my / our own except where indicated, and that We have met the following conditions:

- Clearly referenced / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

We understand that any false claim for this work will be penalized in accordance with the university policies and regulations.

DECLARATION:

We are aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is our own work, except where indicated by referring, and that we have followed the good academic practices noted above.

If you are working in a group, please write your registration numbers and sign the date for every student in your group.

G. Sairam
G. Sairam

Natchathra
Natchathra

Reshmi
Reshmi



**DEPARTMENT OF NETWORKING AND COMMUNICATIONS
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

ACKNOWLEDGEMENT

We take opportunity to express our heartfelt gratitude to everyone who contribute to the successful completion of this research paper, as no research work is ever a solitary endeavour. It is the outcomes of collaboratives efforts, guidance, and support.

We extend our deepest gratitude to our **Research Supervisor (Dr. Annapurani Panaiyappan .K)**, for their invaluable guidance, encouragement, and throughout this research. Their expertise and insights have been instrumental in shaping this work and providing clarity at every stage.

We would like to thank the **Department of Networking and Communication, SRM Institute of Science and Technology, Kattankulathur**, for providing us with the necessary resources and an environment conducive to research.

TABLE OF CONTENTS

ABSTRACT.....	vi
LIST OF FIGURES.....	vii
CHAPTER : 1 Introduction.....	1
1.1 Objective.....	2
1.2 Problem Statement.....	2
CHAPTER : 2 Literature Review.....	3
CHAPTER : 3 Design of Customer service chatbot.....	4
3.1 Setting up Amazon Lex for Intent Handling	5
3.2 Configuring Amazon Lambda for Backend Integration	5
3.3 Multi-Channel Deployment.....	6
3.4 Edge-Case Handling.....	6
3.5 Testing and Fine-Tuning.....	7
3.6 Link Amazon Lex and Lambda.....	7
3.7 Testing Bot.....	8
3.8 Deploy your Bot.....	8
3.9 Monitor and Improve.....	9
CHAPTER : 4 Implementation.....	10
4.1 Implementing in Amazon Lex.....	10
4.2 Set-up Amazon Lex.....	11
4.3 Implementing for Aws Lambda.....	13
4.4 Deploying code in Aws Lambda.....	14
4.5 Cloud watch for Monitoring.....	15
4.5.1 Invocation metrics.....	16
4.5.2 Latency metrics.....	16
4.5.3 Error metrics.....	16
4.6 Connection of Lex and Lambda with Alias.....	17
4.7 Creating twilio for conversation with customers on WhatsApp.....	19
4.8 Connecting Twilio with AWS.....	20
CHAPTER : 5 OUTPUT.....	23
CHAPTER : 6 Conclusion & Future Enhancement.....	27
6.1 Future work.....	27
References.....	29

ABSTRACT

In the client support Chatbot project, using Amazon Lex and Amazon Lambda, the idea is to alter customer service by providing automatic replies to frequent customer requests, thus reducing response times as well as increasing overall happiness among customers. Serverless computing through Amazon Lambda analyzes and retrieves pertinent data on natural language understanding capabilities of Amazon Lex, which enable the chatbot to correctly understand the inputs of the client.

The integration enables the chatbot to offer assistance real time across many platforms that include; sites and applications on mobile. The objective of the chatbot will be to service a variety of customer support requests, including FAQs, order status updates, processing refund requests, providing product information, and helping with account-related issues. Through this simple automation of contacts, businesses can minimize some of the burden on their human agents, thereby lowering operational costs and saving the human team for more complicated, value-driving tasks. Step one in the development process is determination of the purposes, or jobs the chatbot can perform.

The second step is to set up Amazon Lex to understand what the user is typing regarding those jobs. Meanwhile, Amazon Lambda is the backend service; it talks to the existing support systems of the customers who acquire the data or perform specific tasks.

Other considerations also include testing and fine-tuning the chatbot so it will respond correctly and adequately to diverse consumer inputs, particularly tough cases or edge scenarios. It should be a multi-channel, conversational, customer support chatbot that can manage a wide range of inquiries and communicate with customers in an effective manner that maximizes user delight, cuts down the response time, and provides constant support.

That is significant because it could help cut down operating costs, enhance customer experiences, and provide assurance to businesses that they would continually support a stream of continuous information flow without having to completely rely on human effort

LIST OF FIGURES

FIG.NO	TITLE	PAGE NO.
3.1	Chatbot design	3
3.2	Testing of bot	7
4.1	Implementation of Amazon Lex	9
4.2	Creating hotel intent	10
4.3	Creating hotel inputs	11
4.4	Creating hotel intent details	11
4.5	Creating hotel booking Lambda	12
4.6	Deploying code in AWS Lambda function	13
4.7	Cloudwatch Log Groups	14
4.8	Cloudwatch Log Streams	15
4.9	Cloudwatch Metrics	16
4.10	Creation of alias for Lex and Lambda	17
4.11	Hotel bot alias	17
4.12	Creating Twilio account	18
4.13	Assigning number for Twilio	19
4.14	Channel integration for Twilio and AWS	20
4.15	Sandbox configuration	20
5.1	Connecting Twilio account for Whatsapp	22
5.2	Runtime success latency	23
5.3	Conversation flows	24
5.4	Fall back intents and Booking Hotel Intents	24
5.5	Visualization graph of Intents	25
5.6	Conversation of Twilio with user	25

CHAPTER 1

INTRODUCTION

The Customer Service Chatbot project is based on using Amazon Lex and Amazon Lambda. Seeks to address the escalating issues that businesses now face in the provision of efficient, scalable, and accessible customer service. Customers want prompt, personalized service in the digital age, but most businesses find it challenging to cope with the sheer volume of requests. Even though the use of Human agents only can be expensive, time and labor; hence, customer satisfaction is frequently reduced.

This project will bridge the gap by automating repeat customer care services using AI technology, cost reduction, and quickening both customer experience and response time. Enabling this chatbot was made possible by a foundation in Amazon Lambda's serverless architecture and natural language understanding (NLU) of Amazon Lex so it could answer most of the prevalent queries such as the status of an order, refund details, and product-related details.

The more customer demand grows, the more impossible it becomes for businesses to deal with the constraints of conventional models of customer service. This is a big push factor in the initiative. Normally, simple automated systems lack the strength to create meaningful experiences, which contributes to terrible user experience. For improving customer satisfaction through giving precise, time-sensitive information besides reducing the effect of human agents, this chatbot will concentrate on automating answers to frequently requested questions and managing more complex enquiries.

The scope of this project is limited to customer support functions, including product details, return processing, and standard inquiries. Sales and marketing as well as matters sensitive in nature requiring judgment will be excluded. This defined scope will ensure efficiency as well as reliability in its covered areas.

The project goes well with national priorities through encouraging automation and digital transformation and eventually supports firms in improving performance efficiency and service delivery to customers. This perspective, in a broad way, carries with it much larger initiatives toward encouraging AI and economic growth.

1.1 Objective

The Amazon Lex and Amazon Lambda Customer Service Chatbot project is designed to create a smart, scalable, and cost-effective solution that enhances customer service operations. By utilizing Amazon Lex, this chatbot can engage with customers in a conversational manner and make use of AI and NLP. It can perform an assortment of tasks such as answering frequently asked questions, resolving technical-related problems, and giving recommendations based on personal preferences. The chatbot also learns from past experiences, meaning that it becomes better and more equipped to respond to customers' needs over time.

Scalability is one of the ultimate goals of the project. Companies tend to have increased customer inquiries when they start growing, thus the need for the chatbot to be able to handle many interactions at a go. Amazon Lex and Amazon Lambda provide an architecture of a serverless model that automatically scales to match demand. Therefore, no business has to spend money on additional infrastructure when peak periods occur. This allows the chatbot to service customers in different regions, languages, or business units and is ideal for global operations.

1.2 Problem statement

Another major advantage is cost-effectiveness. Traditional customer service models require significant investments in personnel and infrastructure. Amazon Lex's pay-as-you-go pricing model cuts costs as businesses will have to pay only for the resources used. The chatbot can handle repetitive tasks such as store hours or order status updates to make the task easier for human agents to focus on more complex issues.

With this automation of mundane tasks and providing the capability of 24/7, the chatbot will ensure quick responses to the customer with maximum satisfaction. Customer dissatisfaction decreases since it receives an instant solution. Thus, Amazon Lex and Lambda Customer Service Chatbot is an easy and scalable approach for a solution to be efficiently executed for customers while cutting costs for businesses to promote their growth.

CHAPTER 2

LITERATURE REVIEW

Most of the researchers focus on the suitable frameworks and platforms for the development of chatbots. The tools such as Amazon Lex, which falls under Amazon Web Services, have the ability to design conversational interfaces,[7]. Such tools provide prebuilt functionalities in natural language processing, thus ideal for designing hotel booking bots[16]. Other researchers include, which integrated the chatbot in third-party platforms expanding more delivery of services into even integration of payment systems[11]. Chatbot points out the conversational AI solution is much customer-based and flexible to answer some customer questions or queries like availing rooms, or price or recommendations that seem to fit[20].

These studies, such as by chatbot, bring into consideration the inclusion of anthropomorphic attributes such as a personal avatar or style of talk just like human conversation in chatbots that can contribute to user satisfaction and foster trust between the user and the technology. Anthropomorphic characteristics[1][15], such as the hotel booking bot, introduce an entirely different quality into the interaction itself-thus, making it more pleasurable and engaging, particularly in service recovery contexts. Also, [17]careful approach to designing interaction is fundamental, involving intuitive interfaces and fast response times for creating a smooth and pleasurable user experience.

This is one of the underlying features that are key in developing and creating great chatbots, especially in hospitality.[2]This research postulates several new strategies in understanding the unwritten and sometimes subtle desires of the customers through advanced techniques of NLP. For instance, a good chatbot would infer a customer's desire for room size or location closer to tourist attractions solely through indirect questions that might not even state these preferences explicitly.

The implementation of deep neural networks is quite supportive, aiming to increase precision and accuracy levels of these processes, especially while dealing with hotel booking systems to manage simple to intricate and complex queries[17].

Chapter 3

DESIGN OF CUSTOMER SERVICE CHATBOT

There is a Dialogue Management System which has been designed by the chatbot itself, manages to push the flow of the conversation ahead in sequence while keeping multiple user inputs in context. There would always be an information repository-be it Knowledge Base or the Database, which is known to deliver answers or take actions with the help of room check availability and making a reservation. It also makes for connectivity with third-party systems through APIs-things like a third-party payment gateway, Hotel Management Systems, etc. As shown in Fig 3.1 This architecture is hosted on some form of Cloud Infrastructure, such as AWS, that offers benefits like scalability and reliability and real-time processing. All these give rise to a responsive and effective chatbot experience.

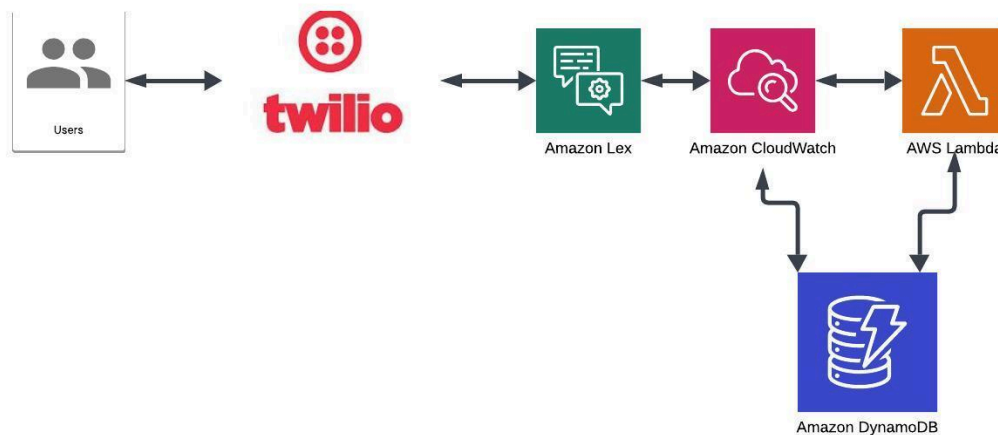


Fig 3.1 Chatbot design

In itself, the architecture of a customer service chatbot is fundamentally based upon several essential elements that, when combined together, serve to make communication between the bot and the user smooth and efficient. At its very heart is a module called Natural Language Processing, which essentially interprets what the user feeds into it. It achieves this in several advanced techniques, for example, through tokenization; it breaks the text down into manageable bits, or by sentiment analysis, through which it ascertains the emotional content behind the words. And as for how perfectly the NLP module would integrate with the Intent Recognition Engine, this pivotal component makes it systematically map the user's queries into specific

actions or appropriate responses, therefore improving the efficiency and responsiveness of the chatbot overall.

3.1. Setting Up Amazon Lex for Intent Handling

This allows you to create an Amazon Lex conversation chatbot that can be developed to quickly handle customer service queries efficiently. Here, the bot is set to handle the "OrderStatus" intent where it is supposed to collect the basic details, which in this case is order number through slots, then processes the request via AWS Lambda. When a customer asks about their order status, Lex invokes the Lambda function using the slot values captured earlier, such as the order number. The Lambda function emulates a query to a backend system - which, in a production environment would be a live database or API - to retrieve the order's current status and delivery details.

After retrieving the status, the Lambda function formats a response, sending it back to Lex and delivering that back to the customer. This way, the processing becomes smooth and eliminates repetitive tasks such as order inquiry responses, ensuring a scalable, cost-effective, and efficient customer support model. The chatbot takes advantage of Amazon Lex's AI capability of intent matching and AWS Lambda's serverless capability, thereby scaling according to user needs and bringing high-precision, real-time information without the need for any manual intervention.

3.2 Configuring Amazon Lambda for Backend Integration

Configuration of Amazon Lambda for backend integration: Amazon Lambda functions are integrated with other external systems such as databases or APIs, that need to be fetched or modified. In this context, the Lambda function connects with a DynamoDB database in order to retrieve details on an order. The function, `get_order_details`, queries the "OrdersTable" DynamoDB table using the order number to get specific order information based on the status and date for delivery. The Lambda function processes this data and formats it into a user-friendly response, such as "Order [order_number] is [status]. Estimated delivery date: [delivery_date]." This backend integration enables the chatbot to provide real-time, dynamic information to users by querying live data from a database, ensuring that responses are accurate and up-to-date. Businesses can efficiently store and manage huge volumes of order data with DynamoDB, then integrate them in their workflow into their chatbots for the customer's added convenience.

3.3 Multi-Channel Deployment

Multi-channel deployment would enable the Amazon Lex chatbot to reach users on different devices and platforms, such as websites, mobile applications, or social media. For web-based deployment, the AWS SDK can be used to integrate Amazon Lex by adding a chat widget to a website. Users could then interact with the chatbot directly. This is achieved by embedding a script that connects the chatbot to the Lex service using the AWS SDK. In the given example, the script uses the class of `'AWS.LexRuntime'` to send user input such as "Track my order" to the Lex bot titled "CustomerSupportBot" in the `'postText'` call of the API. The bot processes the input and shows relevant information such as about the status of an order. This configuration will provide complete interaction between the user with the chatbot on the web site and will be seamless on the website. Nevertheless, Lex can be incorporated into other mobile applications or social media like Facebook Messenger and WhatsApp using APIs, thus providing businesses with the opportunity to reach customers on their preferred channels. This achieves the business's objective of combining accessibility and better customer service.

3.4 Edge-Case Handling

Edge-case handling in Amazon Lex is important because when there are complex or ambiguous questions, the bot will likely not understand them. This could be achieved by creating a Fallback Intent, which engages whenever Lex cannot match any input from the user to the available predefined intent. The bot sends a message like "Sorry, I didn't catch your request. Can I transfer you to a human?" to escalate the conversation to a human agent. This makes sure that the users are not left without assistance and are connected with a human if their problem cannot be solved by the bot.

Furthermore, for the bot to better comprehend subtle customer inputs, there are NLP tools, such as Amazon Comprehend. It analyzes the sentiment or detects key phrases in the user's message, thereby aiding the bot in understanding the emotional tone of the question or extracting important information from ambiguous and complex questions. For example, if a customer user gets frustrated, the bot might detect negative sentiment and insist on escalating to a human agent for a better customer experience. In sum, the fallback mechanisms combined with more sophisticated NLP capabilities ensure that a wide range of customer inputs could be taken care of by the chatbot, resulting in accurate responses while offering seamless escalation when needed.

3.5 Testing and Fine-Tuning

Testing and tuning is crucial to the reliability and performance of an Amazon Lex chatbot with AWS Lambda. Lambda function unit testing corresponds to simulating backend system responses using mock data to ensure that the function behaves as expected. For instance, the function `test_handle_order_status` uses test data for a mock order status query and then verifies that it returns an object that the function should be expected to output, with this including an expected message such as "Your order". This form of testing confirms that the Lambda function will serve requests correctly. Performance testing is equally important to simulate high traffic and assess how the chatbot handles concurrent requests. By simulating large volumes of users interacting with the bot at the same time, businesses can test its scalability and identify any bottlenecks or issues related to latency or system capacity. This is crucial to ensure that the chatbot can handle peak traffic periods without affecting performance.

This ranges from testing for invalid or incomplete data, such as missing order numbers, and irregular phrasing or typos, where users may not phrase their queries in standard formats. For instance, the ability of the bot to deal with variations like "Where my order is?" and still return a meaningful response could be tested. Such tests form the basis for fine-tuning the system to enhance the robustness of the chatbot so that it delivers consistent, accurate, and user-friendly experiences even in not-so-ideal scenarios.

3.6 Link Amazon Lex and Lambda

Go to the Lex Console, select the bot you have just created, and click through into the bot's configuration. From there, click on the OrderStatus intent you have just created, the one specifically designed to handle customer requests about order tracking. Select the Fulfillment section of the intent configuration and choose to use an AWS Lambda function for fulfillment. You will then select the Lambda function you previously created, titled CustomerSupportFunction, from the available list of Lambda functions. After you have selected your Lambda function, click Save to save the configuration and then rebuild the bot to apply the new configuration. It is with this integration that when the OrderStatus intent is triggered by a user, Lex invokes the CustomerSupportFunction Lambda function, which then processes the request and returns an appropriate response. By linking the Lambda function to the intent, you enable dynamic, server-side interactions that could provide real-time information to your users.

3.7 Testing bot

To test your bot, you will go to the Test Bot section in the Lex Console. The interface lets you simulate interactions with the bot before its deployment. In the test window, enter sample utterances such as "Where is my order?" or "Track my package 12345," which are expected to trigger the OrderStatusintent you set up earlier. These inputs are expected to cause Lex to invoke the associated CustomerSupportFunction Lambda function. The bot should return the order status and date of delivery from the simulated backend data. For example, "Your order 12345 is Shipped and is expected to arrive by 2024-12-05". In Fig 3.7 Confirmation that the bot returns with the correct information confirms that the Lambda function is correctly hooked up to the intent and that the system is correctly processing the customer queries. This is a crucial testing step in ascertaining that the bot will indeed function as expected before it is released for actual usage.

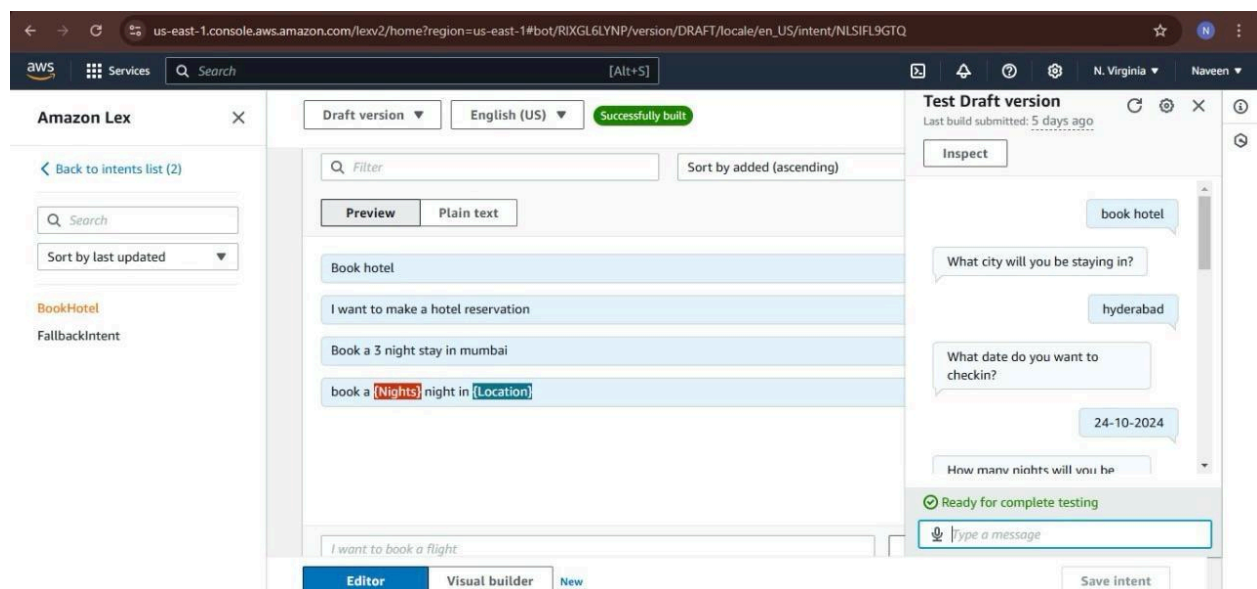


Fig 3.2 Testing of bot

3.8 Deploy Your Bot

To deploy your Amazon Lex chatbot, you have several deployment options depending on your target platform. For a web app, you can use the AWS Amplify framework or a custom front-end to easily embed the chatbot into your website. One quick deployment option is to use the Lex Web UI, which provides a pre-built interface for integrating the chatbot with minimal setup. For mobile apps, you can integrate the chatbot using the Lex API, allowing the bot to function within

both iOS and Android applications, providing users with a seamless chat experience right on their devices. In addition, the bot can be deployed on third-party channels such as Facebook Messenger, Slack, or WhatsApp. These can be integrated into webhooks or using the appropriate APIs to allow you to further reach out and include interaction with customers on your chosen popular messaging platforms. Each method ensures that your chatbot is able to be reached by the users in all the places they desire to reach it, making it available and offering users consistent experience on various devices and platforms.

3.9 Monitor and Improve

Use Amazon CloudWatch:

- a. Monitor Lambda execution logs for errors or performance bottlenecks.
- b. Add log statements in Lambda for debugging
- c. Analyze Lex Logs
- d. Enable Lex conversation logs to understand user interactions and improve the bot's accuracy.

Refine the Bot:

- e. Add more intents and slots based on customer feedback.
- f. Handle edge cases using fallback intents.

CHAPTER 4

IMPLEMENTATION

4.1 Implementing in Amazon Lex:

Implementing a customer-based chatbot using Amazon Lex and AWS Lambda involves creating a seamless interaction system that handles customer queries efficiently. First, Amazon Lex is used to build the chatbot by defining intents, which represent the goals or actions users want to achieve, such as checking order status or getting product information. These intents are paired with sample user phrases and optional slots for necessary details (e.g., order number). For such dynamic fulfillment of intents, AWS Lambda functions are created for the backend logic to process such queries as querying a database or communicating with external APIs. Then, these Lambda functions are linked to certain intents in Lex, where the chatbot can obtain real-time data and present the information accordingly. In Fig 4.4 the testing and training of the bot is also crucial in making sure that it can give appropriate responses to users' inputs. After successful testing, the bot is integrated with web or mobile applications through Amazon Lex SDK, where customers can be interacted with in real-time. It is possible to monitor the system continuously with Amazon CloudWatch to optimize performance and to handle errors. This solution, leveraging Amazon Lex for natural language understanding and AWS Lambda for backend automation, offers a scalable, cost-effective, and efficient method of automating customer support operations to improve response times and overall user satisfaction.

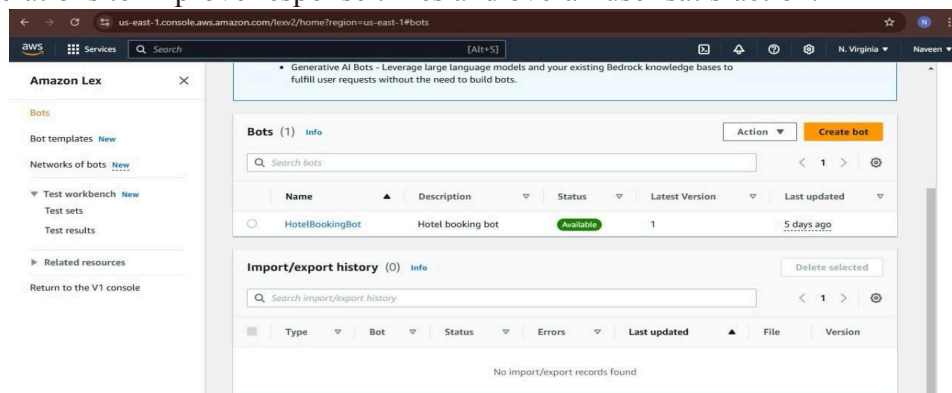


Fig 4.1 Creating Hotel Booking Intent

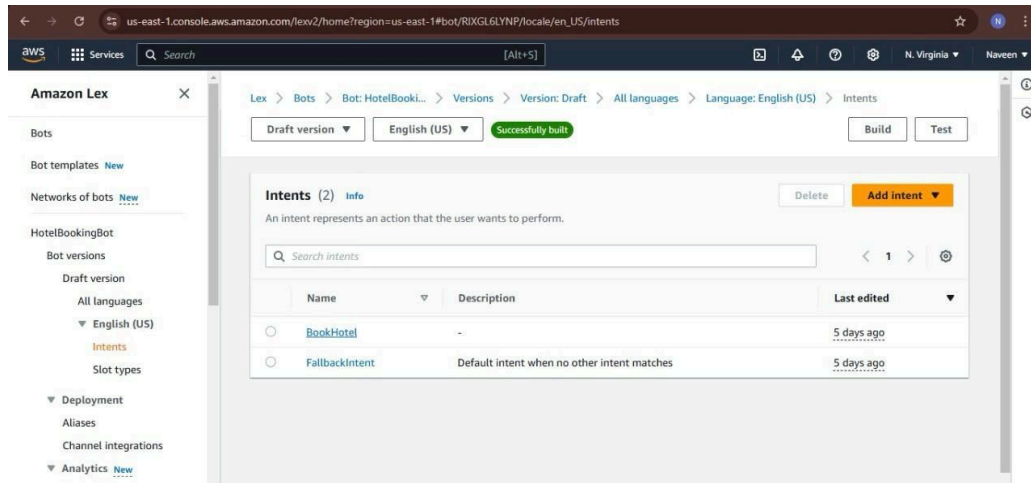


Fig 4.2 Creating hotel intent

4.2 Set Up Amazon Lex

Steps:

1. Create a Bot in Amazon Lex:
 - Go to the Lex Console and click Create Bot.
 - Give your bot a name (e.g., CustomerSupportBot).
 - Choose None for the IAM role initially; Lex will create the necessary permissions.
 - Select Create.
2. Define Intents:
 - Add intents for different tasks, like checking order status, processing refunds, etc.
 - Example Intent: OrderStatus
 - Sample Utterances:
 - "Where is my order?"
 - "Track my package."
 - "What's the status of my order?"
 - Slots:
 - OrderNumber: Slot type: AMAZON.Number.
 - Fulfillment: Choose AWS Lambda function as the fulfillment mechanism.
3. Enable Error Handling:
 - Add a fallback intent to handle unrecognized queries gracefully.
4. Save and Build the Bot:
 - Click Build to compile your bot's configuration.

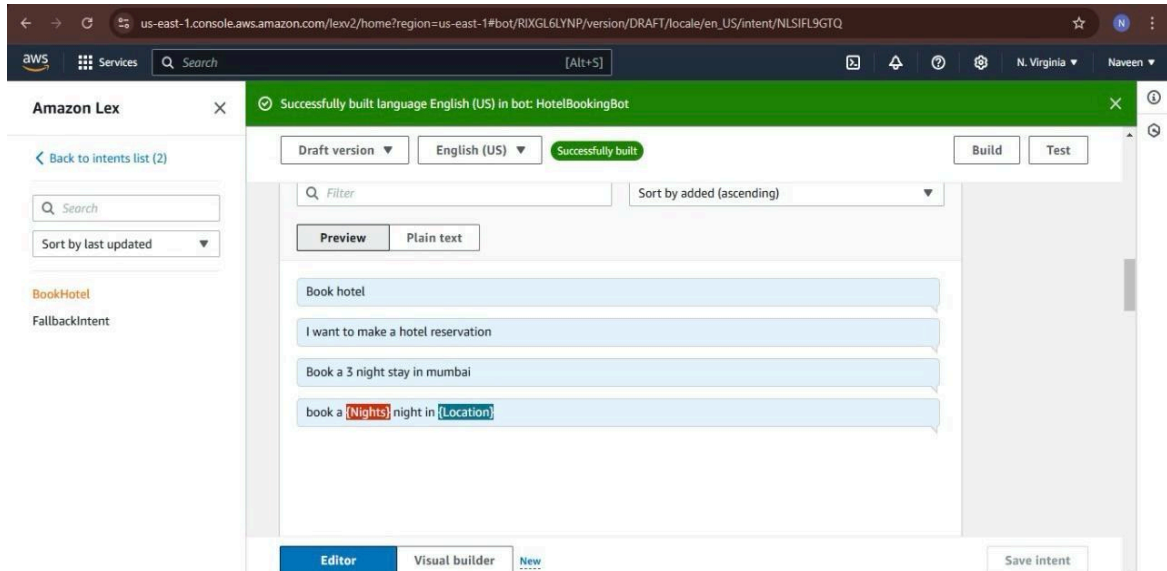


Fig 4.3 Creating Hotel Inputs

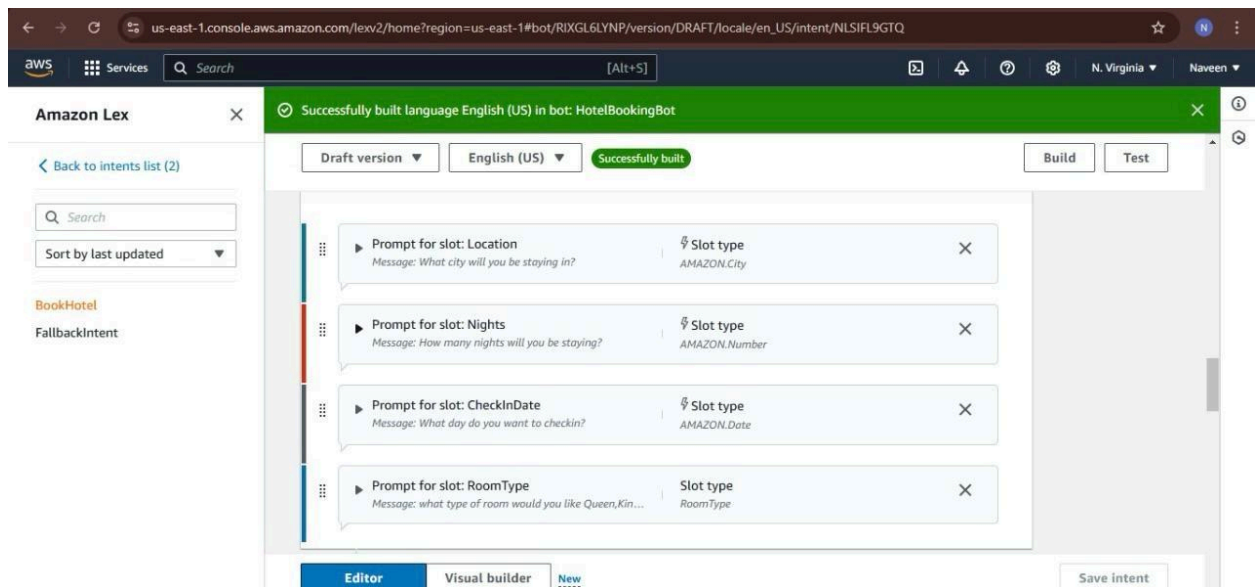


Fig 4.4 Assigning Slot values in Hotel intent

4.3 Implementing for aws lambda:

Implementing AWS Lambda for a customer-based chatbot with Amazon Lex allows you to automate backend processes and enhance the interactivity of the chatbot. For instance, imagine a customer who wants to know the status of his order. The chatbot, powered by Amazon Lex, captures the user's intent, such as "Track my order," and extracts relevant information, like the order number, using Lex's slot functionality. The order's current status is retrieved from an external system or database, such as Amazon DynamoDB, using an AWS Lambda function to which this data is passed. The request is processed by Lambda, and the response is returned to Lex. Lex communicates the result to the customer. This allows real-time interactions, where customers can receive personalized information without human assistance. This allows the Lambda function to handle the back-end logic and integrations behind the scenes, ensuring that the chatbot can efficiently respond to various user queries such as retrieving order details, checking product availability, or processing requests. As shown in Fig 4.5 this serverless architecture allows businesses to scale their customer service operations effectively and cost-efficiently since Lambda runs only when events occur and does not require continuous server management.

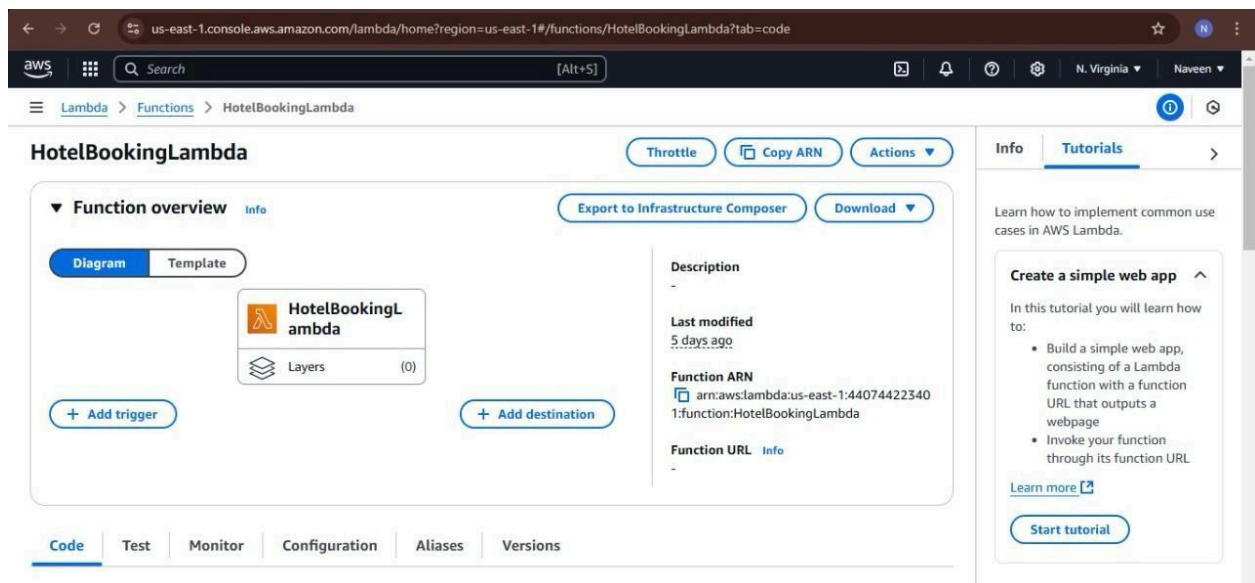


Fig 4.5 Creating Hotel Booking Lambda

4.4 Deploying code in aws lambda

To deploy code in AWS Lambda, in Fig 4.6 you have to go through several steps to ensure that your function is properly set up, configured, and ready to execute. First, you need to write the code for the Lambda function, typically in a supported runtime like Python, Node.js, or Java. Once your code is ready, you deploy it by creating a new Lambda function in the AWS Management Console. During this process, you'll specify the function name, select the runtime, and assign an appropriate IAM role that grants necessary permissions for the function to access other AWS services, such as Amazon S3 or DynamoDB. Following function code upload, either pasted into the console directly or uploaded via a ZIP file, environment variables can be configured for memory and timeouts to be set on your Lambda function to help it run with maximum efficiency.

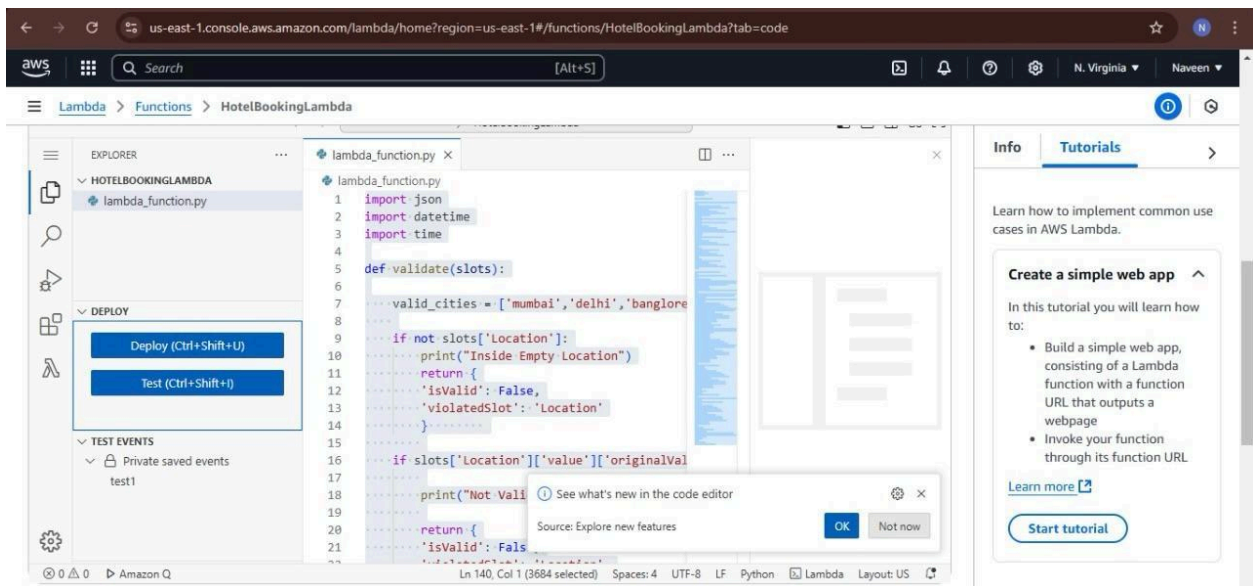


Fig 4.6 Deploying Code in AWS lambda function

After creation, other AWS services may be attached to your Lambda function for integration, including Amazon Lex for chatbots or APIs with Amazon API Gateway. You can test your Lambda function using the testing tools built into the Lambda console or by triggering it via an event from another AWS service. Once tested to your satisfaction, you can deploy the Lambda function in a production environment, where it will execute in response to events such as user interactions, database updates, or file uploads, scaling automatically based on demand. AWS

Lambda is a serverless product, therefore avoiding the management of infrastructure for you, and paying only for the computation time consumed during the execution of a function.

4.5 cloud watch for monitoring

Amazon CloudWatch is a must-monitoring tool for AWS Lambda functions. It gives detailed information about the performance and health of your serverless applications. As shown in Fig 4.7 with automatic log collection and metrics, you can track the execution of Lambda functions in real-time. When you turn on logging for your Lambda functions, CloudWatch Logs captures all output, such as console logs, error messages, and execution details, to enable you to troubleshoot and debug effectively. Beyond that, CloudWatch provides you with key metrics such as invocation count, execution duration, error rate, and throttles so you can get a clear view of how your Lambda functions are performing.

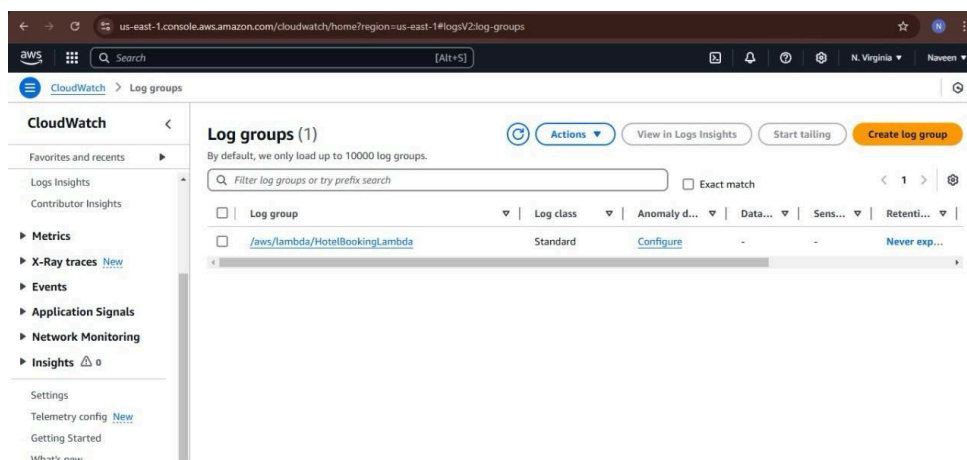


Fig 4.7 Cloud watch Log groups

For example, you can see how frequently your function is invoked, the time it takes to complete requests, and error occurrences. You can extend that by creating alarms to send you notifications when particular thresholds have been crossed, for instance an error rate significantly above a normal rate or even high execution time. These alarms can trigger notifications through Amazon SNS or invoke other AWS services to take corrective actions. You can also create custom visualizations with CloudWatch Dashboards as shown in Fig 4.8 to monitor multiple Lambda functions and their associated metrics in one place, helping you stay on top of your serverless application's performance and ensure issues are addressed proactively.

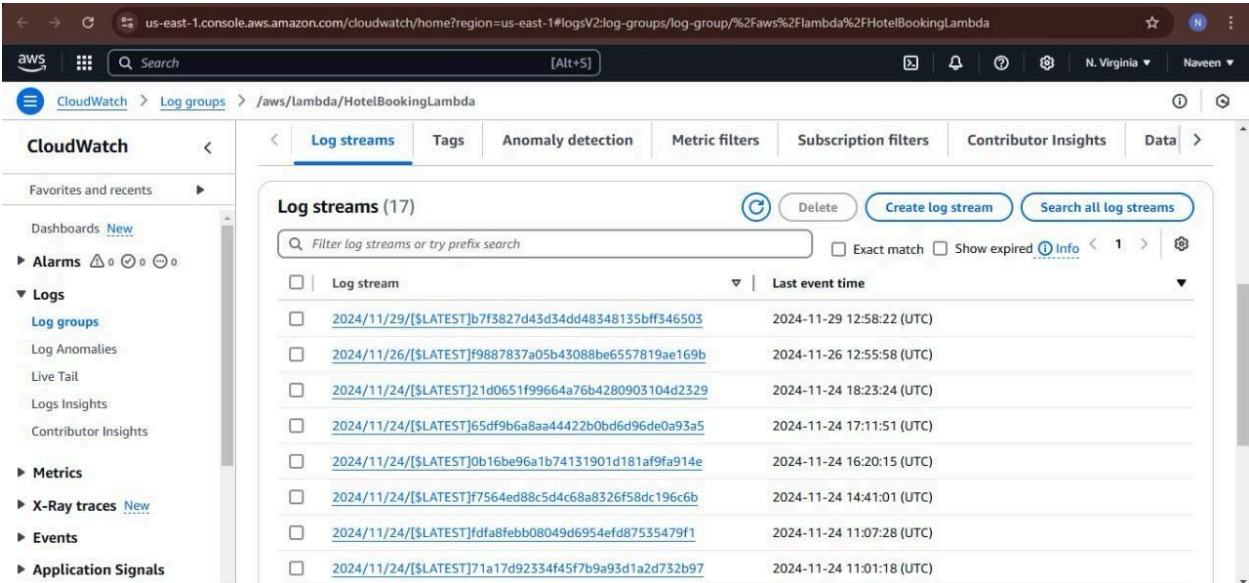


Fig 4.8 Cloud watch log streams

Metrics on Amazon CloudWatch will be of great importance when it comes to tracking the performance and operational efficiency of a customer service chatbot as shown in Fig 4.9. Such metrics will give real-time data that will enable developers and administrators to monitor the health, usage, and responsiveness of the system. Some of the critical metrics for a customer service chatbot include:

4.5.1 Invocation Metrics. Metrics to be tracked include chat instances to which the chatbot is invited. Through this, over time from the provided services, learning will occur in terms of user engagement level and continually demanded services.

4.5.2 Latency metrics: are the measure of how much time the chatbot needs to process and respond to queries raised by a user. The existence of such high latency suggests a problem in performance, usually as a bottleneck in the system, or sometimes an NLP engine and database queries.

4.5.3 Error Metrics: Most organizations need to log errors regarding API calls that failed to proceed, incorrect intent recognition, or connection problems with the database. The closer monitoring of such metrics will enable organisations to quickly identify potential operational issues and act accordingly to counter those problems in a much more prompt manner.

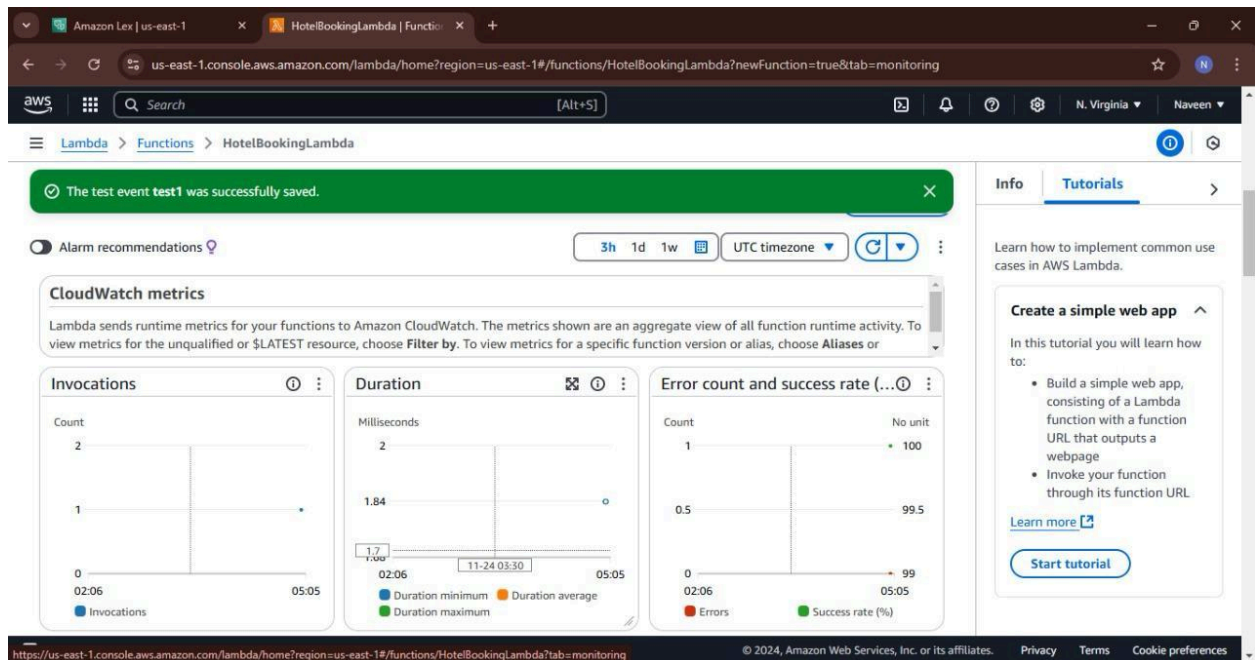


Fig 4.9 Cloud watch Metrics

The metric of throughput should be considered as the number of interactions or conversations it would complete within a specified period. It allows it to monitor continuously, therefore having the ability to assess peak load handling and overall capacity at surge periods. Custom metrics: A given number of developers would be able to provide specific custom metrics for specific purposes through the development of a given use case in the creation of a chatbot so as to track whether more people booked successfully or lost or scores of satisfaction drawn out of feedback made.

4.6 Connection of Lex and lambda with alias:

Connecting Amazon Lex and AWS Lambda using aliases allows for a more controlled and flexible deployment of your Lambda functions in a chatbot environment. An alias in AWS Lambda is essentially a pointer to a specific version of a Lambda function, enabling you to manage different versions of your function seamlessly. When integrating Lex to Lambda, you can use the Lex configuration to invoke a specific version or alias of a Lambda function instead of invoking the latest version directly. This way, you can make new versions of your function appear in a controlled environment without making them live. As an example, you can have

aliases like `dev` for development versions of the Lambda function and `prod` for production versions. When Amazon Lex interacts with the chatbot, it invokes the function through the alias, ensuring that the correct version is used based on the deployment stage.

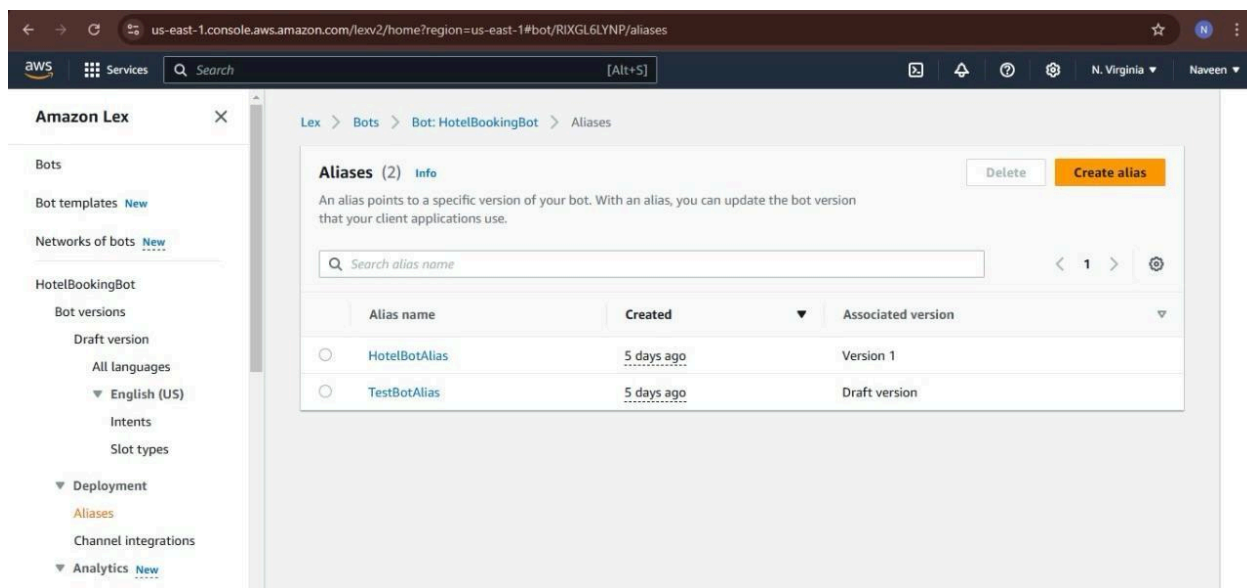


Fig 4.10 Creation of Alias for Lex and Lambda

This aliasing feature ensures that you can roll out changes incrementally, perform version testing without affecting users, and easily switch between different Lambda versions as required, all while maintaining stability and minimizing downtime for your Lex-powered chatbot. By managing Lambda aliases with Lex, you can have better version control and ensure that your Lambda functions are invoked in the right context, offering greater flexibility in handling.

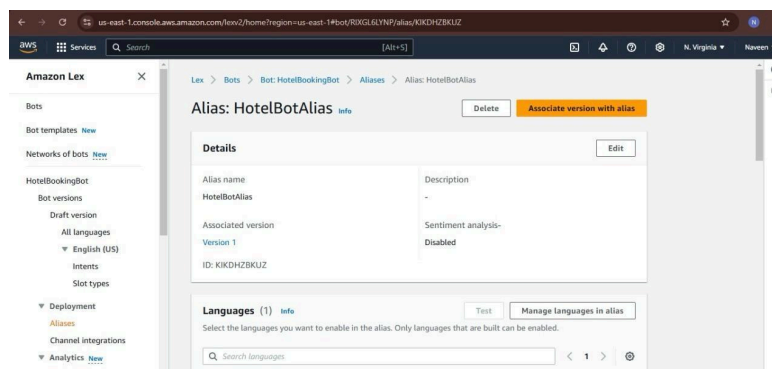


Fig 4.11 Hotel Bot Alias

4.7 Creating twilio for conversation with customers on WhatsApp

This is how businesses can integrate Twilio for conversations with customers on WhatsApp. Creating a Twilio integration allows businesses to interact with users on one of the most widely used messaging platforms seamlessly. To get started, you first need to set up a Twilio account and enable the WhatsApp API through Twilio's console. Once you get a Twilio phone number that can send WhatsApp messages, you'll be able to use the Twilio API to receive and send messages from the WhatsApp users. You'll configure a webhook in the Twilio console, and it will forward incoming messages to your server or backend system, where you will process the messages and reply in real-time as shown in Fig 4.12. This setup can then be integrated with a chatbot powered by platforms such as Amazon Lex or your custom backend logic in AWS Lambda, allowing you to automate responses or handle very specific queries (e.g., order status, customer support). This creates interactive conversational flow with the customers on WhatsApp, managing message delivery and routing as Twilio does, yet your backend processes the actual logic for handling conversations. You can even provide a scalable, real-time customer support solution through the features of Twilio, such as message templates and automated responses, enhancing user engagement and overall customer experience.

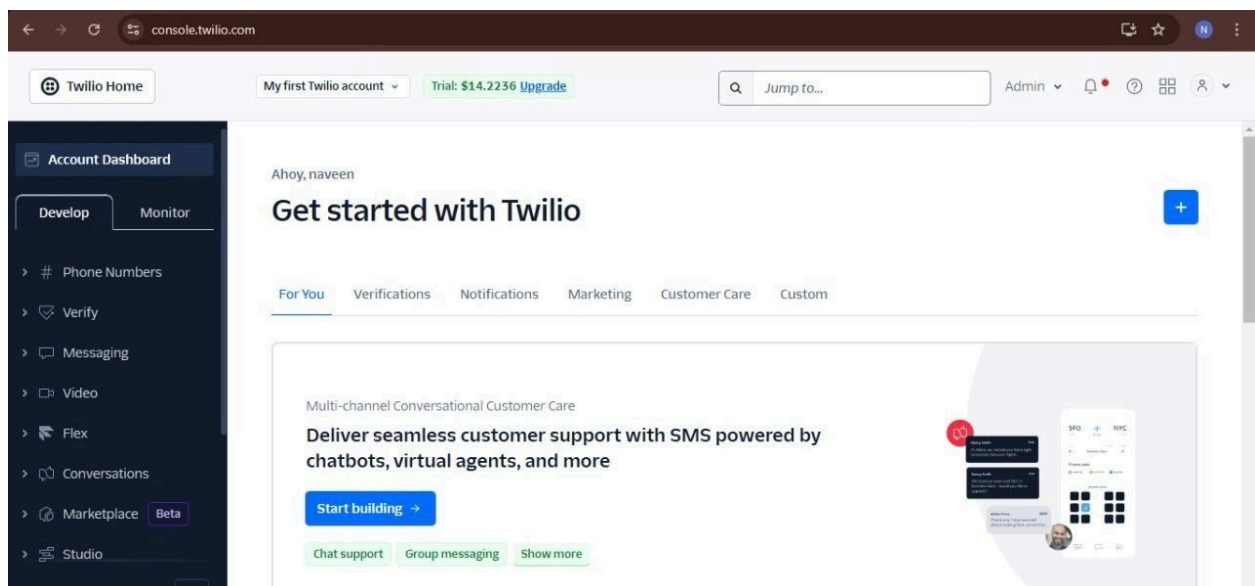


Fig 4.12 Creating Twilio Account

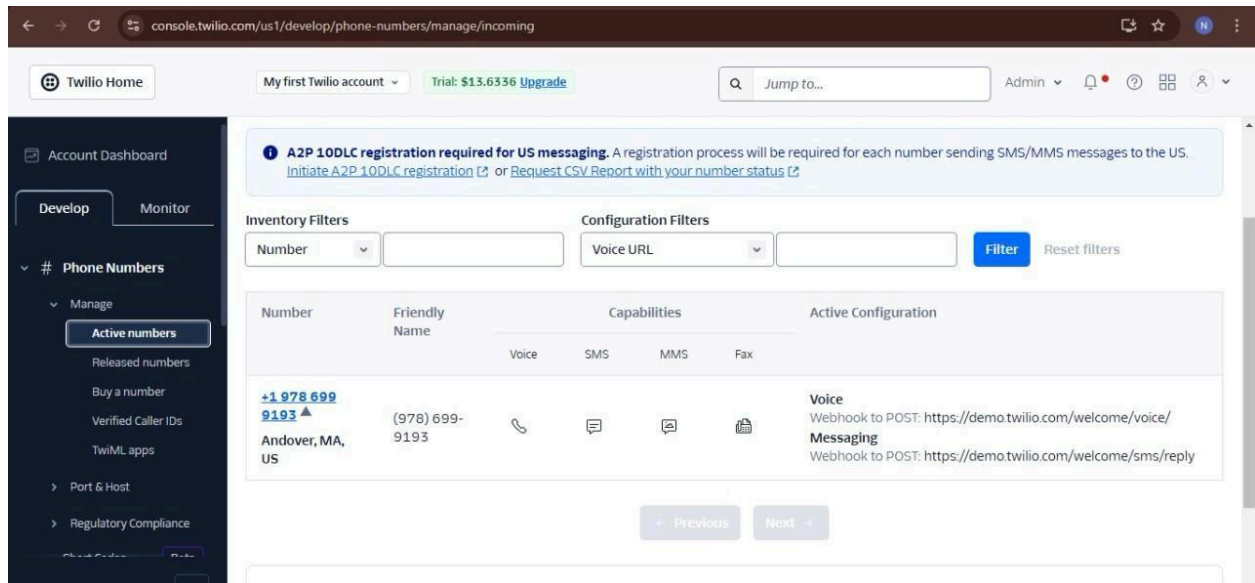


Fig 4.13 Assigning Number for Twilio

4.8 Connecting twilio with aws:

Connecting Twilio with AWS enables businesses to utilize the scalable cloud services of AWS along with Twilio's communication capabilities to engage customers better. This integration is usually set up using Twilio's APIs for messaging, voice, or video services, while AWS provides a robust backend infrastructure for processing and storing data. This is initiated with configurations of Twilio's services, for instance, SMS and voice calls, or even through WhatsApp messaging. All this will forward incoming requests directly to the AWS resources, such as triggering an AWS Lambda function based on the invocation of a webhook triggered in response to an incoming customer message that then processes that particular message. This can also be integrated with Amazon Lex to enable conversational AI for chatbots or Amazon S3 for storing media files. Moreover, by using AWS services like Amazon SNS for notifications or Amazon DynamoDB for data storage, communication is both dynamic and scalable. As shown in Fig.4.14 by connecting Twilio to AWS, businesses can automate communications with customers, handle any volume of messages, and leverage powerful cloud-based analytics tools to make customer services and support even better. This combination of Twilio's messaging infrastructure with AWS' cloud computing power creates seamless, cost-effective solutions for communication needs in modern times.

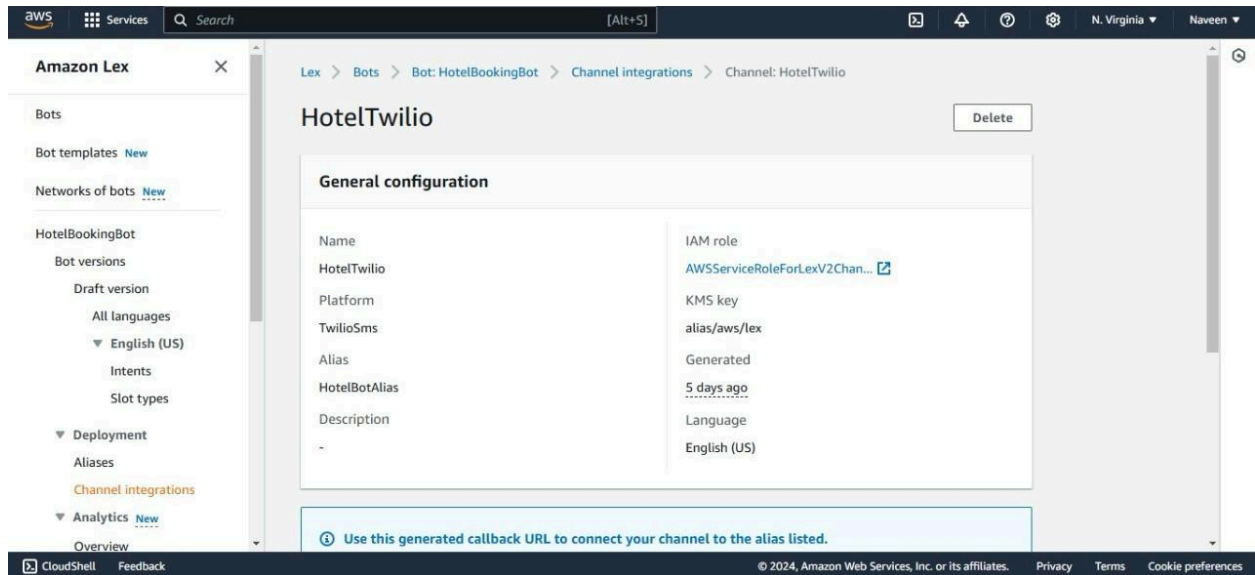


Fig 4.14 Channel Integration for Twilio and AWS

After connecting Twilio with AWS, businesses can create powerful, scalable communication solutions that automate and streamline customer interactions. With this integration, AWS services like Lambda, API Gateway, and S3 work in tandem with Twilio's messaging and communication capabilities. For example, if a customer sends a message via WhatsApp or SMS to a Twilio number, Twilio can trigger an AWS Lambda function to process the incoming message, analyze its content, or even call external APIs.

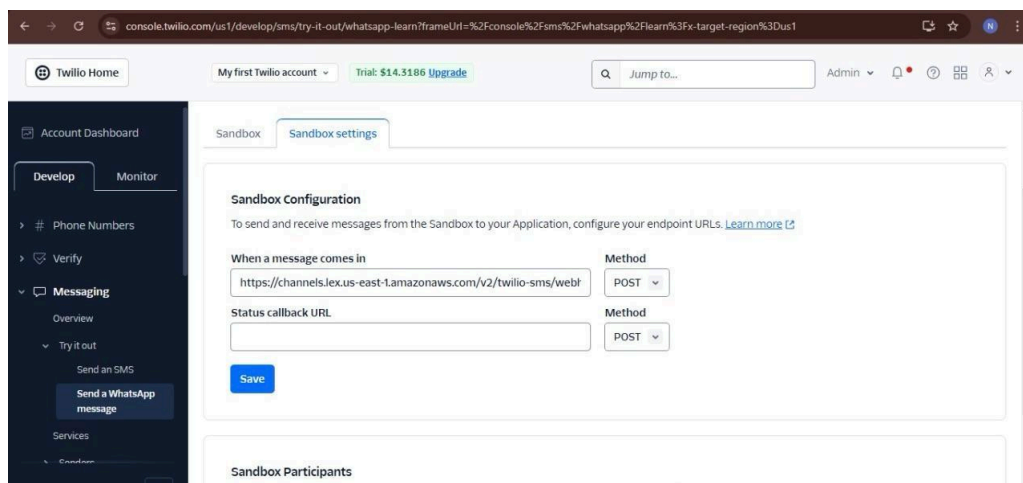


Fig 4.15 Sandbox configuration

The Lambda function can then respond to the user with dynamic content or trigger additional actions, such as updating a database in Amazon DynamoDB or sending a notification through Amazon SNS as shown in Fig.4.15 . If your system uses Amazon Lex, then the lambda function will manage the flow for a chatbot so that end users will be receiving automatic responses dependent on the query. And the entire system can also be easily scaled to allow for more traffic, or businesses can safely store log files, user data or media files in Amazon S3. By integrating Twilio with AWS, companies can deliver personalized, real-time customer experiences, while also leveraging AWS's cloud infrastructure to manage and process the vast amounts of data generated by customer interactions. This seamless connection allows for cost-effective, efficient, and automated customer support, marketing campaigns, and notifications across multiple communication channels.

CHAPTER 5

OUTPUT

The output of connecting Twilio with AWS can be a fully automated, scalable, and efficient communication system that handles a variety of customer interactions. In Fig 5.1 This integration allows businesses to send and receive messages across multiple channels, such as SMS, WhatsApp, or voice calls, while leveraging AWS's cloud services for processing and data management. For example, when a customer sends a message to a Twilio phone number, Twilio triggers an AWS Lambda function to process the message. The Lambda function can analyze the message content, query a database (using Amazon DynamoDB), or call an API to retrieve relevant information. The system can respond to the customer with dynamic, personalized replies, like order status or support information

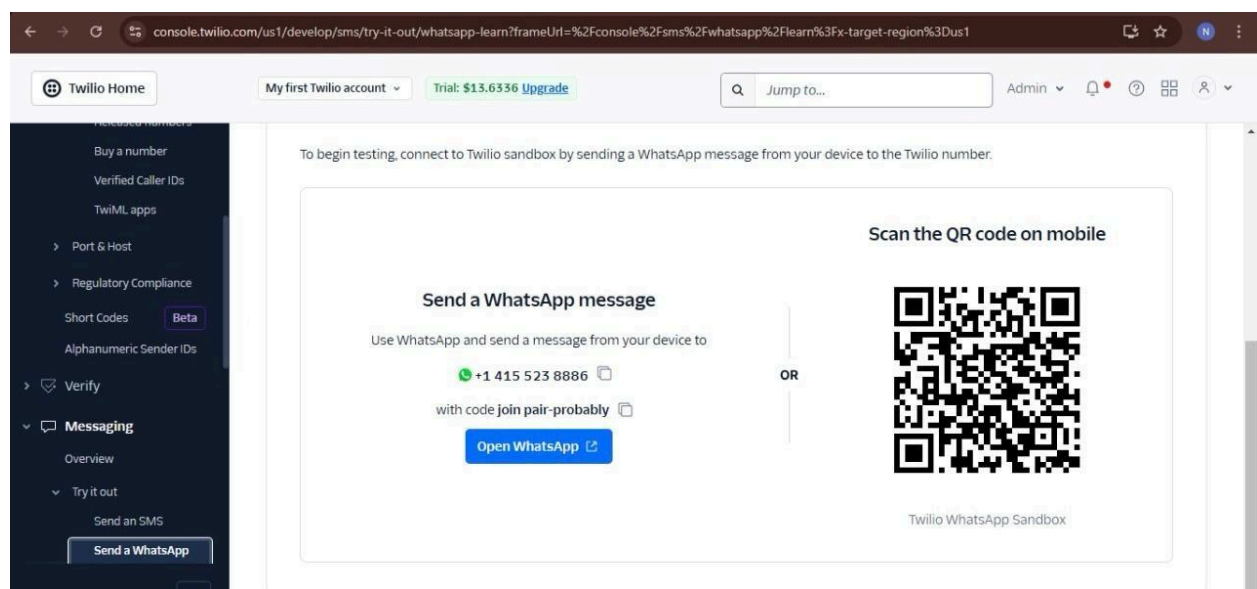


Fig 5.1 Connecting Twilio Account for Whatsapp

Additionally, if the integration uses Amazon Lex for conversational AI, the system can carry out intelligent conversations, guiding customers through predefined flows or handling specific requests, such as booking an appointment or resolving a query. Logs, user data, and media files generated from these interactions can be stored securely in Amazon S3, ensuring that businesses have easy access to historical data for analysis and optimization. Ultimately, the output of connecting Twilio with AWS is a robust, real-time communication platform that can scale with

business needs, automate routine tasks, and provide seamless, personalized customer support while reducing the need for manual intervention. The most recent metrics of CloudWatch from the customer service chatbot hosted by Amazon Lex and Lambda indicates that the system is stable because it's having a steady number of user interactions and requests. In Fig 5.2 the graph shows an increasing request count in the chart, showing increasing user engagement over time, while the latency graph has minimal response times to provide smooth customers. Error rates are very low, which points out system reliability with effective error handling. Invocation metrics demonstrate an equally spread pattern over various Lambda functions, implying a good workload management. Besides this, the memory utilization graph along with concurrency graph shows the correct usage of resources to have cost-effective limits and maintain the high availability of the chatbot.

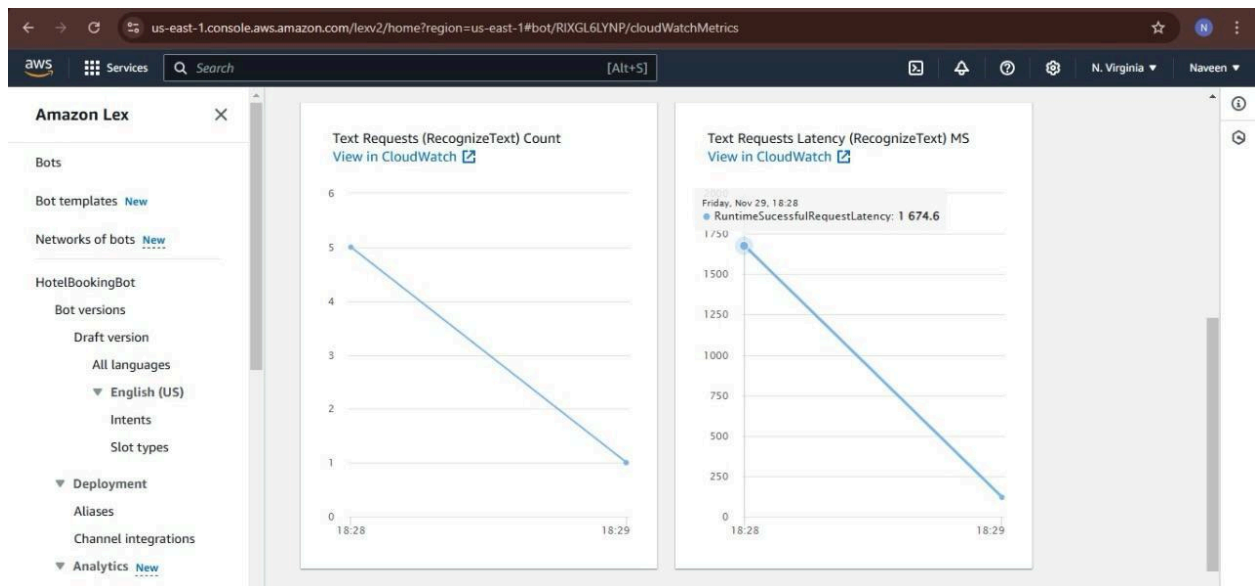
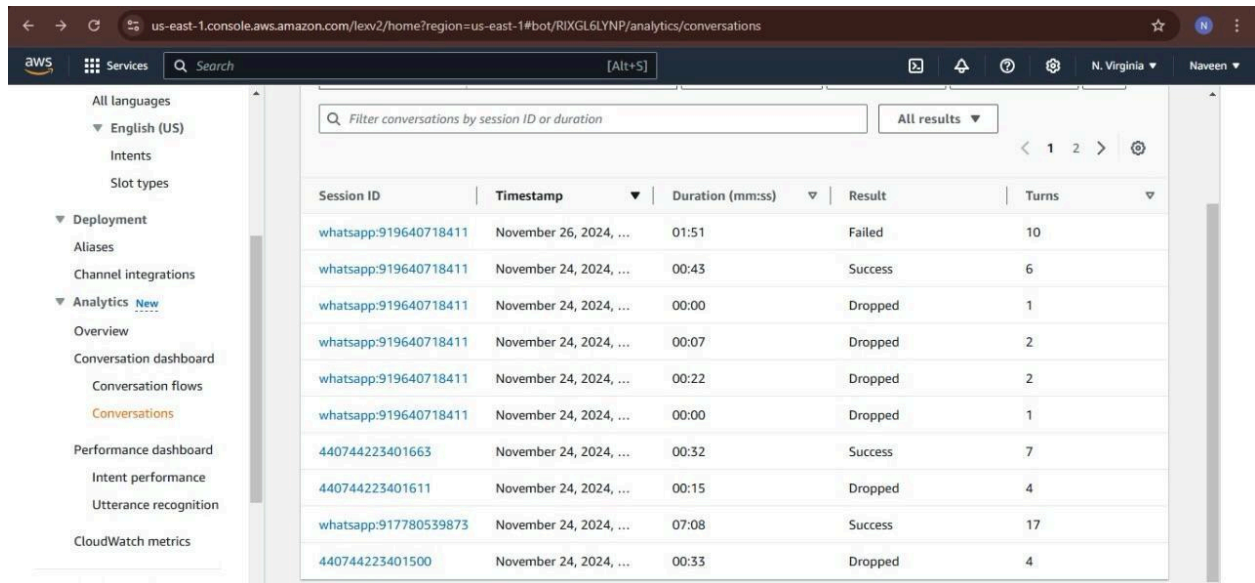


Fig 5.2 Runtime Success Latency

To create an output that effectively represents your customer service chatbot using Amazon Lex and AWS Lambda, we can break it down into two parts:

Conversational Flows: A description of how the chatbot will interact with users, handle intents, and execute corresponding Lambda functions as shown in Fig 5.3.



Session ID	Timestamp	Duration (mm:ss)	Result	Turns
whatsapp:919640718411	November 26, 2024, ...	01:51	Failed	10
whatsapp:919640718411	November 24, 2024, ...	00:43	Success	6
whatsapp:919640718411	November 24, 2024, ...	00:00	Dropped	1
whatsapp:919640718411	November 24, 2024, ...	00:07	Dropped	2
whatsapp:919640718411	November 24, 2024, ...	00:22	Dropped	2
whatsapp:919640718411	November 24, 2024, ...	00:00	Dropped	1
440744223401663	November 24, 2024, ...	00:32	Success	7
440744223401611	November 24, 2024, ...	00:15	Dropped	4
whatsapp:917780539873	November 24, 2024, ...	07:08	Success	17
440744223401500	November 24, 2024, ...	00:33	Dropped	4

Fig 5.3 Conversation Flows

Visual Representation: provided images in the explanation or generate new ones if needed for fall back intents and success rate , dropped rate and Failure rates as shown in Fig 5.4and Fig 5.5.

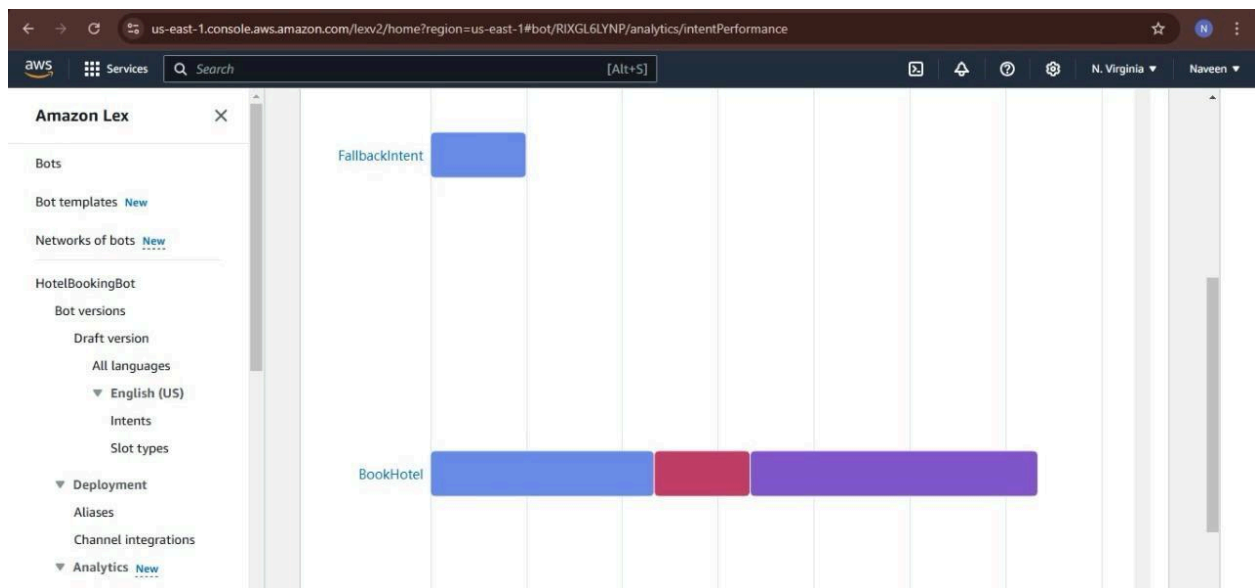


Fig 5.4 Fall back intents and Booking Hotel intents

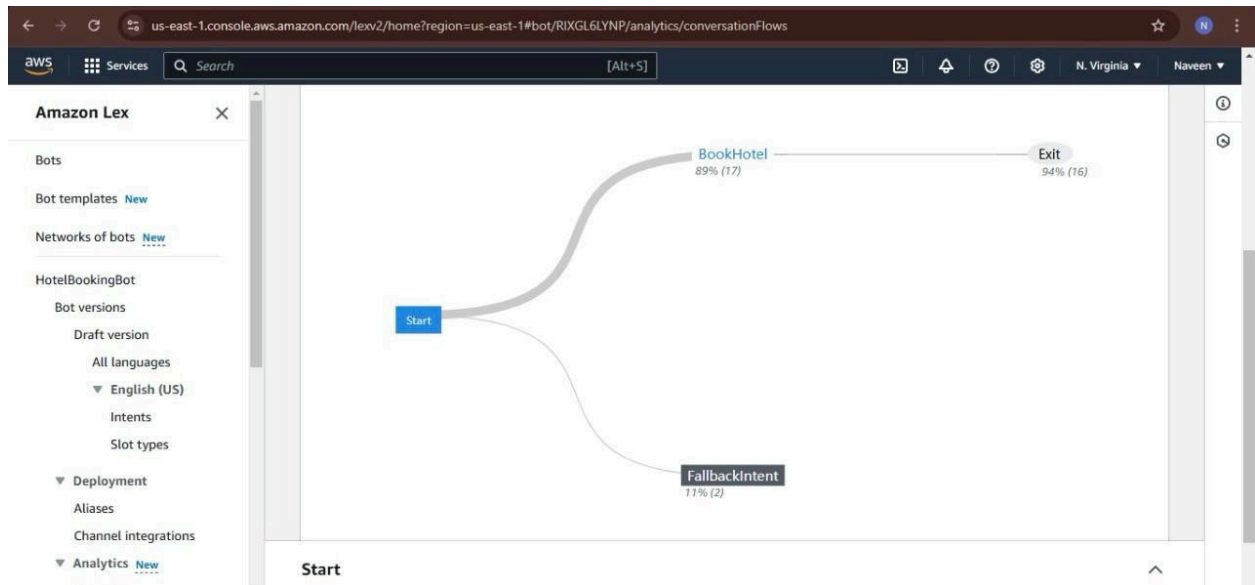


Fig 5.5 Visualisation graph of Intents

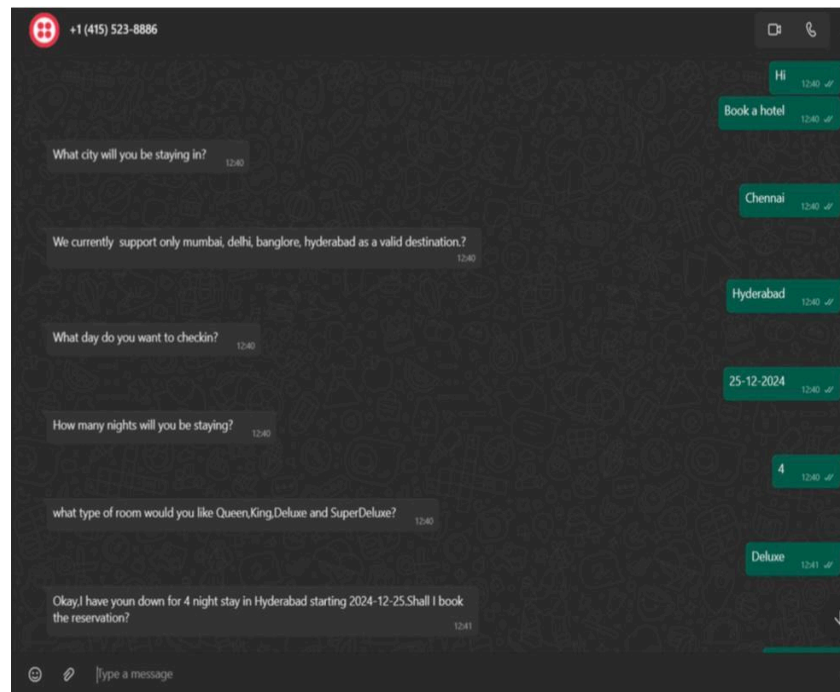


Fig 5.6 Conversation of Twilio with User

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

This is a customer-based chatbot powered by Amazon Lex and AWS Lambda, designed to deliver intelligent, scalable, and efficient customer support. The **natural language processing (NLP)** functionality of Amazon Lex allows the chatbot to automatically comprehend and respond to the questionnaires of customers in real-time, mimicking the manner in which a human interacts. AWS Lambda auto-completes the back-end operations and ensures dynamic, customized responses with minimal human intervention. This integration makes it possible for the chatbot to communicate with external systems like databases or third-party APIs seamlessly so that real-time data can be fetched and processed.

Amazon Lex and AWS Lambda combination greatly enhance customer experience through quick, accurate responses to most routine queries, reducing the need for human involvement in everyday interactions. In addition, Amazon Lex **scalability** means that when the number of users grows, so does the efficiency of the system. Therefore, it is available to businesses of all kinds. Serverless architecture costs the business only for use; this approach is the most cost-effective. Elimination of infrastructure management minimizes overhead costs and streamline operations. This customer-based chatbot with Amazon Lex and AWS Lambda provides a powerful solution for automating customer support. Businesses can use it as an economical and scalable method for addressing queries, speeding up response times, and ensuring a better customer experience.

6.1 Future Work

Customer service chatbots will be even more focused on improving personalization and context awareness, meaning that much more customized and tailored user experiences can be provided by them. The chatbots become better informed about various preferences of a user, histories of a user, or even specific behavioral patterns using advanced natural language processing techniques in conjunction with very robust machine learning models.

For example, take the booking hotel chatbot: it would be able to give much more personalized suggestions based on user bookings done previously, his preferred amenities, or particular

location choices, thereby increasing the pertinence of interaction. That's when integrating the underlying structure of the chatbot architecture and with sentiment analysis would greatly help in the response adjustments from its emotional states. It, therefore encourages empathetic responses not to mention those that are high in humanness, therefore raising user experience levels as a whole.

Yet another important feature with tremendous scope for further growth is the multimodal communication capability. This will develop the ability of bots to interact with users in a chat not only in text but through voice and visual interfaces in an integrated manner. With this holistic view, customer service bots can handle more complex and involved applications. They can decode voluminous bills with accompanying statistics or help users navigating inter-active maps of hotels or resorts to make their engagement more interesting.

Furthermore, ongoing development in speech-to-text as well as text-to-image applications will continue to improve and advance further to further enhance and enrich the richness and effectiveness of these capabilities of making interactions more vibrant and powerful. Beyond their potential functionalities, chatbots will be effortlessly integrated into any possible wearable devices, novel AR platforms, or high-end VR applications to deliver a much more enormous and immersive customer service that might lead to improved customer engagement and satisfaction.

As chatbots continue to evolve and become more complex and sophisticated, so does the demand for deploying explainable artificial intelligence - in short, XAI - that helps to build transparency and trust with end users. Perhaps the focus of research and development would be on having such chatbots explain their logic and reasoning behind giving or making certain recommendations or even decisions.

For instance, a hotel chatbot could explain why it suggested certain room types or tier prices to a potential guest so as to give better comprehension of the basis behind those suggestions. All this above will pay to be a breakthrough as significant in implementing the principles of ethical AI concerning this matter of data privacy and security and bias, especially about that as related, such as health and hospitality. Innovations in how they would be dealing, encrypting, and anonymizing inside these lines, compatible or matching that to HIPAA, are the keys going to pave the proper sense and logic for establishing routes for chatbots.

REFERENCE

- [1] Hu, Yuxin, with Yongqiang Sun, wrote an article entitled "Understanding the impact of internal and external anthropomorphic cues of customer service bot" vol. 7, no. 3, 2023, under article identifier 100047.
- [2] Pawlik, Łukasz, et al. "A method for improving bot effectiveness by recognising implicit customer intent in contact centre conversations." 143 (2022): 33-45.
- [3] Chakrabarti, Chayan, and George F. Luger. "Architecture, algorithms, and evaluation metrics for artificial conversations in user service bots." 42.20 (2015): 6878-6897.
- [4] Chattaraman, Veena, et al. "Smart" Choice? Evaluating AI-powered mobile decision bots for decision-making. 183 (2024): 114801.
- [5] Jones, Carol L. Esmark, et al. "Engaging the avatar: Effects of authenticity signals in chat-based service recoveries.." Journal of Business Research 144 (2022): 703-716.
- [6] Ngai, Eric WT, et al. "Smart Knowledge-Based on User Service Chatbot": Electronic Commerce Research and Applications Electronic Commerce Research and Applications 50 (2021): 101098.
- [7] Sreeharsha, A. S. S. K., Sai Mohan Kesapragada, and Sai Pratheek Chalamalasetty. "Building a chatbot using Amazon Lex and integrating it with a chat application." International journal of scientific research in engineering and management 6.04 (2022): 1-6.
- [8] Williams, Sam. Hands-On Design chatty interfaces with Alexa skills and Amazon Lex for personalized conversational and voice user interfaces for your aws echo devices and web platforms. Packt Publishing Ltd, 2018.

- [9] Pakanati, Dhanush, Gourav Thanner, and R. Ravinder Reddy. "Implementing a college chatbot on AWS." (2020).
- [10] Khandagale, Mr HP, and Ms Shraddha Vaibhav Mane. "Develop a voice chatbot for payment, Amazon Lex serviced, based on an Eyowo platform."
- [11] Samuel, Isaac, et al. "Development of a voice chatbot for payment using amazon lex service with eyowo as the payment platform." 2020 International Conference on Decision Aid Sciences and Application (DASA). IEEE, 2020.
- [12] Nsaif, Wassem Saad, et al. Chatbot development: work and assistancs metricsThe Eurasia Proceedings of Science Technology Engineering and Mathematics 27 (2024): 50-62.
- [13] Das, Projit, et al. Designing Chatbot an Intelligent Technique for First COVID-19 Test.Journal of Computer Science Research 4.4 (2022): 26-35.
- [14] Dihingia, Himanta, et al. "Implementation of chatbot in customer service sector with deep neural networks.." 2021 International Conference on Computational Performance Evaluation (ComPE). IEEE, 2021.
- [15] Chauhan, Shambhavi, and Deepak Arora. "Developing of cloud services of voice assistance using AWS." 2023 2nd International Conference on Applied Artificial Intelligence and Computing (ICAAIC). IEEE, 2023.
- [16] Rai, J. H., and P. O. Bagde. "Bulldig bots for frame working “ AIPs conference proceeding. Vol. 3180. No. 1. AIP Publishing, 2024.
- [17]Haugeland, Isabel Kathleen Fornell, et al. Understanding the user experience for bots 161 (2022): 102788.

- [18]Agnihotri, Arpita, and Saurabh Bhattacharya. "Efficiency and effectiveness of bots." International Journal of Information Management 76 (2024): 102679.
- [19]Sierszen, A., and D. Drabek. " Understanding AIML on AWS" INTED 2024 Proceedings. IATED, 2024.
- [20]Chumpitaz Terry, Alessandro, Liliana Yanqui Huarocc, and Daniel Burga-Durango. "Artificial intelligence for technological customer service Cham: Springer Nature Switzerland, 2023.
- [21]Cordero, Jorge, Luis Barba-Guaman, and Franco Guamán." Use of bots for MSME" Applied Computing and Informatics (2022).
- [22]Dash, Mihir, and Suprabha Bakshi. "Study of customer chatbots in hospital industry." International Journal on Customer Relations 7.2 (2019): 27-33

Shobana M

Sai

 Paper 7

 NWC Class

 SRM Institute of Science & Technology

Document Details

Submission ID

trn:oid::1:3108199728

Submission Date

Dec 9, 2024, 3:00 PM GMT+5:30

Download Date

Dec 9, 2024, 3:01 PM GMT+5:30

File Name

research_report_1.pdf

File Size

1.8 MB

31 Pages

6,842 Words

37,780 Characters





0% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.



Filtered from the Report

- Bibliography
- Quoted Text

Match Groups

-  **1 Not Cited or Quoted 0%**
Matches with neither in-text citation nor quotation marks
-  **0 Missing Quotations 0%**
Matches that are still very similar to source material
-  **0 Missing Citation 0%**
Matches that have quotation marks, but no in-text citation
-  **0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 0%  Internet sources
- 0%  Publications
- 0%  Submitted works (Student Papers)

Integrity Flags





0 Integrity Flags for Review

No suspicious text manipulations found.




Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

-  **1 Not Cited or Quoted 0%**
Matches with neither in-text citation nor quotation marks
-  **0 Missing Quotations 0%**
Matches that are still very similar to source material
-  **0 Missing Citation 0%**
Matches that have quotation marks, but no in-text citation
-  **0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 0%  Internet sources
- 0%  Publications
- 0%  Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

-  **Student papers**
- De Montfort University**

0%