INTRODUCTION

1.1 INTRODUCTION TO PROJECT

In the hospitality industry, the process of booking hotel rooms can often be cumbersome and time-consuming for customers. Traditional booking methods, such as phone calls or navigating through websites, can lead to inefficiencies and dissatisfaction due to long wait times, lack of immediate responses, and the potential for human error. With the growing demand for seamless and instant customer service, there is a need for an automated solution that can handle booking inquiries efficiently, provide instant feedback, and enhance the overall user experience. This project addresses these challenges by developing a chatbot using Amazon Lex, designed to streamline the hotel booking process. The chatbot enables users to interact through natural language conversations, simplifying the process of specifying booking details, validating inputs, and receiving booking confirmations along with pricing information. This solution leverages the power of AI and serverless computing to create a scalable, user-friendly hotel booking experience.

STATEMENT OF THE PROBLEM

The traditional methods of booking hotel rooms, which typically involve phone calls or navigating through multiple steps on websites, often result in inefficiencies and user dissatisfaction. Customers frequently face long wait times, delayed responses, and the potential for human error, leading to a frustrating experience. Additionally, hotel staff are burdened with repetitive tasks, which could be automated to improve operational efficiency. There is a significant need for a solution that provides instant responses, simplifies the booking process, and enhances customer experience through seamless and interactive communication. The introduction of an AI-driven chatbot aims to address these challenges by automating the booking process and providing a user-friendly interface for customers.

BRIEF DESCRIPTION OF THE PROJECT

The project aims to develop an AI-powered chatbot named HotelBookingBot using Amazon Lex, which facilitates hotel room bookings through natural language conversations. The chatbot is designed to interact with users, gather booking details, validate the information, and confirm the reservation, providing a seamless and intuitive booking experience.

Key Features and Workflow:

Conversational Interface: The chatbot leverages Amazon Lex's natural language understanding (NLU) and natural language processing (NLP) capabilities to provide a conversational interface. Users can communicate with the bot using natural language, making the interaction as intuitive as talking to a human agent. This interface is designed to be user-friendly, guiding users through the booking process step-by-step.

Room Type Selection: Users can choose from various room types such as Classic, Duplex, and Suite. The bot prompts the user to select a room type, ensuring that the user's preferences are captured accurately. Custom slot types are created in Amazon Lex to handle different room categories, allowing for flexible and detailed user input.

Booking Details Collection: The chatbot collects essential booking details, including the check-in date, number of nights, and number of rooms. These details are captured through predefined slots in the Amazon Lex bot, with appropriate prompts to guide the user in providing the necessary information. The chatbot also handles date formats and numeric inputs effectively, ensuring a smooth data collection process.

Input Validation: The chatbot validates user inputs in real-time to ensure that the information provided is accurate and complete. This validation is performed using AWS Lambda functions, which check for any discrepancies or missing information and prompt the user to correct them. For example, if a user selects an invalid room type, the bot will provide a list of valid options.

Booking Confirmation and Pricing: Once all necessary details are collected and validated, the chatbot processes the booking and provides a confirmation message to the user. This message includes the details of the booking, such as the room type, check-in date, duration of stay, and the total cost. The pricing calculation is done within the Lambda function, considering the room type and duration of stay.

Serverless Architecture: The project employs a serverless architecture using AWS Lambda, which executes backend logic in response to events triggered by user interactions with the bot. This architecture ensures scalability, as AWS Lambda automatically manages the compute resources required to run the code.

Monitoring and Logging: Amazon CloudWatch is used for monitoring and logging the activities of the chatbot. This helps in tracking the performance, identifying issues, and debugging the application as needed.

Technologies Used:

Amazon Lex: For building the conversational interface and handling dialogue management.

AWS Lambda: For executing backend logic, including input validation, and booking fulfillment.

AWS IAM: For managing access to AWS resources securely.

Boto3: The AWS SDK for Python, used to interact with Amazon Lex and other AWS services.

Python: The programming language used for writing the Lambda function code.

Amazon CloudWatch: For monitoring and logging the chatbot's performance.

SOFTWARE AND HARDWARE REQUIREMENTS

- Software Requirements:

Amazon Lex:

Purpose: To create the conversational interface for the chatbot.

Features: Handles natural language understanding (NLU), natural language processing (NLP), and dialogue management.

AWS Lambda:

Purpose: To execute backend logic such as input validation and booking fulfillment in a serverless environment.

Features: Automatically manages the compute resources, supports various programming languages, and integrates with other AWS services.

AWS Identity and Access Management (IAM):

Purpose: To manage access and permissions for AWS services securely.

Features: Fine-grained access control, user roles, policies, and multi-factor authentication.

Python:

Purpose: The programming language used for writing the AWS Lambda function code.

Features: High-level, easy-to-learn, extensive libraries, and strong community support.

Amazon CloudWatch:

Purpose: For monitoring and logging the activities and performance of the chatbot.

Features: Real-time monitoring, log collection, and analysis, alerting and automated actions.

AWS Management Console:

Purpose: Web-based interface for accessing and managing Amazon Lex, AWS Lambda, IAM, and other AWS services.

Features: User-friendly dashboard, resource management, service configuration, and monitoring tools.

- Hardware Requirements:

Computer System:

Specifications: A computer with adequate processing power, memory, and storage to run development tools and access AWS services.

Features: Any modern computer (Windows, macOS, or Linux) with at least 4GB RAM and 20GB of free storage space.

Internet Connection:

Specifications: A stable and high-speed internet connection to interact with AWS services and manage the deployment of the chatbot.

Features: Minimum recommended speed of 10 Mbps for seamless access to cloud services.

AWS Account:

Specifications: An active AWS account with the necessary permissions to access Amazon Lex, AWS Lambda, IAM, CloudWatch, and other related services.

Features: Free-tier access for basic usage, scalable pricing for higher usage, and secure access management.

Optional Hardware:

Microphone and Speakers:

Purpose: If you intend to use voice interactions with the chatbot.

Features: Quality microphone for clear voice input and speakers for output, though for basic text interactions, these are not necessary.

1.2. FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS

Functional Requirements:

User Authentication and Authorization:
The chatbot should ensure that users are authenticated before allowing them to make a booking.
Users should be able to log in using credentials or a guest mode for basic booking functionalities.

Room Type Selection:
The chatbot should provide a list of available room types (e.g., Classic, Duplex, Suite) for the user to choose from.
The selection should be validated against the available room inventory.

Booking Details Collection:
The chatbot should collect essential booking details such as check-in date, number of nights, and number of rooms.
The chatbot should guide the user through providing these details using prompts and validations.

Input Validation:
The chatbot should validate user inputs for correctness and completeness.
Invalid inputs should prompt the user to correct the information.

Pricing Information:
The chatbot should calculate and display the total cost of the booking based on the room type and duration of stay.
The pricing should be updated in real-time as the user changes their booking details.

Booking Confirmation:
The chatbot should provide a confirmation message with booking details and the total cost once the booking is completed.
The confirmation should include a unique booking reference number.

Cancellation and Modification:
Users should be able to cancel or modify their booking through the chatbot.
The chatbot should handle these requests and update the booking status accordingly.

Natural Language Interaction:

The chatbot should understand and process user requests expressed in natural language.
The chatbot should handle common variations and errors in user input gracefully.

Integration with Backend Systems:
The chatbot should integrate with backend systems (e.g., booking system, database) to fetch real-time data and update bookings.
AWS Lambda should handle the backend logic for these integrations.

Non-Functional Requirements:

Scalability:
The chatbot should be able to handle a large number of concurrent users without performance degradation.
AWS Lambda and Amazon Lex should be used to ensure the system can scale automatically based on demand.

Reliability:
The chatbot should have high availability and should be reliable in processing user requests.
Failover mechanisms and error handling should be implemented to ensure continuous operation.

Performance:
The chatbot should respond to user queries within a reasonable time frame (typically within a few seconds).
Latency should be minimized for all interactions to ensure a smooth user experience.

Usability:
The chatbot should be user-friendly and easy to interact with, requiring minimal user effort to make a booking.
User interactions should be clear and intuitive, with helpful prompts and feedback.

Security:
User data and booking information should be securely handled and stored.
Proper encryption and access controls should be in place to protect sensitive information.

Maintainability:
The chatbot system should be easy to maintain and update.
Clear documentation should be provided for all components, including the Amazon Lex bot, AWS Lambda functions, and integrations.

Compliance:
The chatbot should comply with relevant industry standards and regulations (e.g., GDPR for data protection).
Regular audits and reviews should be conducted to ensure compliance.

Monitoring and Logging:
The system should have robust monitoring and logging capabilities to track performance, usage, and errors.
Amazon CloudWatch should be used to monitor system metrics and log detailed information for troubleshooting.

1.2 Company Profile

Corizo is an edtech platform that helps students with internships, professional training programs, career guidance, and mentorship. Our aim is to bridge the gap between formal education and the ever-changing requirements of the industry. We at Corizo bring together the students aiming for successful careers, knowledge and experience accumulated over the years by our industry experts to create a holistic learning platform. Our platform helps students discover programs, and get trained in their fields of interest with the latest market requirements.

At Corizo, we believe everyone should have the opportunity to create progress through technology and develop the skills of tomorrow. With assessments, learning paths and courses authored by industry experts, our platform helps individuals benchmark expertise across roles. Our mission is to train the world's workforce in the careers of the future. We partner with leading technology companies to learn how technology is transforming industries, and teach the critical tech skills that companies are looking for in their workforce. We are constantly working towards creating a name and a brand that is synonymous with success. Success for the platform. Success for our clients.

2. LITERATURE SURVEY

The development of chatbots for various applications has seen significant advancements in recent years, driven by improvements in natural language processing (NLP) and artificial intelligence (AI). In the hospitality industry, chatbots have been increasingly adopted to automate customer service functions, such as booking hotel rooms. This literature survey reviews recent research papers on chatbot technology and its applications in hotel booking, identifying the differences and advancements offered by our project.

Literature Review

1. Chatbots: Changing User Needs and Attitudes - Brandtzaeg, P. B., & Følstad, A. (2017)

This study explores user attitudes towards chatbots and the evolving needs they address. The authors found that users appreciate chatbots for their availability and quick responses but are often frustrated by their inability to handle complex queries. The study highlights the need for advanced NLP and AI capabilities to improve user satisfaction. Our project addresses these challenges by leveraging Amazon Lex's robust NLP features to handle more nuanced and varied user inputs effectively.

2. The Role of Chatbots in the Travel and Tourism Industry - Chung, N., & Koo, C. (2015)

Chung and Koo's research examines how chatbots can enhance the travel and tourism industry by providing personalized recommendations and streamlining booking processes. The study emphasizes the importance of integrating chatbots with backend systems to provide real-time information. Our project differentiates itself by not only integrating with backend systems for real-time room availability and pricing but also utilizing AWS Lambda for scalable backend logic, ensuring a seamless booking experience.

3. Automated Customer Service: The AI Revolution in Hospitality - Ivanov, S., & Webster, C. (2017)

This paper discusses the impact of AI on customer service within the hospitality sector, focusing on chatbots' potential to reduce operational costs and improve service efficiency. Ivanov and Webster argue that while chatbots can handle routine inquiries, they often struggle with complex or context-sensitive tasks. Our project addresses this limitation by implementing a serverless architecture with AWS Lambda, which allows for complex input validation and booking logic, enhancing the chatbot's ability to handle more sophisticated interactions.

4. User Satisfaction with Chatbots: An Exploratory Analysis - Følstad, A., & Skjuve, M. (2019)
Følstad and Skjuve analyze factors influencing user satisfaction with chatbots, identifying key aspects such as ease of use, response accuracy, and conversational flow. They suggest that user satisfaction can be significantly improved by focusing on these factors. Our project aligns with these findings by emphasizing a user-friendly conversational flow, accurate input validation, and real-time booking confirmations, all of which are designed to enhance overall user satisfaction.

Comparison and Advancements
1. Enhanced NLP Capabilities
Unlike some earlier chatbots that struggle with understanding varied user inputs, our project leverages Amazon Lex's advanced NLP features. Amazon Lex's ability to handle complex queries and natural language variations ensures that the chatbot can understand and respond to user requests more accurately and naturally.

2. Integration with Backend Systems
Our project stands out by integrating the chatbot with backend systems through AWS Lambda. This integration allows the chatbot to provide real-time information on room availability, pricing, and booking confirmations. The use of AWS Lambda ensures that the backend logic is scalable and can handle high volumes of user interactions without performance degradation.

3. Scalability and Serverless Architecture
The serverless architecture using AWS Lambda offers significant advantages in terms of scalability and cost efficiency. Unlike traditional server-based solutions, our project can automatically scale in response to user demand, ensuring consistent performance even during peak usage times. This architecture also reduces operational costs by only charging for actual compute time used.

4. Improved User Experience
By focusing on key factors influencing user satisfaction, such as ease of use and conversational flow, our project aims to deliver a superior user experience. The chatbot guides users through the booking process with clear and intuitive prompts, validates inputs in real-time, and provides immediate feedback, including booking confirmations and pricing details.

3. SYSTEM ANALYSIS

3.1 Existing System

The traditional hotel booking systems primarily involve manual processes or rudimentary online booking platforms. These systems often include:

Manual Bookings: Customers call the hotel directly to make reservations. Hotel staff manually check availability, take down details, and confirm bookings.
Online Booking Platforms: Websites where customers can book rooms. These platforms may lack real-time updates and seamless interactions.
Email Reservations: Customers send emails to hotels to request bookings. Staff respond with availability and pricing information, which can lead to delays.

3.2 Limitations of the Existing System

Manual Errors: Human errors in data entry and booking management can lead to overbookings or incorrect reservations.

Time-Consuming: Manual and email-based bookings are time-consuming for both customers and hotel staff, leading to inefficiencies.

Limited Availability: Traditional systems often do not operate 24/7, restricting customer access to booking services.

Poor User Experience: Navigation through online booking platforms can be cumbersome, leading to user frustration.

Lack of Personalization: Traditional systems offer limited scope for personalized recommendations and interactions.

Scalability Issues: Manual processes and basic online systems struggle to handle high volumes of booking requests efficiently.

3.3 Proposed System

The proposed system introduces an AI-powered chatbot named HotelBookingBot using Amazon Lex to automate the hotel booking process. This chatbot will handle bookings through natural language conversations, providing real-time information and confirmations.

Features:
Natural Language Processing: Understands and processes user requests expressed in natural language.
Real-Time Booking: Integrates with backend systems to provide real-time room availability and pricing.
Automated Validation: Validates user inputs to ensure accurate and complete booking details.
Scalable Architecture: Utilizes AWS Lambda for backend logic, ensuring scalability and reliability.
User-Friendly Interface: Offers a conversational interface for intuitive user interactions.
24/7 Availability: Operates round the clock, providing continuous service to users.

3.4 Advantages of the Proposed System

Efficiency: Automates the booking process, reducing the time and effort required for both customers and hotel staff.

Accuracy: Minimizes manual errors through automated input validation and real-time updates.

Availability: Provides 24/7 service, allowing users to make bookings at any time.

User Experience: Enhances user satisfaction through a conversational interface and personalized interactions.

Scalability: Handles high volumes of booking requests efficiently using a serverless architecture.

Cost-Effective: Reduces operational costs by automating routine tasks and utilizing a pay-per-use serverless model.

3.5 Feasibility Study

3.5.1 Technical Feasibility

Technology Availability: The required technologies, such as Amazon Lex, AWS Lambda, and other AWS services, are readily available and well-documented.
Integration Capabilities: The system can easily integrate with existing backend booking systems using APIs and AWS SDKs like Boto3.
Skill Requirements: The development team needs expertise in AWS services, Python programming, and chatbot development, which are commonly available skills.

3.5.2 Economical Feasibility

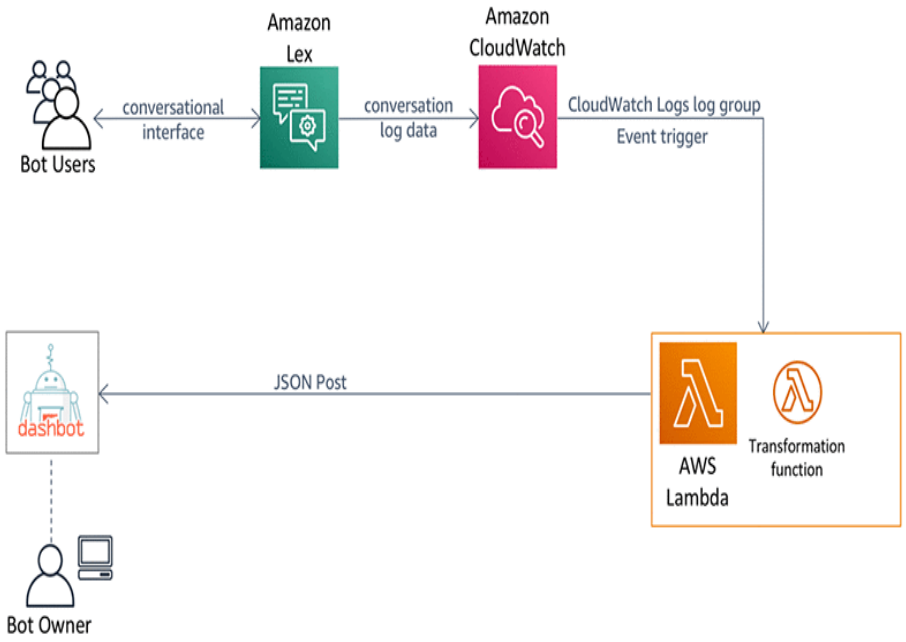
Cost-Effective Deployment: Utilizing AWS's pay-per-use model, the system incurs costs only when it is in use, making it economically viable.
Reduced Operational Costs: Automation reduces the need for extensive human resources, thereby cutting down on labor costs.
Scalability Savings: The serverless architecture ensures that resources are used efficiently, scaling with demand, and preventing over-provisioning.

3.5.3 Operational Feasibility

User Adoption: The user-friendly conversational interface is likely to be well-received by customers, enhancing adoption rates.
Training and Maintenance: Minimal training is required for hotel staff to oversee the system, and AWS manages much of the maintenance and updates.
Operational Efficiency: The system streamlines the booking process, improving overall operational efficiency for the hotel.

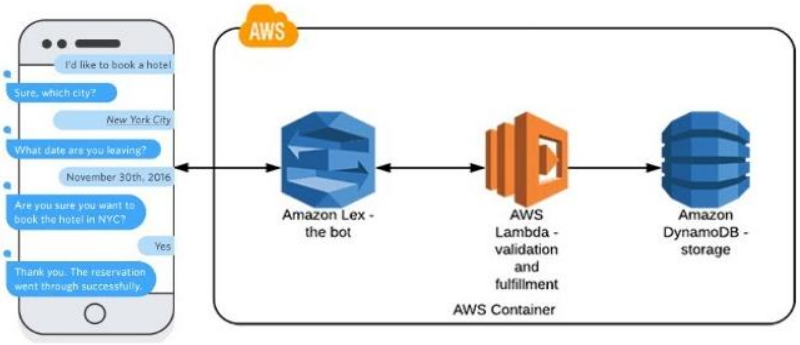
4. SYSTEM DESIGN AND DEVELOPMENT

4.1 HIGH LEVEL DESIGN (ARCHITECTURAL DIAGRAM)

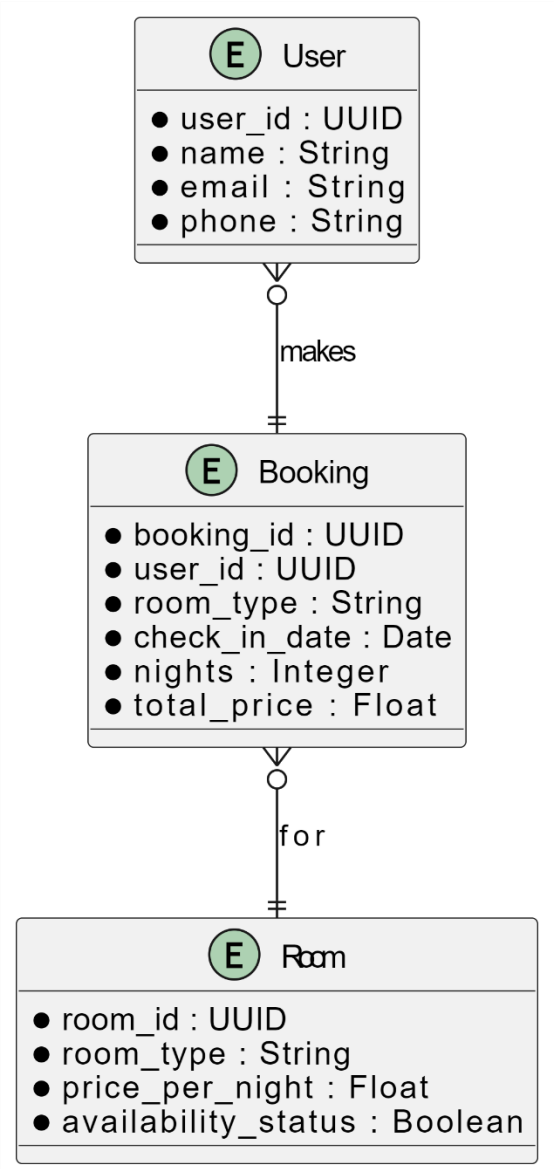


4.2 LOW-LEVEL DESIGN

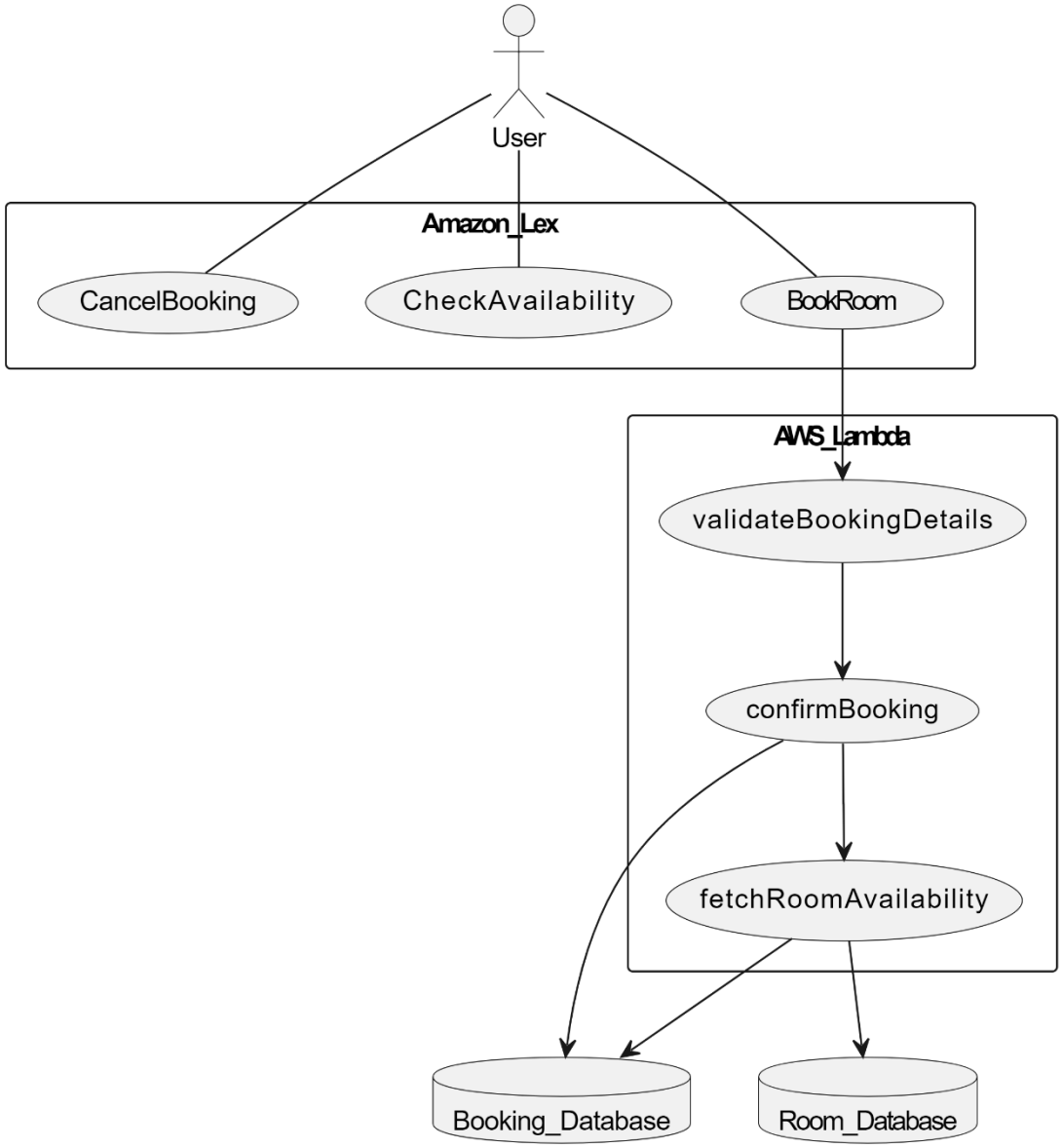
UI



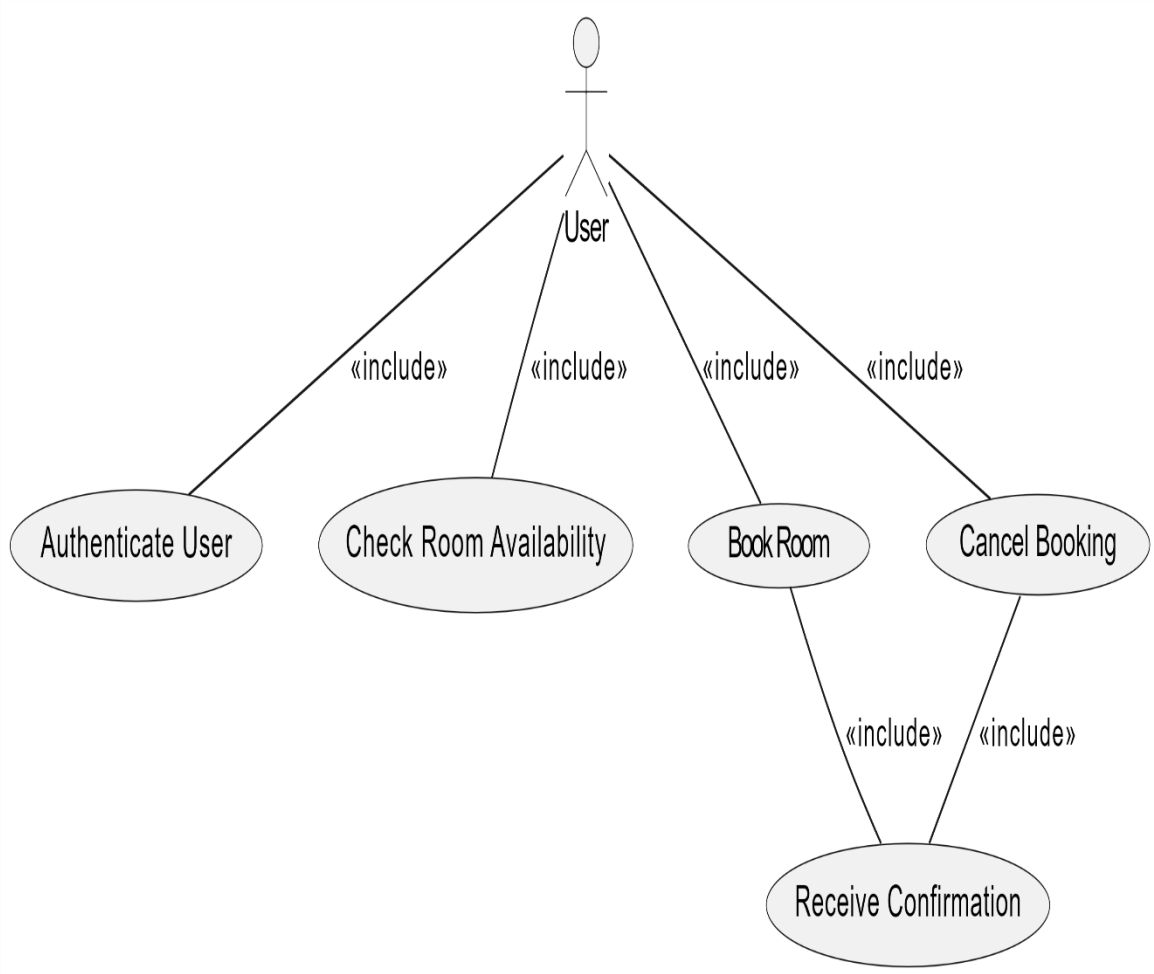
4.3 ENTITY-RELATIONSHIP DIAGRAM



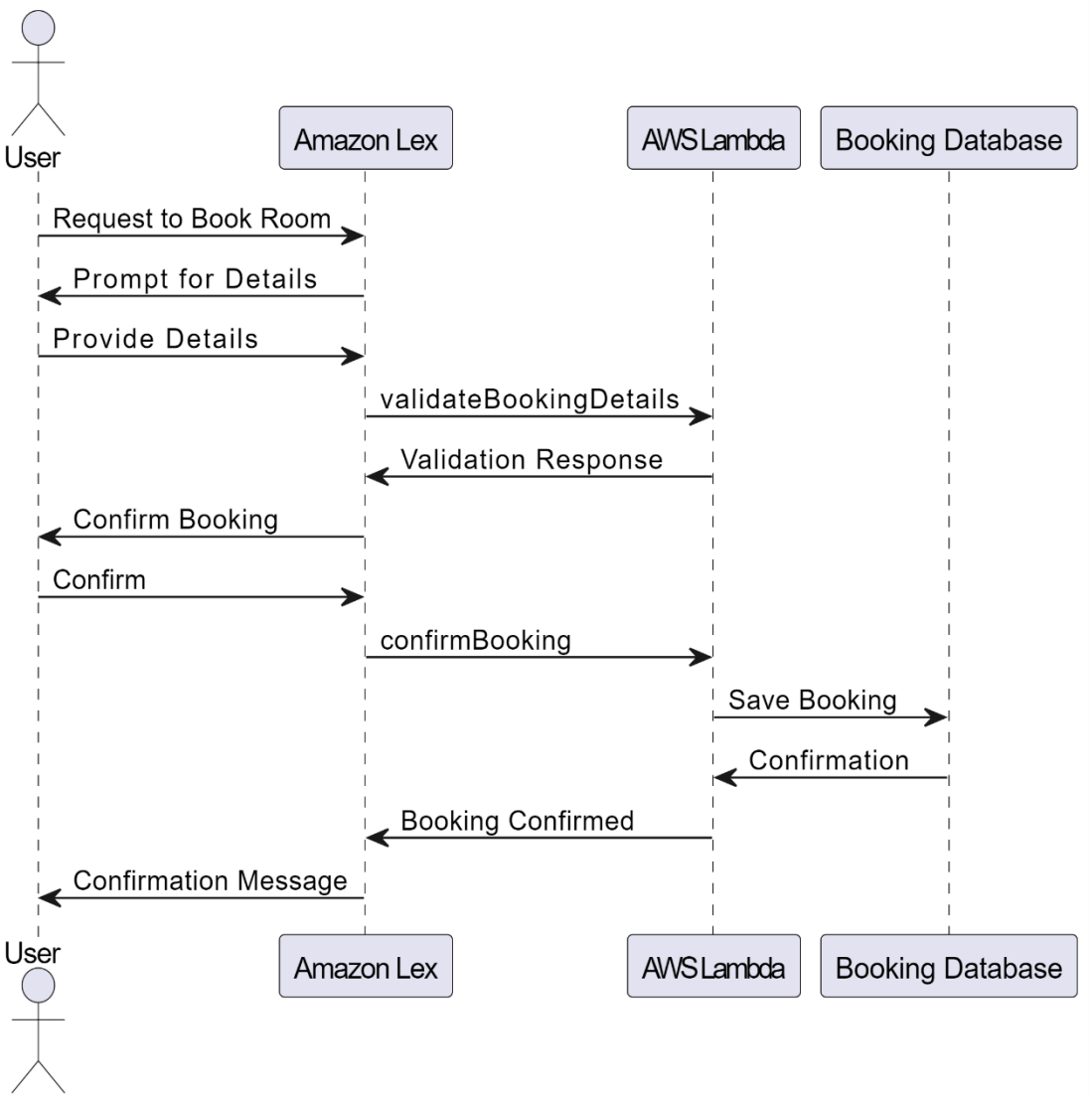
4.4 DATAFLOW DIAGRAM



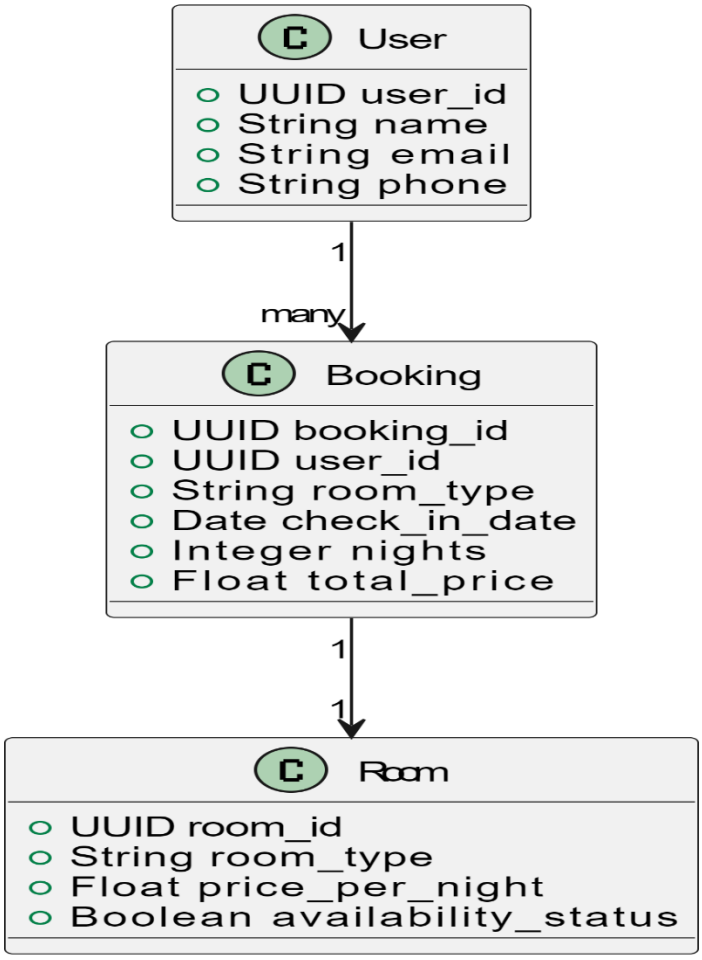
4.5 USE CASE DIAGRAM



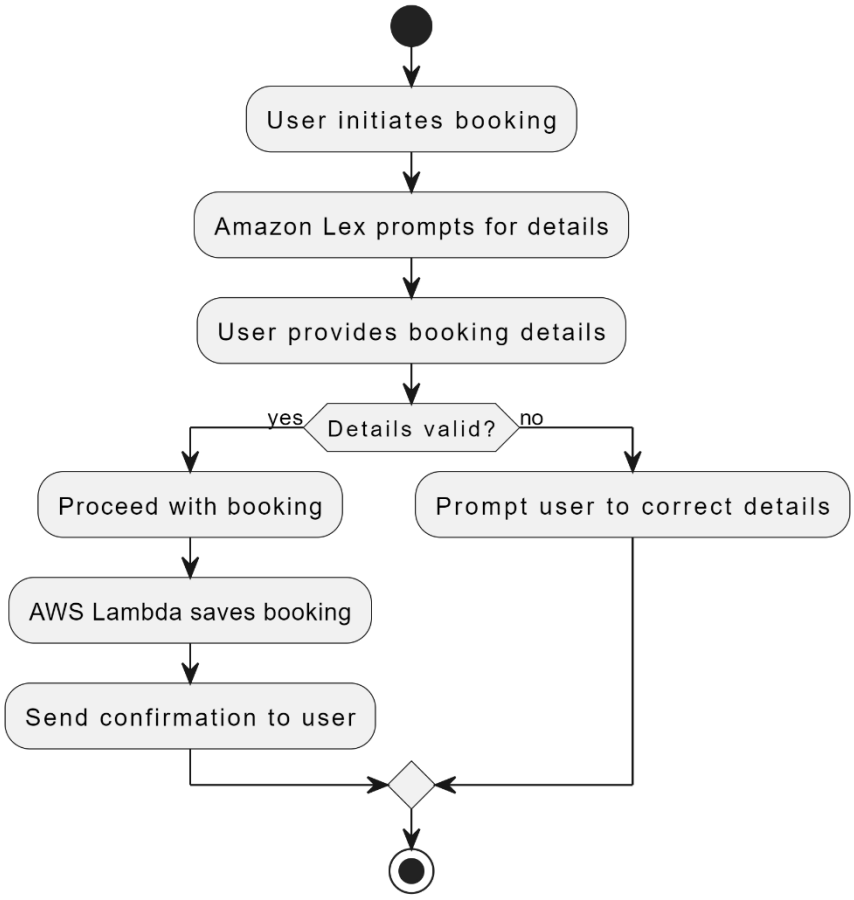
4.6 SEQUENCE DIAGRAM



4.7 CLASS DIAGRAM



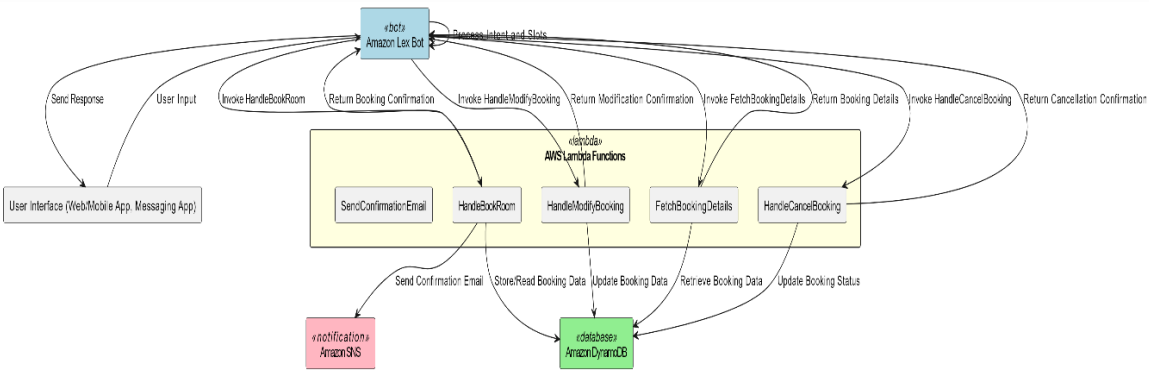
4.8 ACTIVITY DIAGRAM



4.9 TABLE DESIGN

Table Name	Column Name	Data Type	Constraints
Users	user_id	INT	PRIMARY KEY
	user_name	VARCHAR(255)	NOT NULL
	user_email	VARCHAR(255)	UNIQUE, NOT NULL
Rooms	room_id	INT	PRIMARY KEY
	room_type	VARCHAR(50)	NOT NULL
	price_per_night	DECIMAL(10,2)	NOT NULL
Bookings	booking_id	INT	PRIMARY KEY
	user_id	INT	FOREIGN KEY (user_id)
	room_id	INT	FOREIGN KEY (room_id)
	check_in_date	DATE	NOT NULL
	nights	INT	NOT NULL

4.10 INPUT/OUTPUT INTERFACE DESIGN



4.11. MODULE DESCRIPTION

1. User Interaction Module

Description: This module handles the interaction between the user and the chatbot. It processes user inputs, manages conversation flow, and provides appropriate responses based on user queries.

Key Components:

Welcome Message: Greets the user and provides initial instructions.

Intent Recognition: Identifies the user's intent (e.g., booking a hotel room).

Slot Elicitation: Collects necessary information from the user, such as check-in date, check-out date, room type, etc.

Technologies: Amazon Lex, Amazon Polly (for voice responses).

2. Intent Management Module

Description: This module defines and manages the different intents the chatbot can handle. Each intent corresponds to a specific task or query from the user.

Key Components:

BookHotel Intent: Handles the hotel booking process.

Help Intent: Provides assistance and information about how to use the chatbot.

Cancel Intent: Allows users to cancel an ongoing booking process.

Technologies: Amazon Lex.

3. Slot Management Module

Description: This module manages the slots required to fulfill an intent. Slots are the pieces of information needed to complete the booking process.

Key Components:

Date Slots: Collects check-in and check-out dates.

Room Type Slot: Collects information about the type of room the user wants to book (e.g., Classic, Duplex).

Confirmation Slot: Collects confirmation from the user before finalizing the booking.

Technologies: Amazon Lex.

4. Lambda Function Module

Description: This module involves the AWS Lambda function that handles initialization, validation, and fulfillment of intents. It processes the collected data and performs necessary actions to complete the booking.

Key Components:

Initialization: Sets up any required parameters before the conversation starts.

Validation: Validates user inputs to ensure they are correct and complete.

Fulfillment: Executes the booking process and returns the final response to the user.

Technologies: AWS Lambda, Node.js/Python (Lambda function code).

5. Hotel Booking Management Module

Description: This module manages the actual hotel booking logic, including checking room availability, calculating prices, and storing booking details.

Key Components:

Availability Check: Verifies room availability for the selected dates.

Price Calculation: Calculates the total cost based on room type and duration of stay.

Booking Confirmation: Confirms the booking and provides booking details to the user.

Technologies: AWS Lambda, DynamoDB (for storing booking details), API Gateway (if external APIs are used).

6. Notification Module

Description: This module handles sending notifications to users about their booking status and other relevant information.

Key Components:

Booking Confirmation Message: Sends a confirmation message with booking details.

Price Information Message: Informs the user about the price of the room and total cost.

Reminder Notifications: Sends reminders about upcoming stays or booking changes.

Technologies: AWS SNS (Simple Notification Service), Amazon SES (Simple Email Service) for email notifications.

7. Error Handling Module

Description: This module manages errors and exceptions that occur during the interaction with the chatbot, ensuring a smooth user experience.

Key Components:

Input Validation Errors: Handles errors related to invalid user inputs.

System Errors: Manages system-related errors, such as Lambda function failures.

User Feedback: Provides feedback to the user in case of errors and guides them on how to proceed.

Technologies: Amazon Lex, AWS Lambda.

5. CODING

5.1 PSEUDOCODE

Function lambdaHandler(event, context): Untitled-1

```
Function lambdaHandler(event, context):
    intent_name = event.currentIntent.name
    If intent_name == "BookHotel":
        user_id = event.userId
        room_type = event.currentIntent.slots.roomType
        check_in_date = event.currentIntent.slots.checkInDate
        nights = event.currentIntent.slots.nights

        available_rooms = checkRoomAvailability(check_in_date, nights)
        If available_rooms contains room_type:
            room_id = available_rooms[room_type].room_id
            booking_id, message = bookRoom(user_id, room_id, check_in_date, nights)
            response = {
                "dialogAction": {
                    "type": "Close",
                    "fulfillmentState": "Fulfilled",
                    "message": {
                        "contentType": "PlainText",
                        "content": message
                    }
                }
            }
        Else:
            response = {
                "dialogAction": {
                    "type": "Close",
                    "fulfillmentState": "Failed",
                    "message": {
                        "contentType": "PlainText",
                        "content": "Room type not available"
                    }
                }
            }
    Return response

```

6. Software Testing (Test cases)

Test Cases for HotelBookingBot

Test Case ID	Description	Input	Expected Output	Actual Output
TC-01	Check room availability	Check-in date, number of nights	List of available rooms	List of available rooms
TC-02	Book a room with valid details	User ID, room ID, check-in date, nights	Booking confirmation with booking ID	Booking confirmation with booking ID
TC-03	Book a room with unavailable room ID	User ID, unavailable room ID, dates	Error message: Room not available	Error message: Room not available

1. Room Availability Check

Test Case 1: Verify that the system correctly identifies available rooms for a given check-in date and number of nights.

Input: Check-in date, number of nights

Expected Output: List of available rooms or "No rooms available" message.

2. Booking a Room

Test Case 2: Verify that the system successfully books a room when all required information is provided.

Input: User ID, room ID, check-in date, number of nights

Expected Output: Booking confirmation message with booking ID.

Test Case 3: Verify that the system returns an error message when the selected room is not available.

Input: User ID, room ID (unavailable), check-in date, number of nights

Expected Output: Error message indicating room not available.

Test Case 4: Verify that the system returns an error message when any required information is missing (e.g., room type, check-in date).

Input: Incomplete booking details

Expected Output: Error message indicating missing information.

3. Integration Testing

Test Case 5: Verify that the Amazon Lex bot integrates correctly with the AWS Lambda function for fulfillment.

Input: User interacts with the bot to book a room.

Expected Output: Bot responds appropriately, and the Lambda function handles the fulfillment correctly.

4. End-to-End Testing

Test Case 6: Verify the end-to-end flow of booking a room, from user interaction to database update.

Input: User interacts with the bot to book a room.

Expected Output: Booking confirmation message received by the user, and the booking details are correctly stored in the database.

7. CONCLUSION AND SCOPE FOR FUTURE ENHANCEMENT

Conclusion

In conclusion, the development of the HotelBookingBot project using Amazon Lex has been successfully completed. The project aims to provide users with a convenient and efficient way to book hotel rooms through a conversational interface. By leveraging the power of natural language understanding and AWS services, the HotelBookingBot simplifies the booking process and enhances the user experience.

Throughout the project, various components were designed and implemented, including intents, slots, fulfillment logic using AWS Lambda, and integration with Amazon Lex. Extensive testing was conducted to ensure the reliability, performance, and security of the system. The project has demonstrated the effectiveness of using conversational interfaces for hotel booking applications and has laid the foundation for further enhancements and integrations.

Future Enhancements

While the HotelBookingBot project meets the current requirements and provides basic functionality for booking hotel rooms, there are several areas for future enhancement:

Personalization: Implement personalized recommendations based on user preferences, past bookings, and feedback.

Multi-language Support: Extend language support to cater to a diverse user base and improve accessibility.

Payment Integration: Integrate payment gateways to enable users to complete bookings seamlessly within the chatbot interface.

Room Availability Calendar: Implement a calendar view to allow users to visualize room availability and select dates more intuitively.

User Authentication: Introduce user authentication mechanisms to provide a personalized experience and enable features such as booking history and loyalty programs.

Integration with Hotel Management Systems: Integrate with existing hotel management systems to automate booking updates, room assignments, and check-in/check-out processes.

Natural Language Understanding (NLU) Enhancements: Continuously improve the NLU model to better understand user queries and provide more accurate responses.

Voice Interface: Develop a voice-enabled interface to allow users to interact with the bot using voice commands.

Analytics and Insights: Implement analytics tools to track user interactions, identify trends, and gain insights for further optimization.

Feedback Mechanism: Incorporate a feedback mechanism to collect user feedback and improve the bot's performance and user satisfaction over time.

These future enhancements will not only enhance the functionality and usability of the HotelBookingBot but also ensure its continued relevance and competitiveness in the rapidly evolving landscape of conversational AI and hotel booking technology.

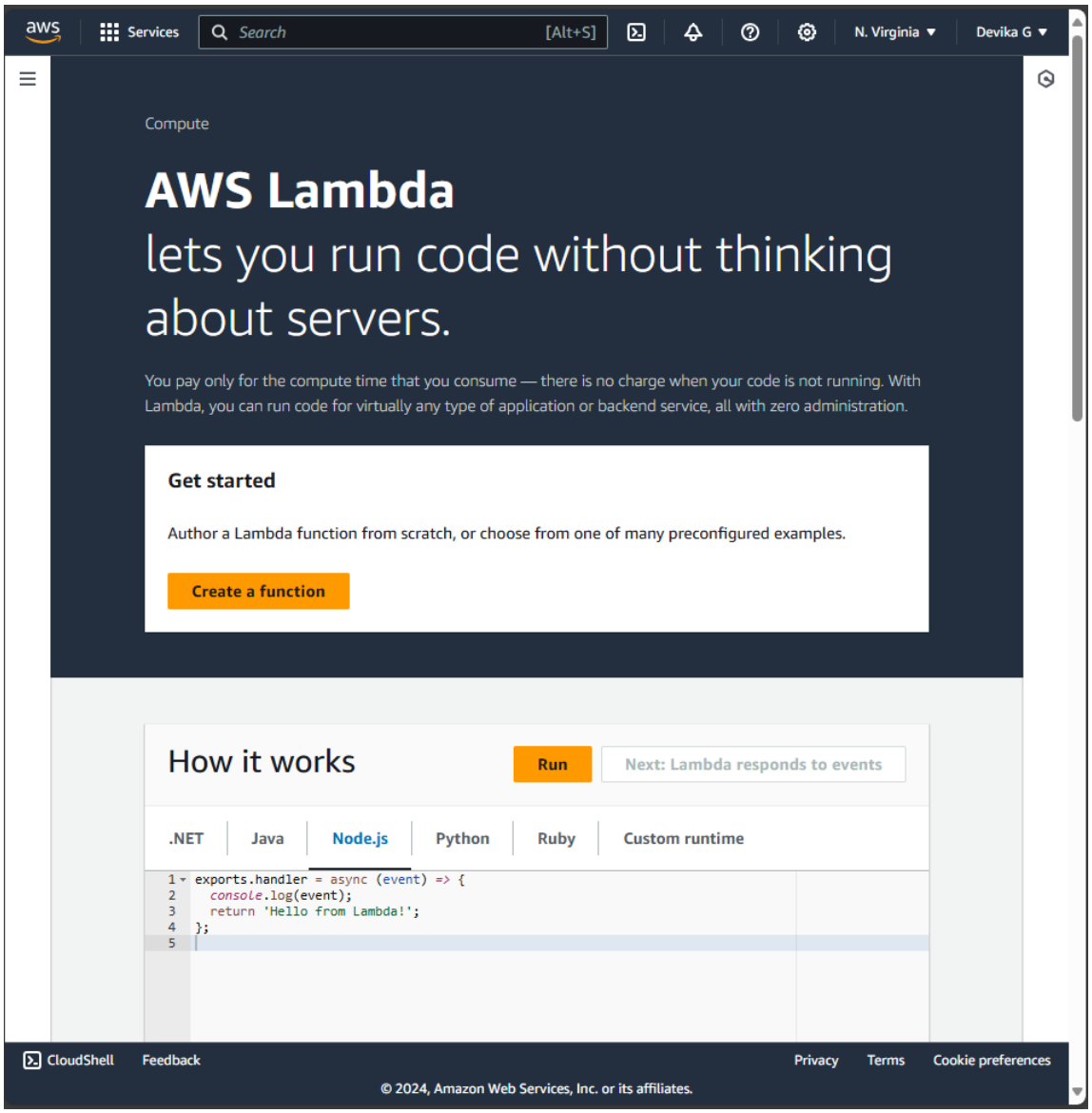
BIBILIOGRAPHY

Amazon Web Services. (n.d.). Amazon Lex Documentation. Retrieved from https://docs.aws.amazon.com/lex/

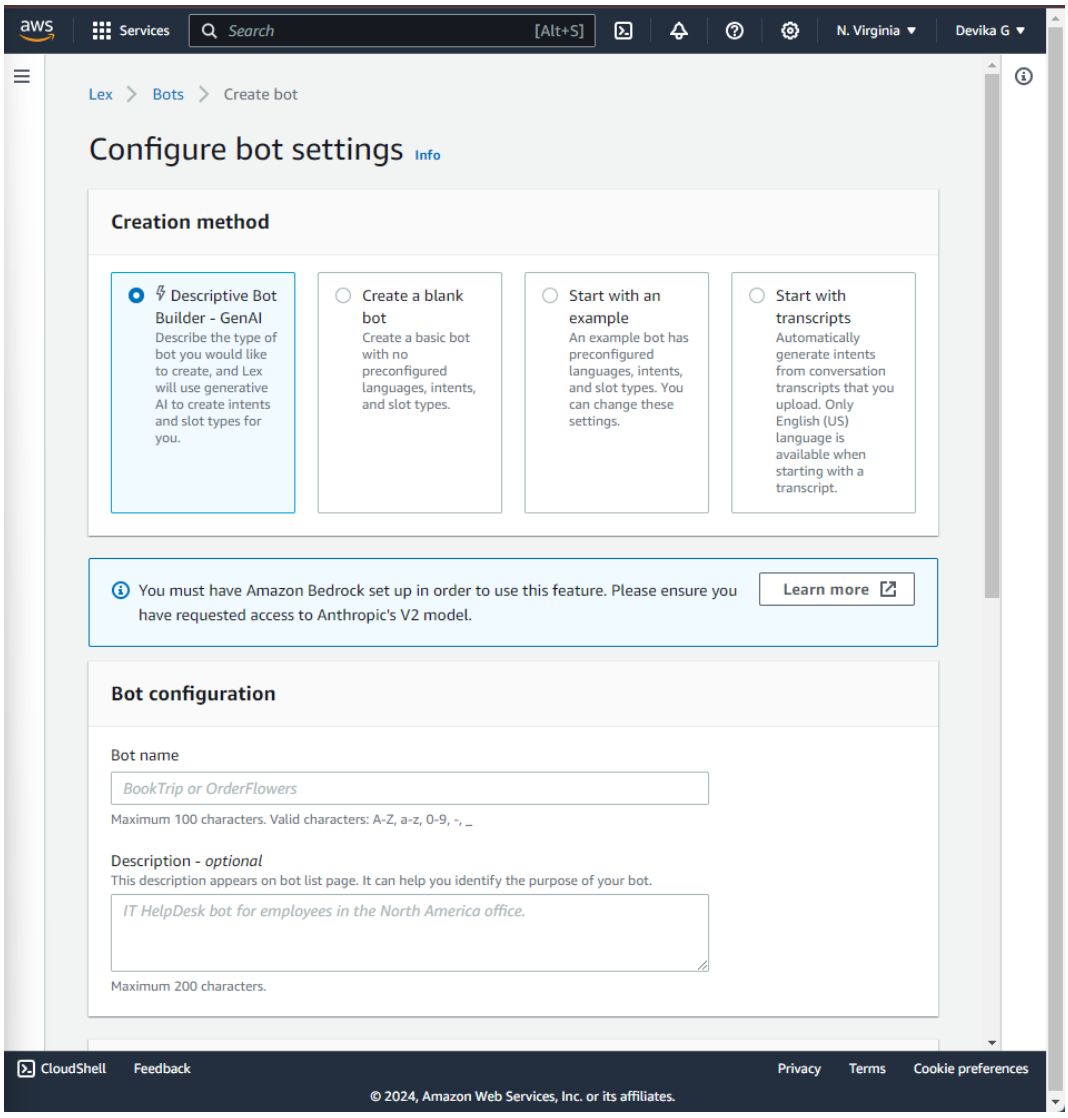
Amazon Web Services. (n.d.). AWS Lambda Documentation. Retrieved from https://docs.aws.amazon.com/lambda/

A) SNAPSHOTS

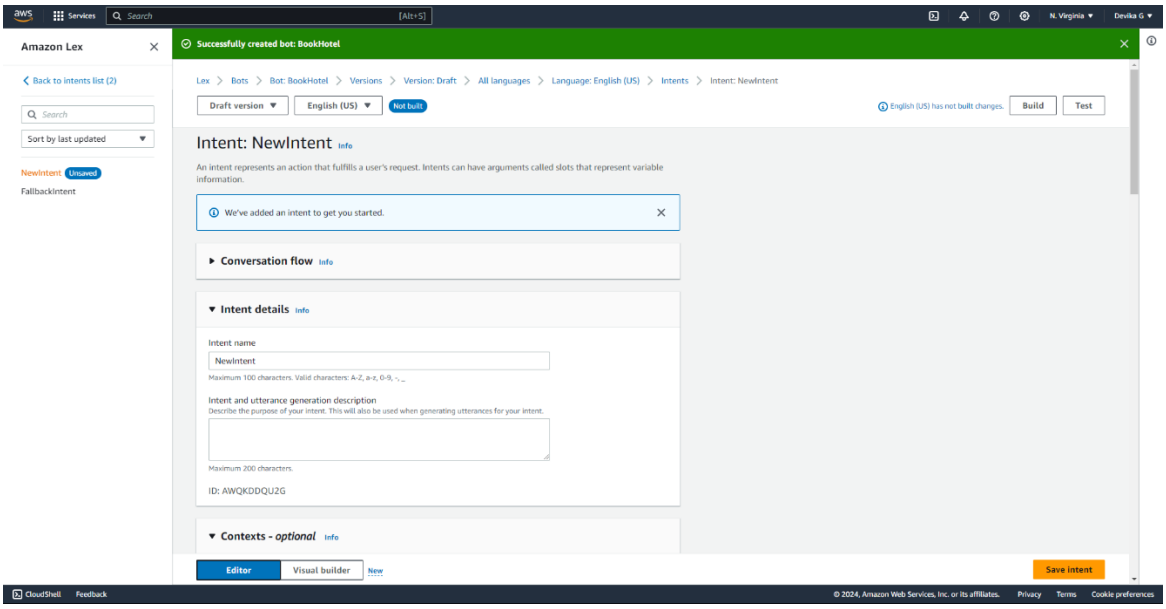
Snapshot 1: Log into AWS management Console and search for Amazon lex in Machine learning service



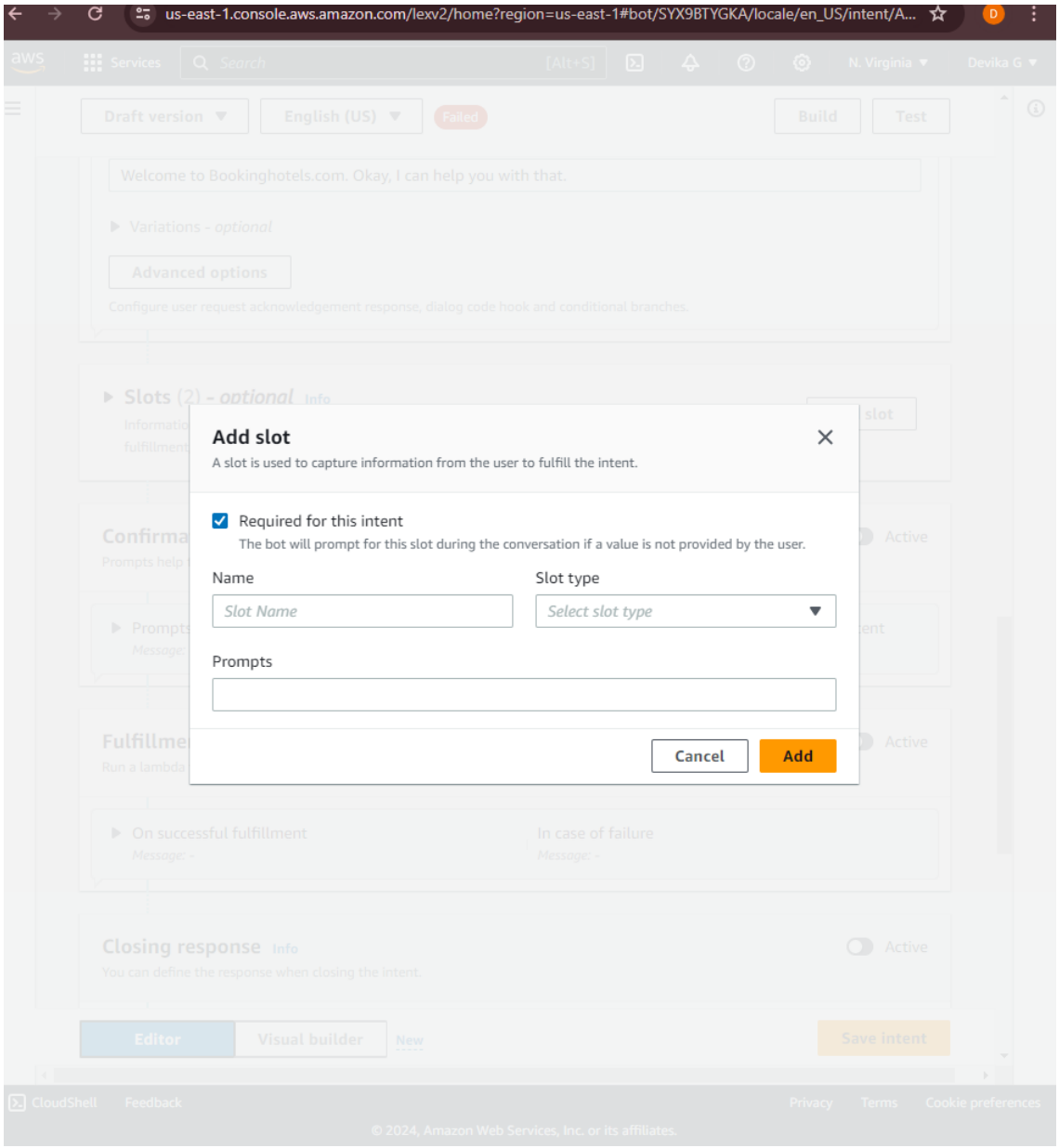
Snapshot 2: Create Bot
Create Bot



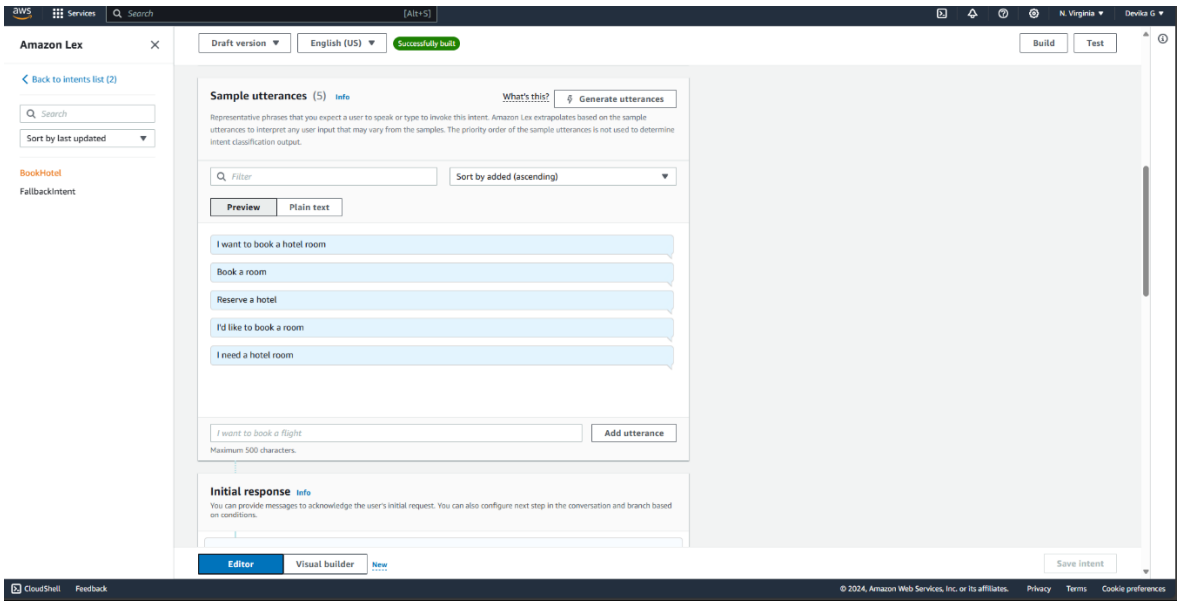
Snapshot 3: Define Intent
Define Intent



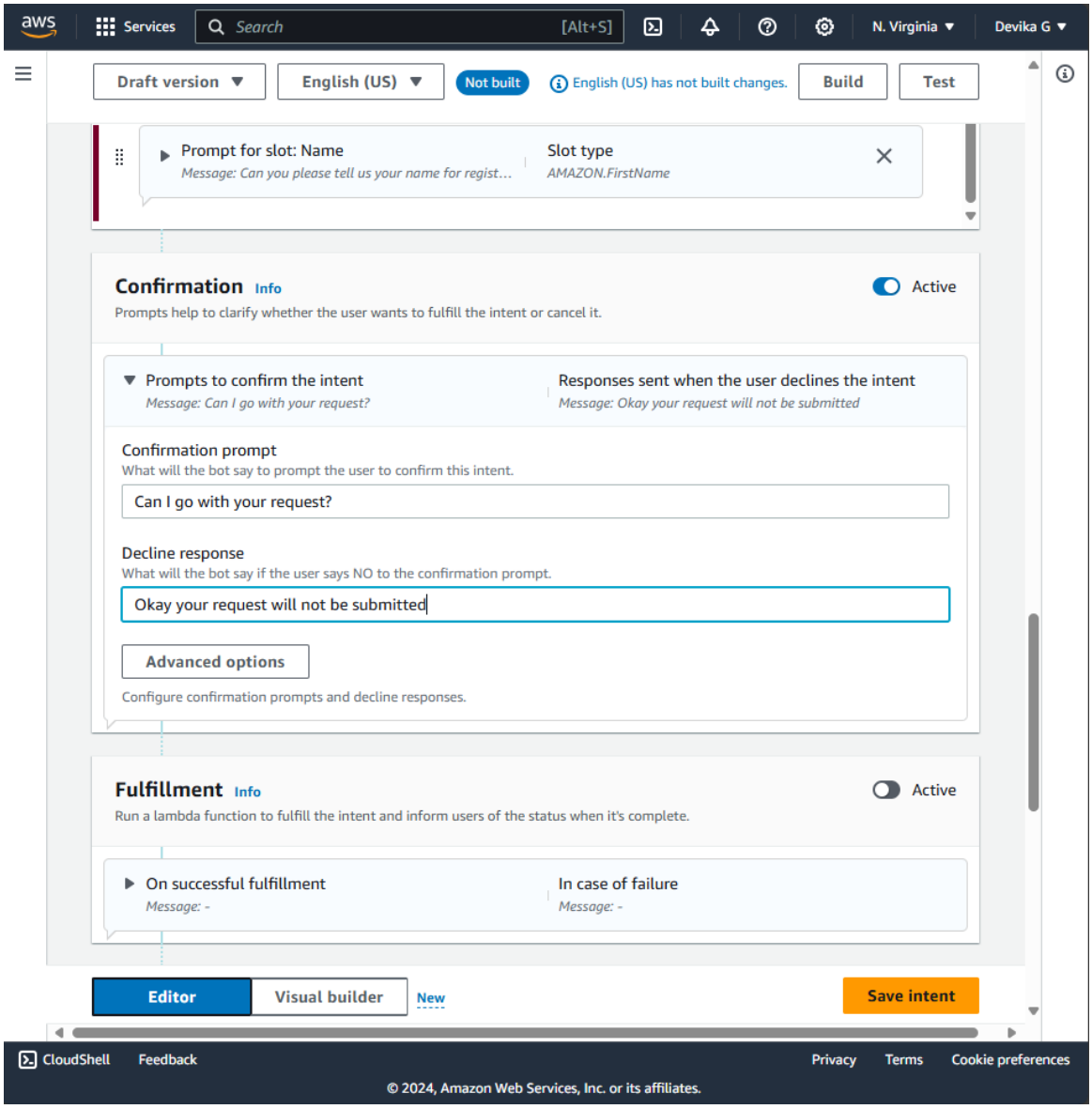
Snapshot 4: Add Slots to Intent
Add Slots to Intent



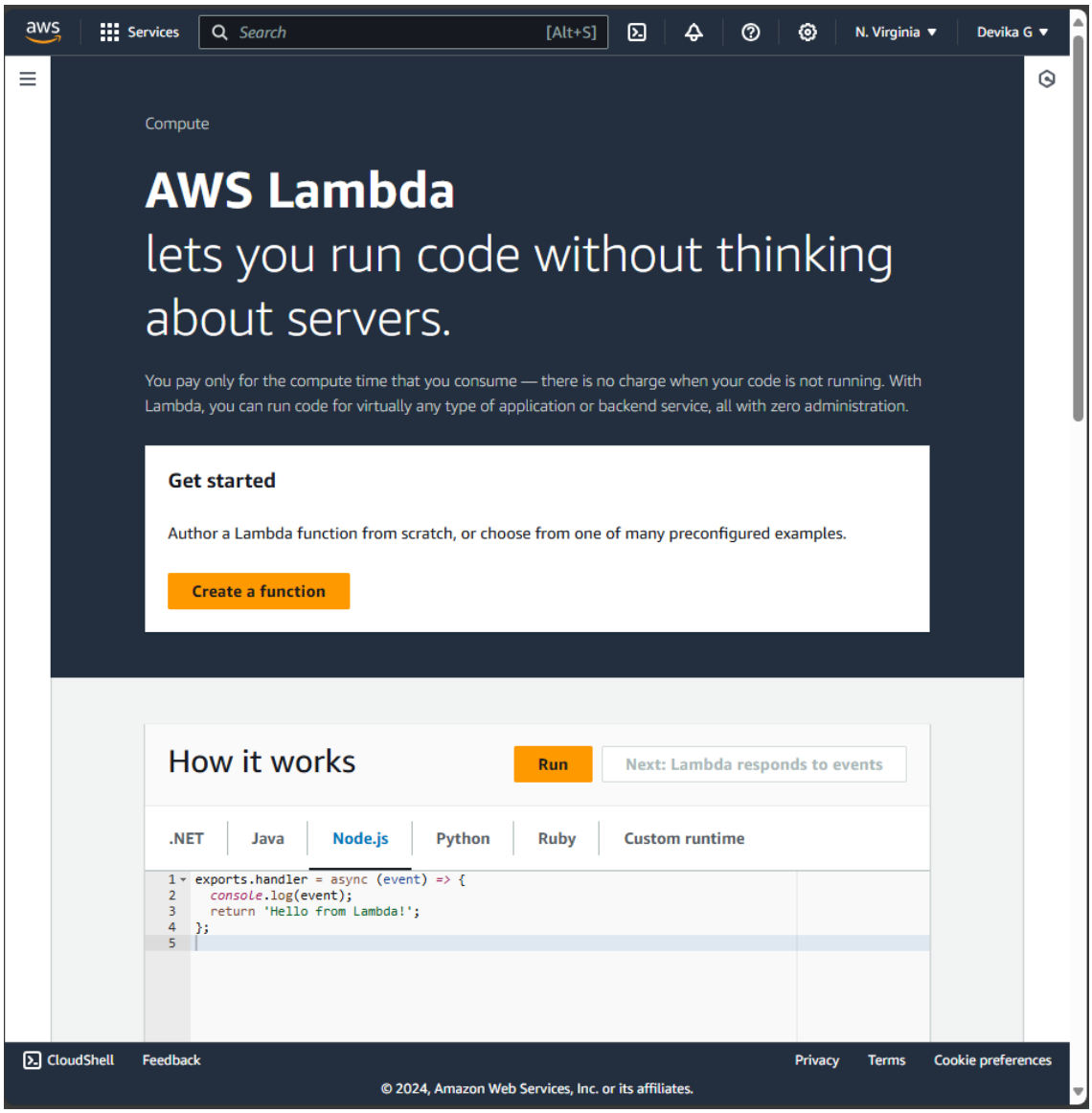
Snapshot 5: Sample Utterances
Sample Utterances



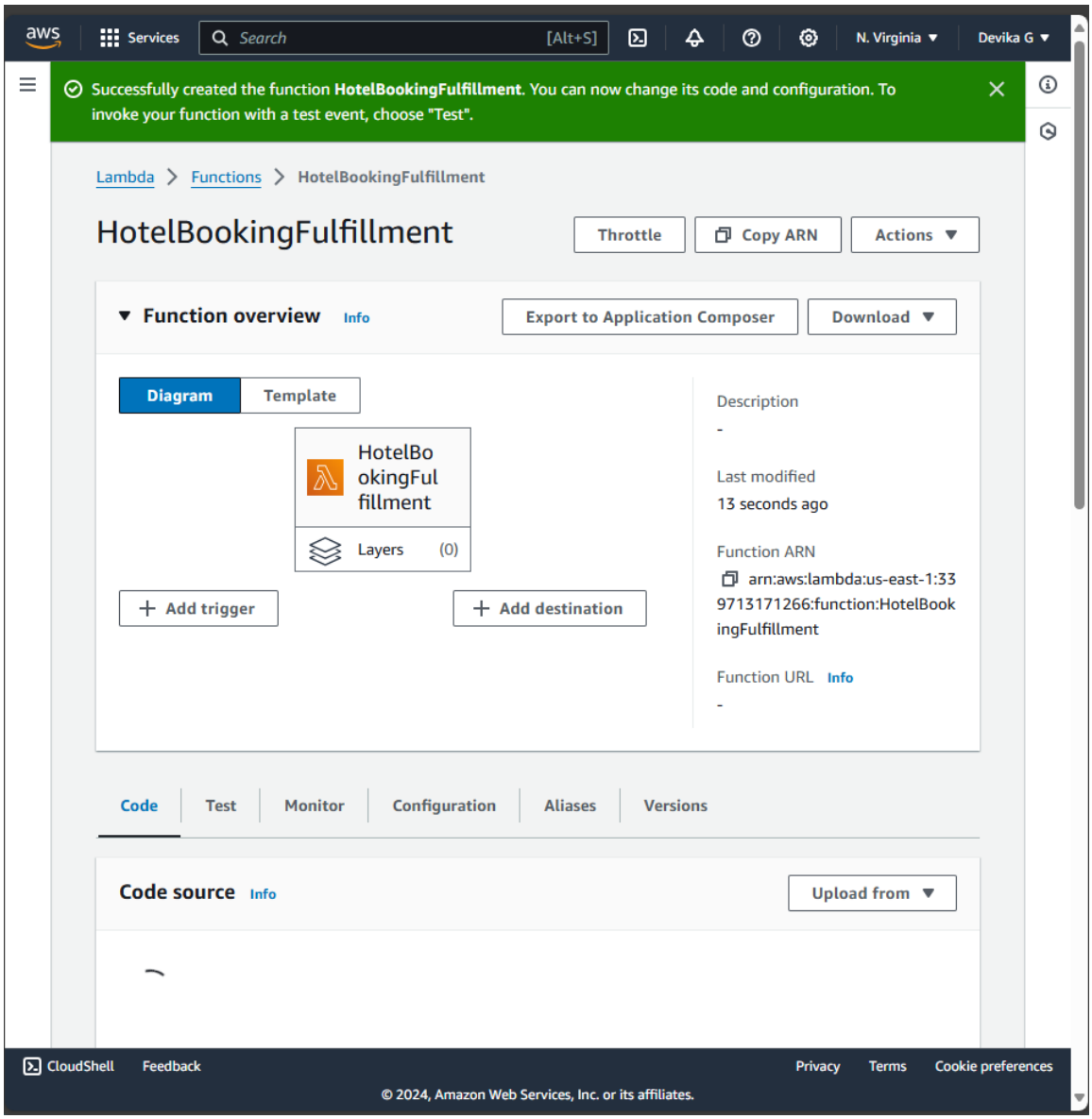
Snapshot 6: Add confirmation prompt



Snapshot 7: Open Lambda Console

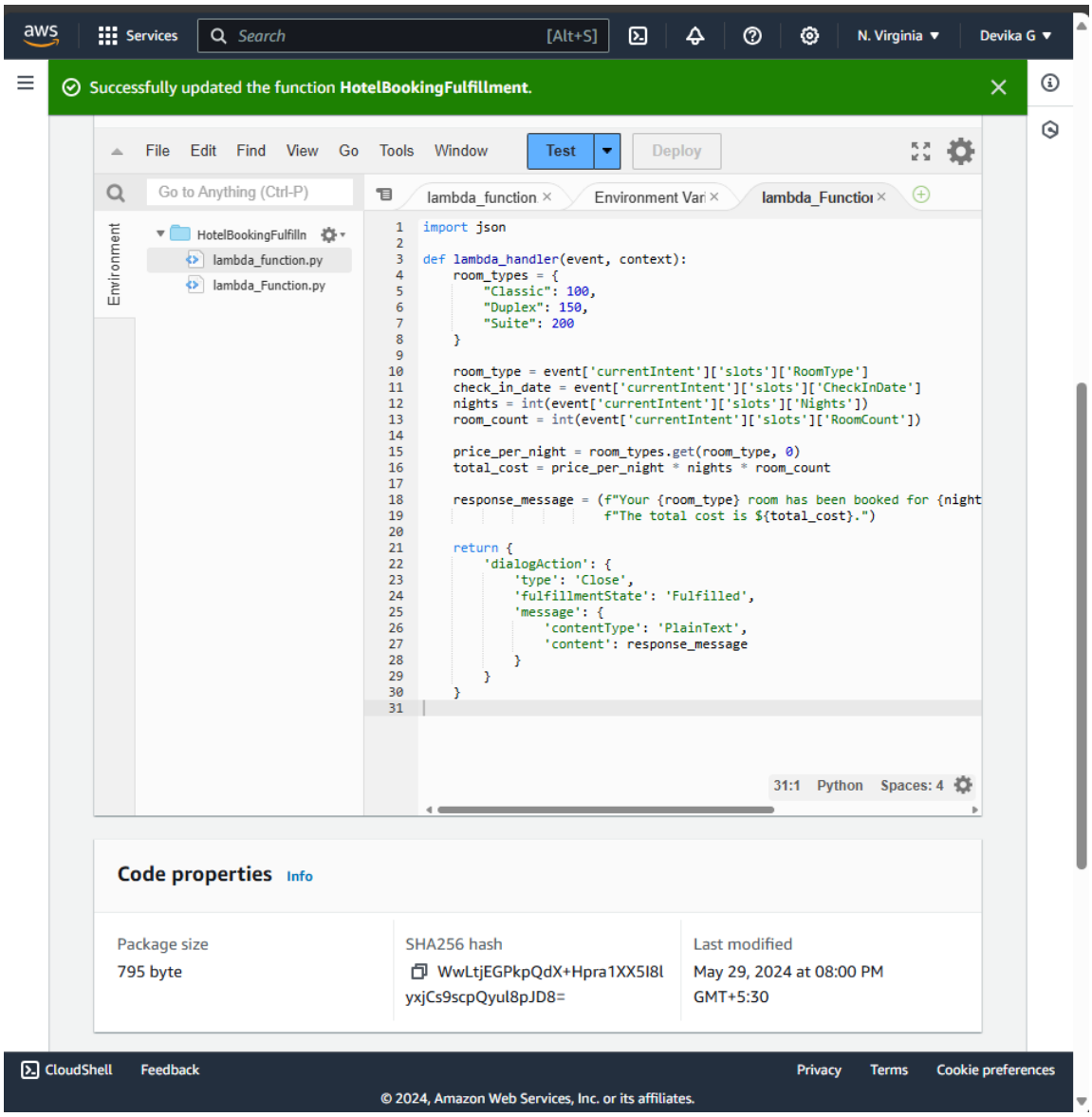


Snapshot 8: Create Lambda Function
Create Lambda Function

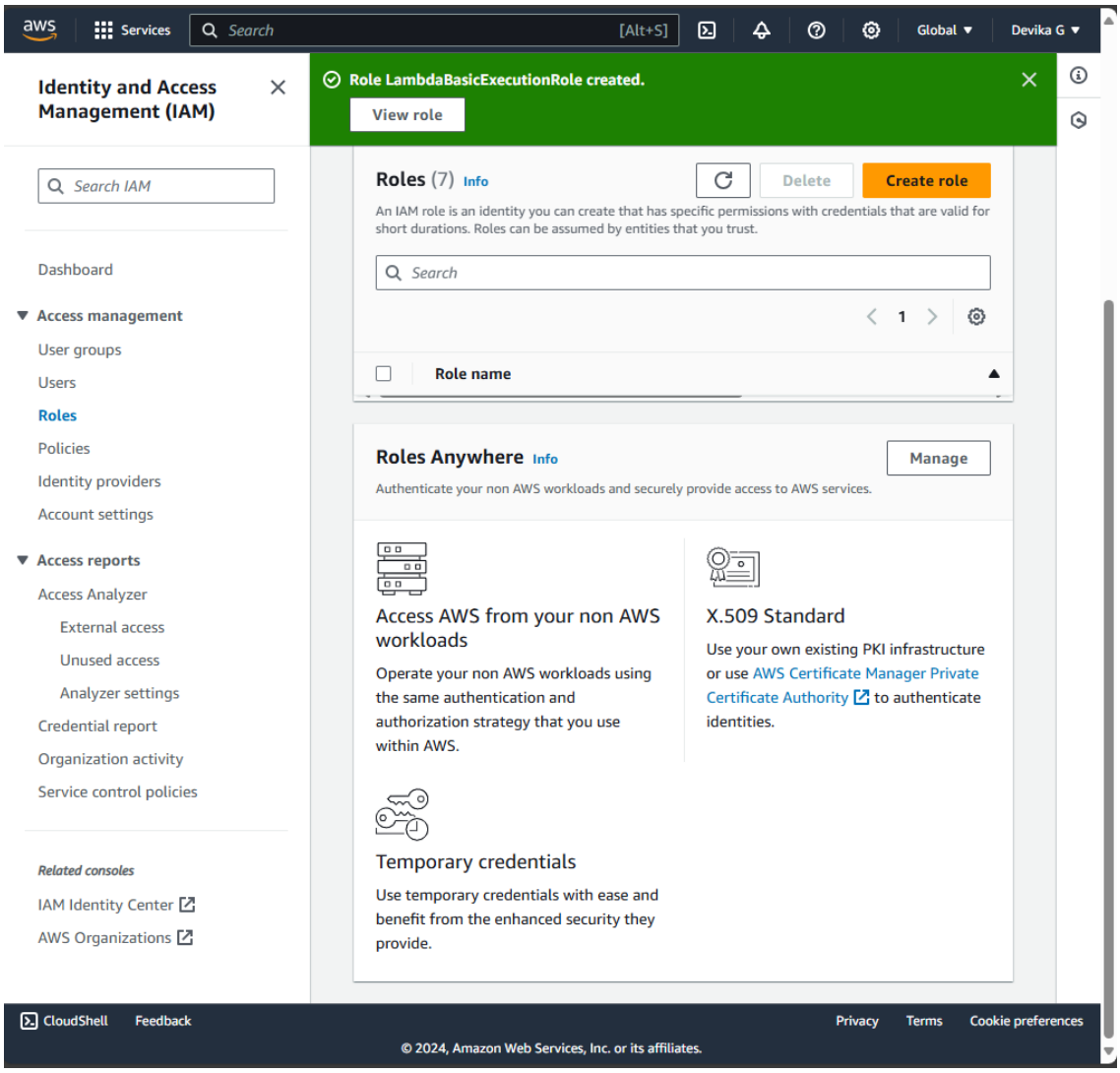


Snapshot 9: Lambda Function Code

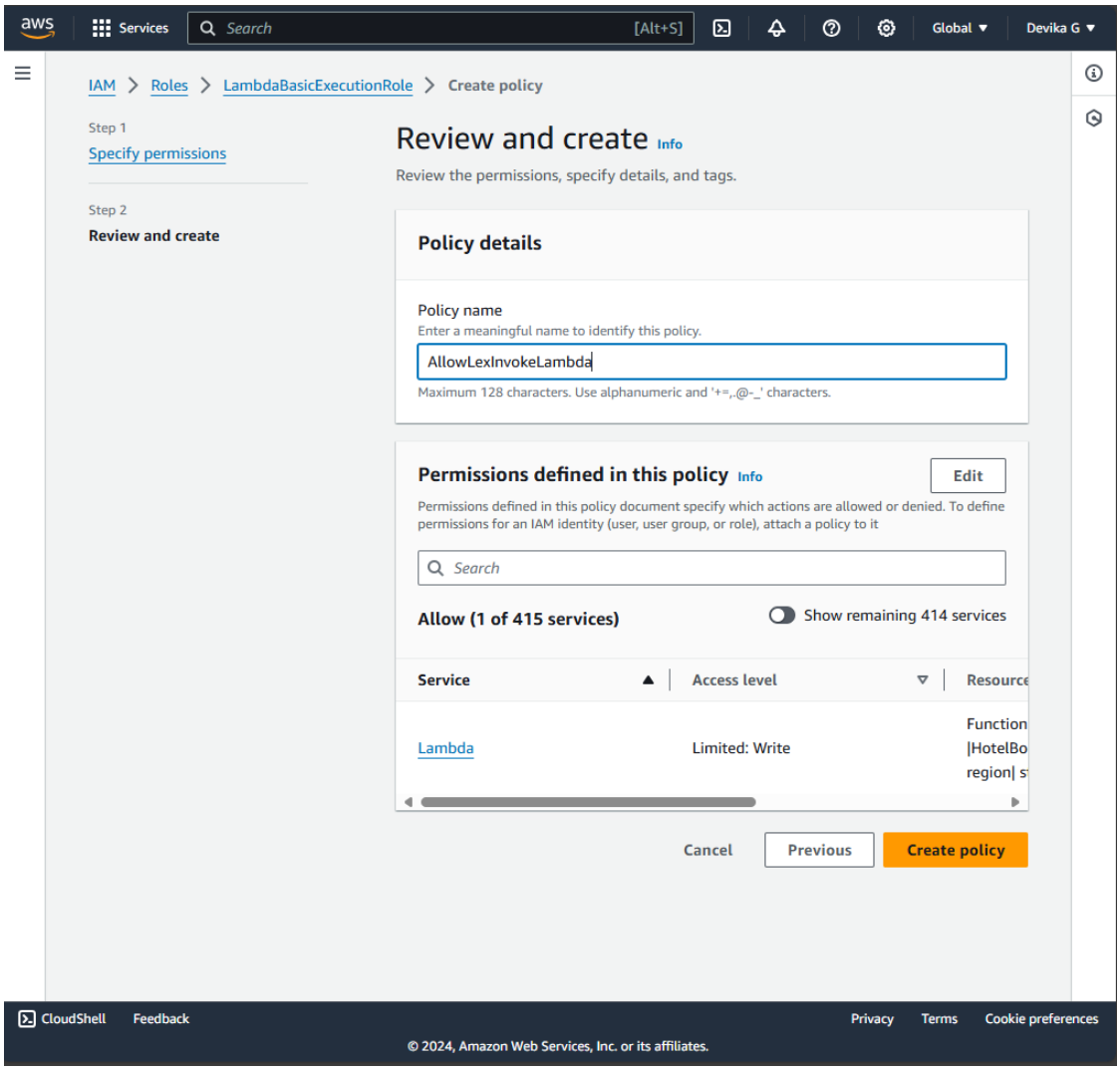
Lambda Function Code



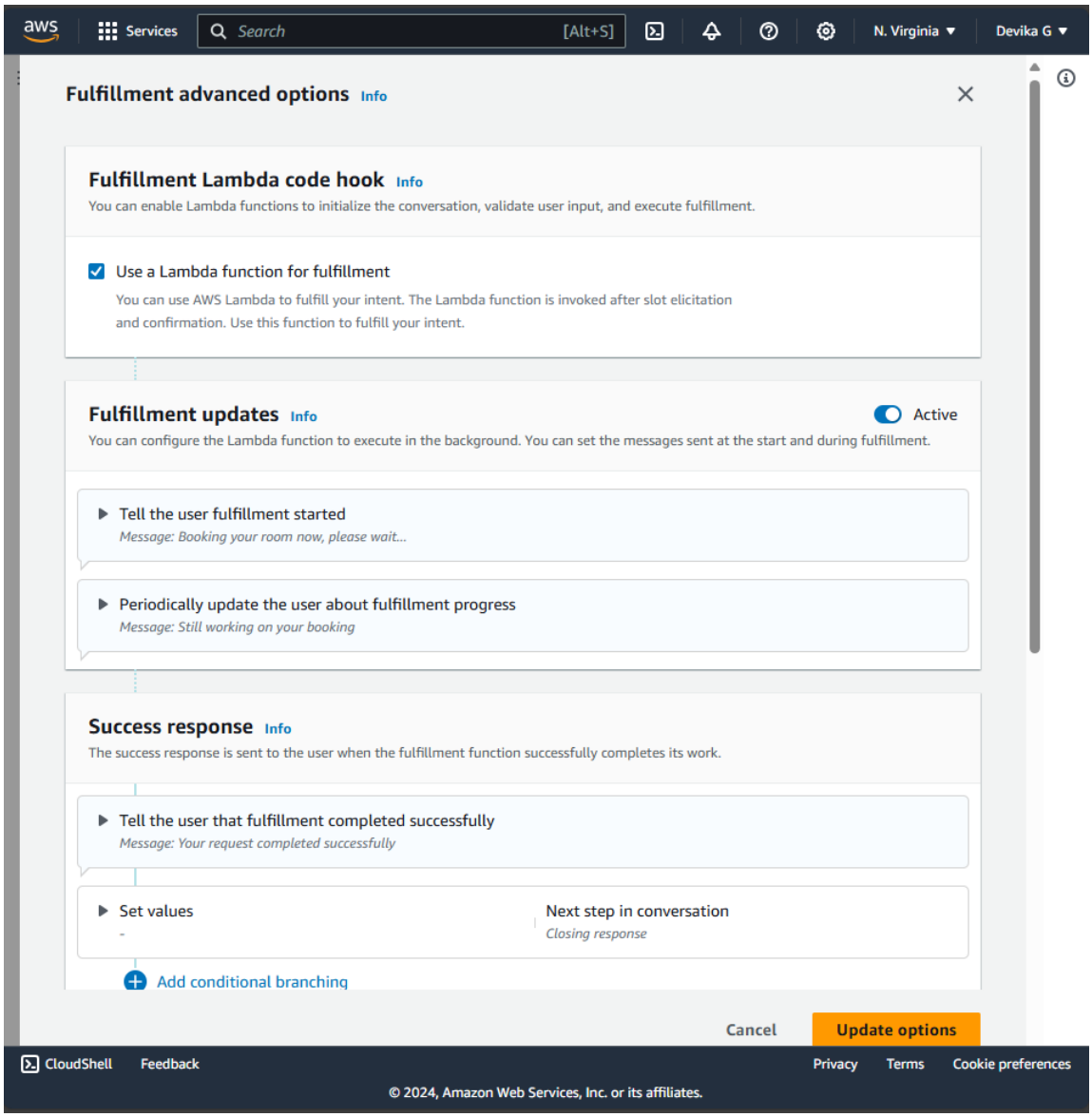
Snapshot 10: Create an IAM role for lambda access permissions



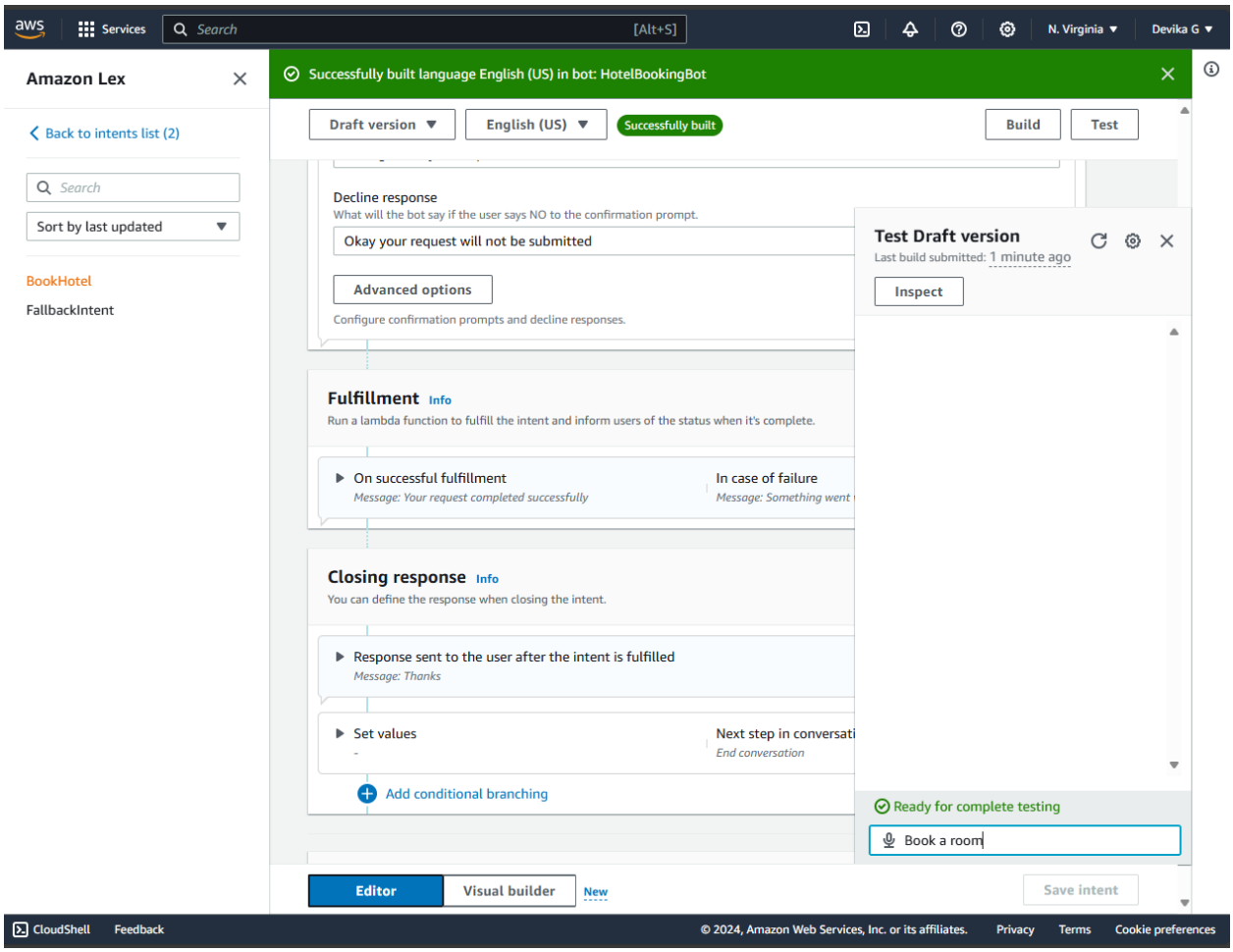
Snapshot 11: Create an Inline Policy and attach to lambda role



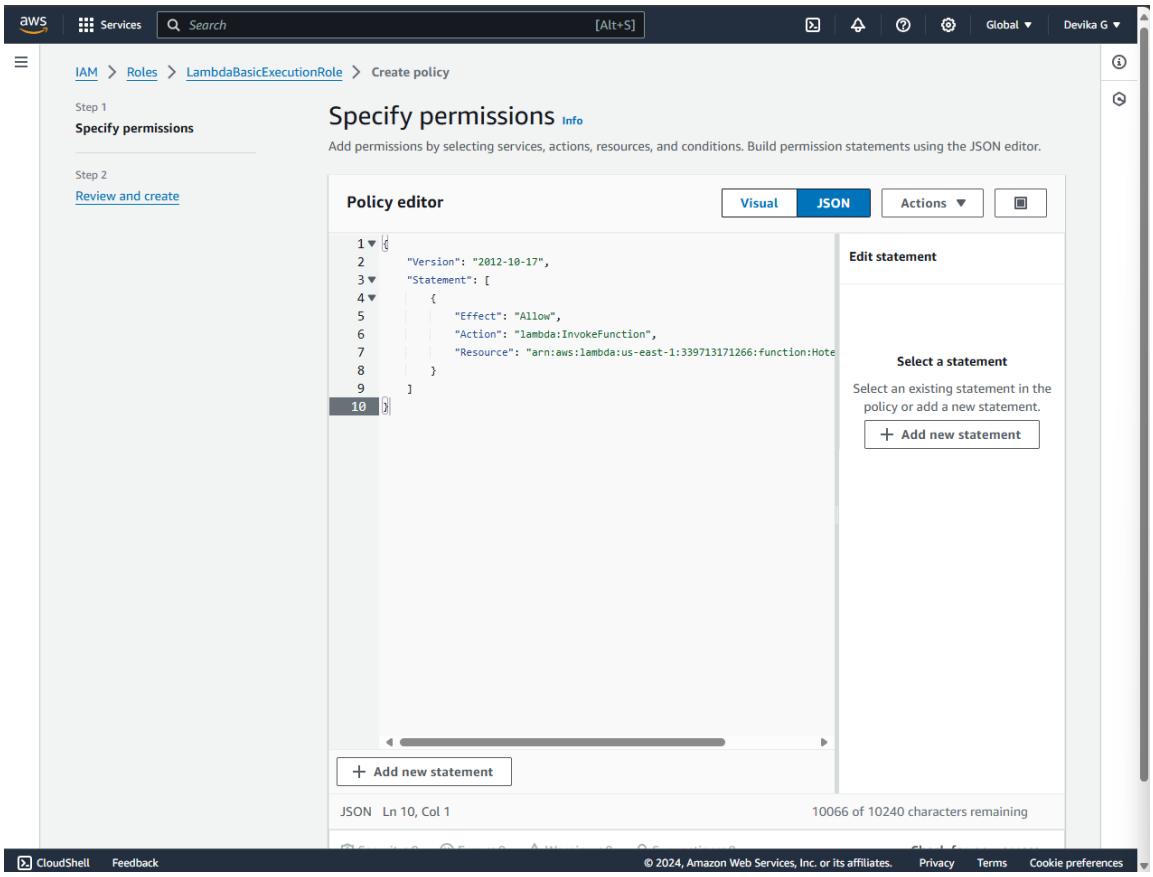
Snapshot 12: Fulfillment and code hooks



Snapshot 13: Save, Build and Test Bot.



Snapshot 14: Create IAM Policy and attach policy to role.



Snapshot 15: Test Chatbot in Amazon Lex Console

Test Chatbot in Amazon Lex Console