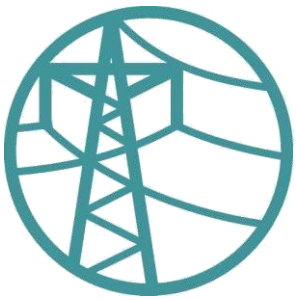


WYDZIAŁ
ELEKTROTECHNIKI
I INFORMATYKI
POLITECHNIKI RZESZOWSKIEJ

Mateusz Rup

Bezserwerowy ChatBot umożliwiający rezerwacje hotelu z wykorzystaniem Amazon Lex i AWS Lambda

Rzeszów, 2024

Spis treści

1. Wstęp

Projekt ten opiera się na integracji zaawansowanych technologii chmurowych oferowanych przez Amazon Web Services (AWS), takich jak Amazon Lex, AWS Lambda i Amazon DynamoDB, które razem tworzą inteligentnego chatbota zdolnego do obsługi rezerwacji hotelowych. Chatbot ten nie tylko automatyzuje proces rezerwacji, ale również zapewnia wysoki poziom interaktywności, umożliwiając użytkownikom korzystanie z naturalnego języka do komunikacji. Amazon Lex odpowiada za analizę i interpretację zapytań użytkowników, podczas gdy AWS Lambda obsługuje logikę biznesową i zarządzanie interakcjami w czasie rzeczywistym. Dane dotyczące rezerwacji są przechowywane w Amazon DynamoDB, wysoko skalowalnej bazie danych NoSQL, która zapewnia szybki i niezawodny dostęp do informacji. Hostowanie na AWS S3 zapewnia wysoką dostępność i bezpieczeństwo systemu, co jest kluczowe dla niezawodnej obsługi rezerwacji.

1.1. Opis problemu

Proces rezerwacji pokoi hotelowych w hotelu stanowi wyzwanie zarówno dla gości, jak i dla personelu. Tradycyjny sposób rezerwacji opiera się głównie na komunikacji telefonicznej, mailowej lub bezpośrednich wizytach w recepcji, co niesie ze sobą szereg problemów i ograniczeń. Tradycyjne metody rezerwacji są ograniczone godzinami pracy recepcji, co oznacza, że goście nie mogą dokonać rezerwacji w dogodnym dla siebie czasie, na przykład w nocy czy podczas dni wolnych od pracy. Brak dostępności w czasie rzeczywistym może prowadzić do sytuacji, w której goście nie są pewni dostępności pokoi w momencie dokonywania rezerwacji.

1.2. Cel projektu

Celem projektu "Bezserwerowy ChatBot umożliwiający rezerwacje hotelu z wykorzystaniem Amazon Lex i AWS Lambda" jest stworzenie nowoczesnego i zautomatyzowanego systemu rezerwacji dla hotelu. Projekt ten ma na celu uproszczenie i przyspieszenie procesu rezerwacji, eliminując konieczność ręcznego zarządzania rezerwacjami przez personel hotelowy. Dzięki wykorzystaniu technologii AWS, system będzie dostępny 24/7, co pozwoli gościom na dokonywanie rezerwacji w dowolnym czasie i z dowolnego miejsca. Chatbot, wyposażony w funkcje przetwarzania języka naturalnego,

zapewni użytkownikom intuicyjną i interaktywną obsługę, poprawiając ich doświadczenia i zadowolenie z usług hotelowych. Automatyzacja procesu rezerwacji zredukuje liczbę błędów, zwiększając tym samym niezawodność i efektywność operacyjną hotelu.

2. Użyte technologie

2.1 Amazon Lambda

Amazon Lambda – to bez serwerowa usługa obliczeniowa oferowana przez AWS, która pozwala na uruchamianie kodu w odpowiedzi na zdarzenia bez konieczności zarządzania serwerami. Lambda automatycznie alokuje zasoby obliczeniowe w miarę potrzeb i skaluje aplikacje bez konieczności interwencji użytkownika. AWS Lambda odgrywa kluczową rolę w projekcie rezerwacji hotelowej, pełniąc funkcję bez serwerowej platformy obliczeniowej. Główne aspekty użycia Lambda w projekcie to:

- Bez serwerowe uruchamianie kodu: Lambda umożliwia uruchamianie kodu bez konieczności zarządzania serwerami. W naszym projekcie funkcje Lambda są wywoływane w odpowiedzi na zapytania użytkowników przesyłane przez Amazon Lex.

- Obsługa logiki biznesowej: Lambda przetwarza zebrane przez chatbota dane, weryfikuje poprawność informacji, takie jak format daty i dane kontaktowe, oraz obsługuje logikę biznesową związaną z rezerwacjami.

- Interakcje z DynamoDB: Funkcje Lambda zapisują i odczytują dane z bazy Amazon DynamoDB, zarządzając rezerwacjami użytkowników. W szczególności, Lambda zapisuje szczegóły rezerwacji do DynamoDB po pomyślnym zweryfikowaniu danych.

- Skalowalność i elastyczność: Lambda automatycznie skaluje zasoby w odpowiedzi na zapotrzebowanie, co jest kluczowe dla obsługi potencjalnie dużej liczby zapytań użytkowników bez opóźnień.

- Walidacja danych: Funkcje Lambda przeprowadzają walidację danych zebranych przez chatbota, sprawdzając ich poprawność i kompletność. Na przykład, weryfikowane są formaty dat, poprawność adresów e-mail oraz czy wszystkie wymagane pola zostały wypełnione. Jeśli dane są

nieprawidłowe, Lambda zwraca odpowiedni komunikat do Lexa, aby ponownie zapytać użytkownika o poprawne informacje.

```python
import json
import re
from datetime import datetime, date
import boto3
import uuid

# Initialize DynamoDB client
dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('HarmonyPeaks')

# Define hotel names and room types
hotels = ['Harmony Peaks Bora Bora', 'Harmony Peaks Imperial', 'Harmony Peaks Deluxe', 'Harmony Peaks Alpine Haven', 'Harmony Peaks Oasis']
room_types = ['Standard room', 'Deluxe room', 'Suite', 'Penthouse', 'Bungalow']

def validate_phone_number(phone):
    # Validate phone number using a regex pattern
    phone_pattern = r"^\+[\d]{1,14}$"
    return re.match(phone_pattern, phone) is not None

def validate_reservation(slots):
    # Validate Hotel
    if not slots['Hotel']:
        return {
            'isValid': False,
            'invalidSlot': 'Hotel',
            'message': 'Which hotel would you like to book a room at?'
        }

    # Validate Room type
    if not slots['Rooms']:
        return {
            'isValid': False,
            'invalidSlot': 'Rooms',
            'message': 'Which room would you like to choose?'
        }
    elif slots['Rooms']['value']['originalValue'] not in room_types:
        return {
            'isValid': False,
            'invalidSlot': 'Rooms',
            'message': 'Invalid room type. Please select from: {}.'.format(", ".join(room_types))
        }

    # Validate CheckIn date
    if not slots['CheckIn']:
        return {
            'isValid': False,
            'invalidSlot': 'CheckIn',
            'message': 'Please provide the Check-in date.'
        }
    else:
        try:
            check_in_date = datetime.strptime(slots['CheckIn']['value']['originalValue'], "%d-%m-%Y")
            today = date.today()
```

Rys 2.1a. Kod w Lambda

```python
            today = date.today()
            if check_in_date.date() < today:
                raise ValueError
        except ValueError:
            return {
                'isValid': False,
                'invalidSlot': 'CheckIn',
                'message': 'Invalid date for Check-in. Please provide a date from today onwards in "dd-mm-yyyy" format.'
            }

    # Validate CheckOut date
    if not slots['CheckOut']:
        return {
            'isValid': False,
            'invalidSlot': 'CheckOut',
            'message': 'Please provide the Check-out date.'
        }
    else:
        try:
            check_out_date = datetime.strptime(slots['CheckOut']['value']['originalValue'], "%d-%m-%Y")
            if check_out_date < check_in_date:
                raise ValueError
        except ValueError:
            return {
                'isValid': False,
                'invalidSlot': 'CheckOut',
                'message': 'Invalid date for Check-out. Please provide a date that is after the Check-in date in "dd-mm-yyyy" format.'
            }

    # Validate FirstName, LastName, Email, Phone
    if not slots['FirstName']:
        return {
            'isValid': False,
            'invalidSlot': 'FirstName',
            'message': 'Now I need your data to book this room. Could you please provide your first name?'
        }
    if not slots['LastName']:
        return {
            'isValid': False,
            'invalidSlot': 'LastName',
            'message': 'Could you please provide your last name?'
        }
    if not slots['Email']:
        return {
            'isValid': False,
            'invalidSlot': 'Email',
            'message': 'Could you please provide your email?'
        }
    if not slots['Phone']:
```

Rys. 2.1b. Kod w Lambda

```
    if not slots['Phone']:
        return {
            'isValid': False,
            'invalidSlot': 'Phone',
            'message': 'Could you please provide your phone number?'
        }
    elif not validate_phone_number(slots['Phone']['value']['originalValue']):
        return {
            'isValid': False,
            'invalidSlot': 'Phone',
            'message': 'Invalid phone number format. Please provide a valid phone number starting with a "+" sign.'
        }

    # If all validations pass
    return {'isValid': True}

def save_reservation_to_dynamodb(slots):
    # Save the reservation details to DynamoDB
    table.put_item(
        Item={
            'Id': str(uuid.uuid4()),  # Unique ID for each reservation
            'Hotel': slots['Hotel']['value']['originalValue'],
            'RoomType': slots['Rooms']['value']['originalValue'],
            'CheckIn': slots['CheckIn']['value']['originalValue'],
            'CheckOut': slots['CheckOut']['value']['originalValue'],
            'FirstName': slots['FirstName']['value']['originalValue'],
            'LastName': slots['LastName']['value']['originalValue'],
            'Email': slots['Email']['value']['originalValue'],
            'Phone': slots['Phone']['value']['originalValue'],
        }
    )

def lambda_handler(event, context):
    print(event)

    # Extract slots and intent name from the event
    slots = event['sessionState']['intent']['slots']
    intent = event['sessionState']['intent']['name']

    # Validate reservation details
    reservation_validation_result = validate_reservation(slots)

    # Handle dialog flow
    if event['invocationSource'] == 'DialogCodeHook':
        if reservation_validation_result['isValid']:
            # Delegate dialog back to Lex
            response = {
                "sessionState": {
                    "dialogAction": {
                        "type": "Delegate"
                    },
                    "intent": {
                        'name': intent,
                        'slots': slots
```

Rys. 2.1c. Kod w Lambda

```
                        'slots': slots
                    }
                }
            }
        else:
            # Elicit the invalid slot from the user
            response = {
                "sessionState": {
                    "dialogAction": {
                        "slotToElicit": reservation_validation_result['invalidSlot'],
                        "type": "ElicitSlot"
                    },
                    "intent": {
                        "name": intent,
                        "slots": slots
                    }
                },
                "messages": [
                    {
                        "contentType": "PlainText",
                        "content": reservation_validation_result['message']
                    }
                ]
            }

            # Provide options if the invalid slot is Hotel or Rooms
            if reservation_validation_result['invalidSlot'] == 'Hotel':
                response_card_sub_title = "Choose a hotel from the options below:"
                response_card_buttons = [
                    {"text": hotel, "value": hotel} for hotel in hotels[:5]
                ]
                response['messages'].append({
                    "contentType": "ImageResponseCard",
                    "content": "Which hotel would you like to book a room at?",
                    "imageResponseCard": {
                        "title": "Hotel Selection",
                        "imageUrl": "https://harmonypeaksbot.s3.eu-central-1.amazonaws.com/images/logomale.jpg",
                        "subtitle": response_card_sub_title,
                        "buttons": response_card_buttons
                    }
                })

            elif reservation_validation_result['invalidSlot'] == 'Rooms':
                response_card_sub_title = "Choose a room type from the options below:"
                response_card_buttons = [
                    {"text": room_type, "value": room_type} for room_type in room_types
                ]
                response['messages'].append({
                    "contentType": "ImageResponseCard",
                    "content": "Which room would you like to choose?",
                    "imageResponseCard": {
                        "title": "Room Selection",
                        "imageUrl": "https://harmonypeaksbot.s3.eu-central-1.amazonaws.com/images/logomale.jpg",
                        "subtitle": response_card_sub_title,
```

Rys. 2.1d. Kod w Lambda

```
                    "subtitle": response_card_sub_title,
                    "buttons": response_card_buttons
                }
            })

# Handle fulfillment
elif event['invocationSource'] == 'FulfillmentCodeHook':
    if reservation_validation_result['isValid']:
        save_reservation_to_dynamodb(slots)
        response = {
            "sessionState": {
                "dialogAction": {
                    "type": "Close"
                },
                "intent": {
                    "name": intent,
                    "slots": slots,
                    "state": "Fulfilled"
                },
                "messages": [
                    {
                        "contentType": "PlainText",
                        "content": "Okay, I have booked your room at {}.".format(slots['Hotel']['value']['originalValue'])
                    },
                    {
                        "contentType": "PlainText",
                        "content": "See you in our hotel!"
                    }
                ]
            }
        }
    else:
        # Handle unexpected failure
        response = {
            "sessionState": {
                "dialogAction": {
                    "type": "Close"
                },
                "intent": {
                    "name": intent,
                    "slots": slots,
                    "state": "Failed"
                },
                "messages": [
                    {
                        "contentType": "PlainText",
                        "content": "Ooops, something went wrong!"
                    }
                ]
            }
        }

print(response)
return response
```

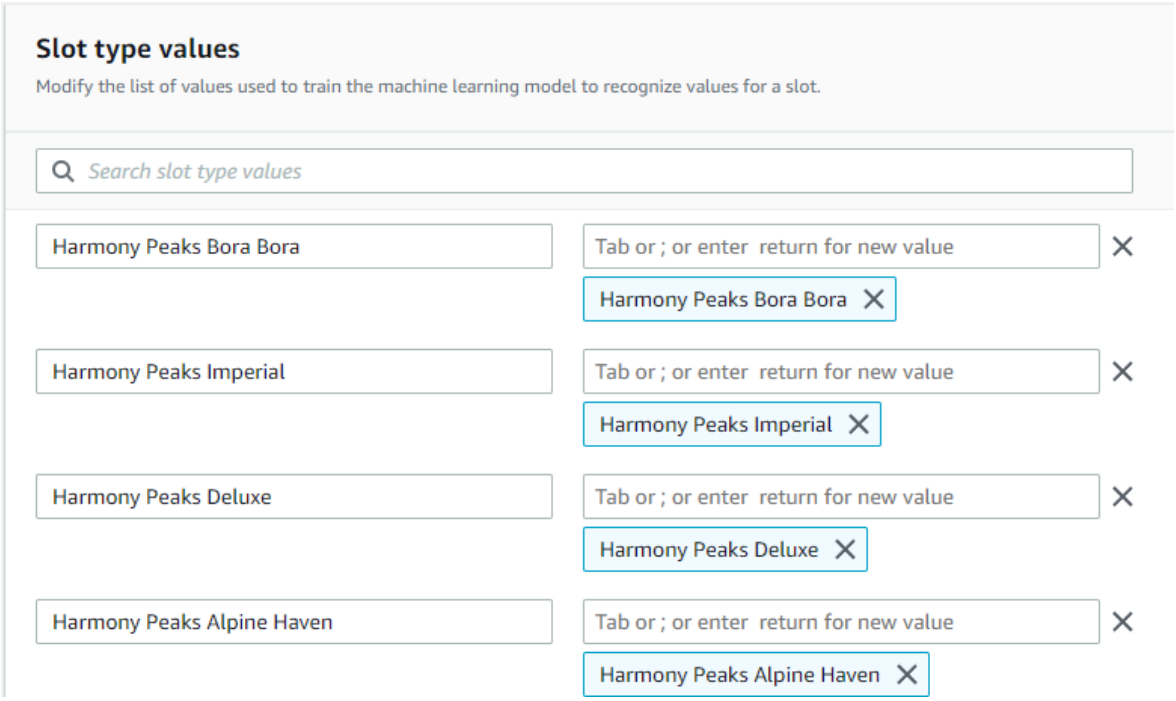
Rys. 2.1e. Kod w Lambda

2.2 Amazon Lex

Amazon Lex – jest usługą przetwarzania języka naturalnego (NLP) oraz rozpoznawania mowy oferowaną przez AWS. Jest używany do tworzenia interfejsów konwersacyjnych dla aplikacji. Amazon Lex jest fundamentem systemu interfejsu konwersacyjnego, umożliwiając użytkownikom interakcję z chatbotem w sposób naturalny. Główne aspekty użycia Lexa w projekcie to:

- Rozpoznawanie mowy i tekstu: Lex przetwarza zarówno tekst, jak i mowę, pozwalając użytkownikom komunikować się z chatbotem w naturalnym języku. Dzięki temu użytkownicy mogą łatwo wprowadzać zapytania o rezerwacje.
- Zarządzanie dialogiem: Lex prowadzi interakcje z użytkownikami, zadając odpowiednie pytania w celu zebrania wszystkich niezbędnych informacji, takich jak nazwa hotelu, typ pokoju, daty pobytu, imię, nazwisko, adres e-mail i numer telefonu.

- Definiowanie intencji i slotów: W Lexie definiowane są intencje (np. "BookHotel") i sloty (np. "Hotel", "CheckIn", "CheckOut"), które strukturalizują rozmowę i umożliwiają precyzyjne zbieranie danych.



Rys 2.2. Przykład użytego slotu

- Integracja z AWS Lambda: Lex wywołuje funkcje Lambda do przetwarzania zebranych danych, weryfikacji informacji i zarządzania logiką rezerwacji.

2.2.1 Tworzenie nowego Chatbota

Aby utworzyć nowego chatbota w Amazon Lex:

a) Zaczynamy od:
   - Kliknij przycisk "Create bot".
   - Wybierz "Create" pod "Custom bot".
   - Podaj nazwę bota, np. ReservationBot.
   - Wybierz język, np. English (US).

b) Następnie dodajemy intencje do bota:
   - Kliknij "Create intent".
   - Podajemy nazwę intencji, np. MakeReservation.

c) Konfigurujemy sloty intencji:

- W sekcji "Slot types", dodajemy interesujące nas sloty np.: ReservationDate (Type: AMAZON.DATE)
- Wprowadzamy przykładowe pytania dla każdego slotu, np. "What date would you like to make a reservation for?" dla ReservationDate.

d) Konfigurujemy fulfillment:

- Przechodzimy do sekcji "Fulfillment and closing".
- Wybieramy "AWS Lambda function" jako fulfillment method.
- Wybieramy utworzoną funkcję Lambda .

e) Na koniec zapisujemy intencję i bota, a następnie go testujemy:

- Klikamy "Save intent".
- Klikamy "Build" w prawym górnym rogu, aby zbudować bota.
- Przechodzimy do zakładki "Test" w konsoli Lex.
- Testujemy bota, wprowadzając przykładowe dane, np. "I want to make a reservation".

2.3 Amazon DynamoDB

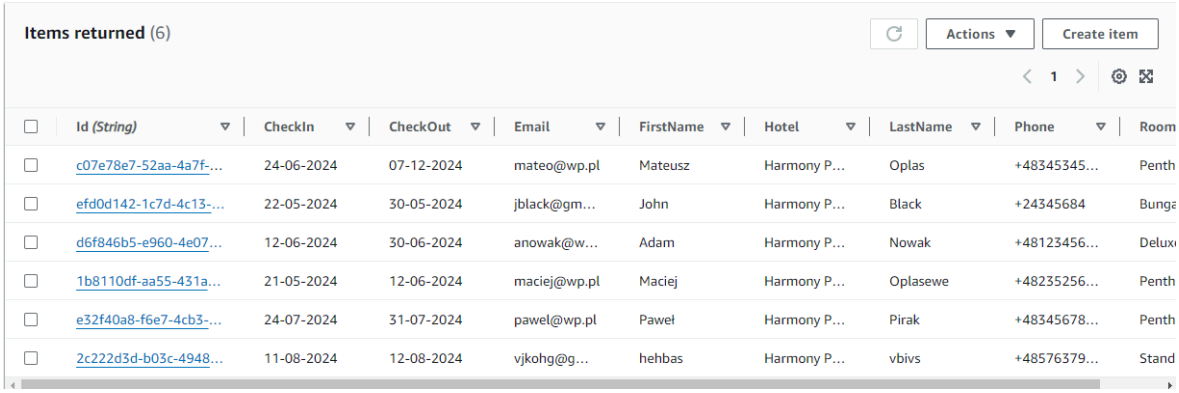
Amazon DynamoDB – służy jako nasza główna baza danych NoSQL, przechowując informacje o rezerwacjach hotelowych. Kluczowe cechy DynamoDB w kontekście naszego projektu to:

- Wysoka dostępność i niski czas odpowiedzi: DynamoDB zapewnia niskie opóźnienia na poziomie milisekund, co jest kluczowe dla szybkości działania naszego chatbota i systemu rezerwacji.

- Skalowalność: DynamoDB automatycznie skaluje się w odpowiedzi na rosnące obciążenie, co jest niezbędne do obsługi dużej liczby jednoczesnych rezerwacji bez utraty wydajności.

- Elastyczne przechowywanie danych: DynamoDB umożliwia przechowywanie różnorodnych danych, takich jak informacje o użytkownikach, szczegóły rezerwacji (np. daty, typ pokoju) i dane kontaktowe, w elastycznym modelu tabelarycznym.

- Bezpieczeństwo i niezawodność: DynamoDB zapewnia wysoką dostępność danych oraz automatyczne tworzenie kopii zapasowych, co chroni dane użytkowników i zapewnia ich trwałość w przypadku awarii.



Items returned (6)

Actions ▼ | Create item

1

Id (String)	CheckIn	CheckOut	Email	FirstName	Hotel	LastName	Phone	Room
c07e78e7-52aa-4a7f-...	24-06-2024	07-12-2024	mateo@wp.pl	Mateusz	Harmony P...	Oplas	+48345345...	Penth
efd0d142-1c7d-4c13-...	22-05-2024	30-05-2024	jblack@gm...	John	Harmony P...	Black	+24345684	Bunga
d6f846b5-e960-4e07...	12-06-2024	30-06-2024	anowak@w...	Adam	Harmony P...	Nowak	+48123456...	Delux
1b8110df-aa55-431a...	21-05-2024	12-06-2024	maciej@wp.pl	Maciej	Harmony P...	Oplasewe	+48235256...	Penth
e32f40a8-f6e7-4cb3-...	24-07-2024	31-07-2024	pawel@wp.pl	Paweł	Harmony P...	Pirak	+48345678...	Penth
2c222d3d-b03c-4948...	11-08-2024	12-08-2024	vjkohg@g...	hehbas	Harmony P...	vbivs	+48576379...	Stand

Rysunek 2.3. Wygląd tabeli w bazie DynamoDB

3. Sposób działania chatbota

1) Inicjacja Rozmowy – Chatbot rozpoczyna swoją interakcję z użytkownikiem, kiedy ten zainicjuje rozmowę na stronie internetowej hotelu Harmony Peaks, na WhatsApp poprzez Twilio lub innym zintegrowanym kanałem. Użytkownik wprowadza swoje zapytanie dotyczące rezerwacji pokoju w hotelu.

2) Przetwarzanie Języka Naturalnego (NLP) – Gdy użytkownik wyśle swoją wiadomość, chatbot przekazuje tekst do Amazon Lex, który jest odpowiedzialny za przetwarzanie języka naturalnego (NLP). Amazon Lex analizuje tekst, aby zrozumieć, jaka jest intencja użytkownika. Proces ten obejmuje:

- Tokenizacja: Dzielenie tekstu na mniejsze jednostki (tokeny), takie jak słowa czy frazy.
- Analiza składniowa: Określanie struktury gramatycznej zdania.
- Rozpoznawanie intencji: Identyfikowanie intencji użytkownika.
- Elicitacja Slotów: Amazon Lex identyfikuje wymagane informacje (sloty) potrzebne do zrealizowania intencji użytkownika.

3) Inicjacja Slotów – Amazon Lex używa zaprogramowanych dialogów do zbierania niezbędnych informacji od użytkownika. Sloty to konkretne dane, które są

wymagane do dokonania rezerwacji. Proces ten obejmuje zadawanie pytań, takich jak np:

- "Który hotel chciałbyś zarezerwować?"
- "Jakiego typu pokój preferujesz?"
- "Jaka jest data zameldowania?"

4) Walidacja Danych – Zebrane informacje są przekazywane do funkcji AWS Lambda, która odpowiada za walidację. Funkcja Lambda sprawdza:

- Poprawność formatu dat: Czy daty zameldowania i wymeldowania są w prawidłowym formacie i logicznie poprawne.
- Format numeru telefonu: Czy numer telefonu jest w prawidłowym formacie (zaczyna się od "+" i zawiera tylko cyfry).
- Format adresu e-mail: Czy adres e-mail jest poprawnie sformatowany.

Jeśli jakiekolwiek dane są niepoprawne lub brakuje informacji, chatbot prosi użytkownika o ponowne wprowadzenie danych. Na przykład, jeśli użytkownik podał datę nieprawidłowym formacie, chatbot poprosi o podanie daty ponownie, podając przykład poprawnego formatu.

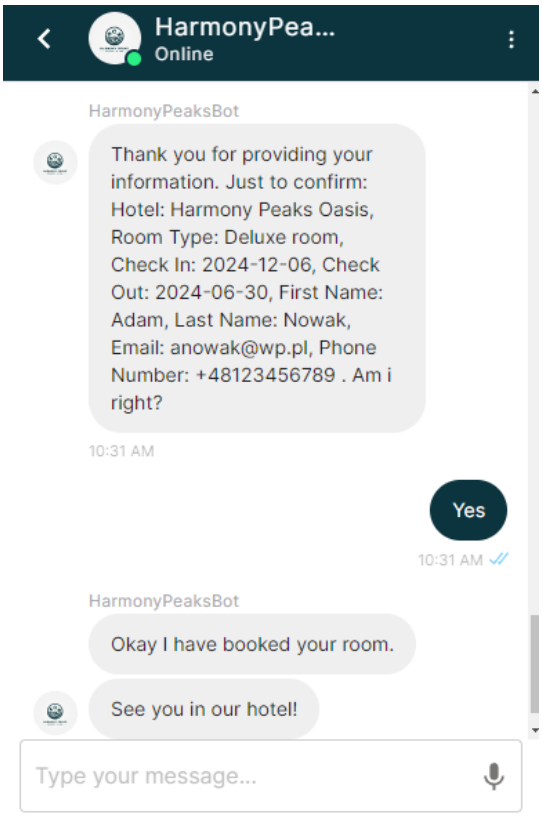
5) Przetwarzanie i Zapisywanie Rezerwacji – Po zweryfikowaniu wszystkich danych, funkcja AWS Lambda przetwarza rezerwację. Proces obejmuje:

- Zapisanie danych rezerwacji w bazie danych: Wszystkie szczegóły rezerwacji są przechowywane w systemie bazy danych hotelu.

6) Generowanie Odpowiedzi i Zamknięcie Rozmowy – Chatbot generuje odpowiedź dla użytkownika, informując go o statusie rezerwacji. Jeśli rezerwacja została dokonana pomyślnie, użytkownik otrzymuje wiadomość potwierdzającą z wszystkimi niezbędnymi szczegółami:

- Nazwa hotelu

- Typ pokoju
- Daty zameldowania i wymeldowania
- Dane klienta



Rys. 3. Poprawnie zakończona rezerwacja pokoju

4. Integracja Chatbota

4.1. Integracja chatbota ze stroną

Aby zintegrować bezserwerowego chatbota, stworzonego przy użyciu Amazon Lex i AWS Lambda, ze stroną internetową hotelu Harmony Peaks, skorzystałem z platformy Kommunicate, która oferuje łatwe i efektywne narzędzia do dodawania chatbota na stronę internetową. Kroki potrzebne do integracji:

1) Utworzenie Konta na Platformie Kommunicate – Pierwszym krokiem jest utworzenie konta na platformie Kommunicate z którego będziemy zarządzać naszym chatbotem.

2) Konfiguracja i integracja Chatbota – Przechodzimy do panelu administracyjnego i podajemy niezbędne dane, takie jak nazwa bota, identyfikatory AWS i inne ustawienia wymagane do połączenia z naszym chatbotem.

3) Dodanie Widżetu Chatbota na Stronę Internetową – Po skonfigurowaniu bota na platformie Kommunicate, otrzymamy fragment kodu JavaScript, który należy dodać do kodu naszej strony internetowej.

```javascript
// Kommunicate settings configuration
var kommunicateSettings = {
  "appId": "2f10426634de874c7b3ca6a55e4c6d8f7", // Kommunicate app ID
  "popupWidget": true, // Enable popup widget
  "automaticChatOpenOnNavigation": false // Disable automatic chat open on navigation
};

// Self-invoking function to load the Kommunicate script
(function(d, m) {
  // Create a new script element
  var s = document.createElement("script");
  s.type = "text/javascript"; // Set the type to JavaScript
  s.async = true; // Load script asynchronously
  s.src = "https://widget.kommunicate.io/v2/kommunicate.app"; // Set the source URL for the Kommunicate script

  // Append the script to the head of the document
  var h = document.getElementsByTagName("head")[0];
  h.appendChild(s);

  // Set the global kommunicate object
  window.kommunicate = m;
  m._globals = kommunicateSettings; // Assign settings to the global kommunicate object
})(document, window.kommunicate || {}); // Pass the document and existing kommunicate object or an empty object

// Event listener for DOM content loaded
document.addEventListener('DOMContentLoaded', function() {
  // Select all elements with the ID 'book-now'
  var bookNowButtons = document.querySelectorAll('#book-now');

  // Add a click event listener to each 'book-now' button
  bookNowButtons.forEach(function(button) {
      button.addEventListener('click', function(event) {
          event.preventDefault(); // Prevent the default action of the button

          // Check if the Kommunicate object and its method 'launchConversation' are available
          if (window.kommunicate && window.kommunicate.launchConversation) {
              window.kommunicate.launchConversation(); // Launch the conversation
          } else {
              console.error('Kommunicate is not loaded yet'); // Log an error if Kommunicate is not ready
          }
      });
  });
});
```

Rys. 4.1. Kod JavaScript do widżetu

4) Personalizacja Widżetu – Platforma Kommunicate pozwala na personalizację widżetu chatbota, aby lepiej pasował do wyglądu strony internetowej hotelu. Możemy zmieniać kolory, logotyp, powitanie oraz inne elementy interfejsu użytkownika

5. Linki

Amazon LEX documentation

Amazon Lambda documentation

Amazon DynamoDB developer guide

Kommunicate documentation

Amazon AWS S3 documentation