AWS Chatbot Challenge

Amazon Lex and AWS Lambda

Benjamin Towner

Cloud Solutions Architect / Engineer

[AWS Chatbot Challenge 2](#_Toc488142952)

[ChatBot Design Concept 3](#_Toc488142953)

[Overview 3](#_Toc488142954)

[Uniqueness 4](#_Toc488142955)

[Design Components 4](#_Toc488142956)

[Contact Management 4](#_Toc488142957)

[Workflow 5](#_Toc488142958)

[Calendar Events 5](#_Toc488142959)

[Models 5](#_Toc488142960)

[Intents 5](#_Toc488142961)

[Lambda 6](#_Toc488142962)

[Testing Framework 7](#_Toc488142963)

[Channels 7](#_Toc488142964)

[Facebook 7](#_Toc488142965)

[Slack 7](#_Toc488142966)

[Twillio 7](#_Toc488142967)

[Entry Requirements 8](#_Toc488142968)

[Demo Video 8](#_Toc488142969)

[Github 8](#_Toc488142970)

[Access to Working Bot 8](#_Toc488142971)

[Testing Instructions 8](#_Toc488142972)

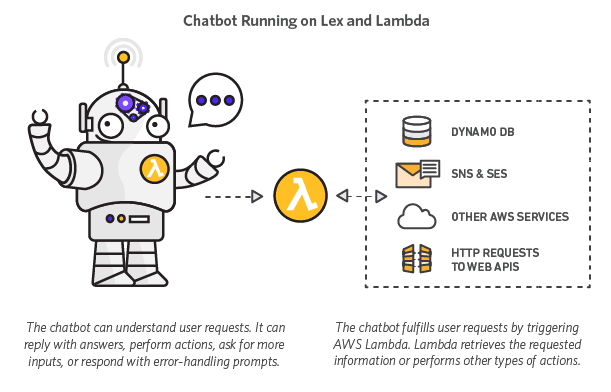
[Text Description 8](#_Toc488142973)

[Devpost Submission 8](#_Toc488142974)

# AWS Chatbot Challenge

Chatbots are changing how companies interface with their customers. With chatbots, you can easily fulfill the needs of your customers in an automated way using natural, human-like chat interfaces. Chatbots serve a variety of use cases, such as customer support, transaction fulfillment, data retrieval, or even DevOps functions (ChatOps).

However, building and running chatbots is a difficult task. First, most developers lack the deep learning expertise necessary to create bots that can intelligently interpret and respond to text. In addition, developers must also provision, manage, and scale the compute resources necessary to run the bot’s code.

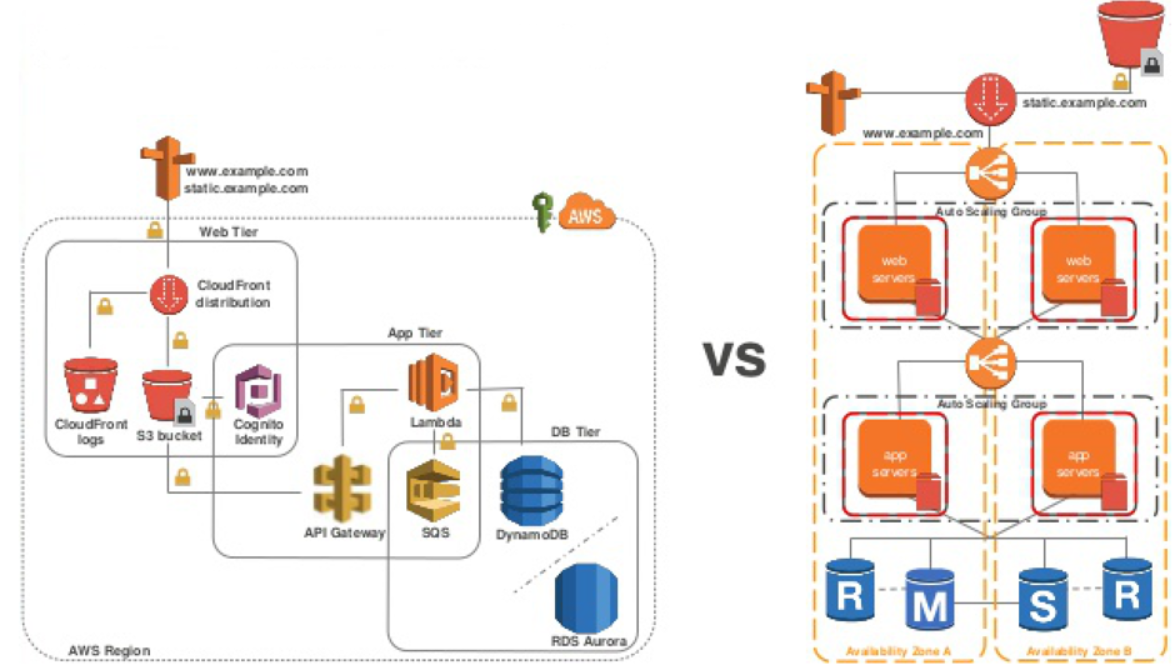


What if you could build chatbots with sophisticated natural language processing and almost no operational overhead? [Amazon Lex](https://aws.amazon.com/lex/) is a fully managed service for building conversational interfaces into any application using voice and text. Lex is powered by the same deep learning technologies that power Amazon Alexa and lets you build natural language chatbots. Lex is integrated with [AWS Lambda](https://aws.amazon.com/lambda/), a service that lets you run code without provisioning or managing servers. Lambda enables you to write and run logic for your chatbot using serverless compute. Getting started with Amazon Lex and AWS Lambda is quick and easy.

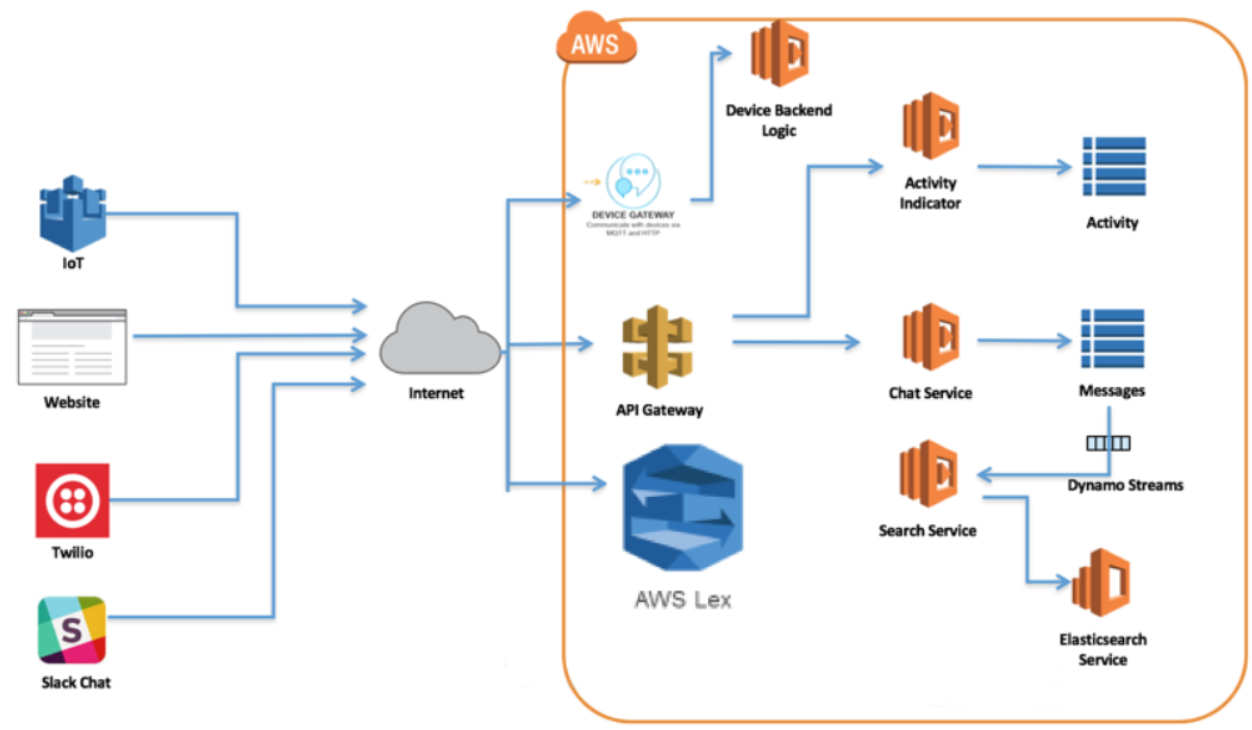
# ChatBot Design Concept

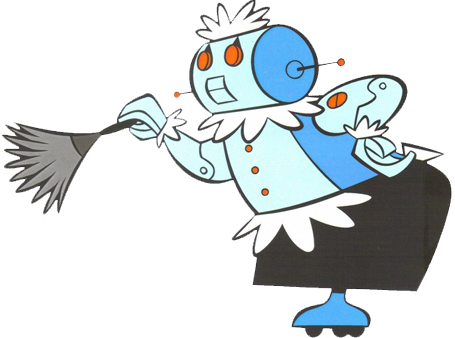
## Overview

AWS PaaS offerings can be compared to the traditional 3 tiered Application designs, with **Bots** replacing the Web tier.



Using the combination of AWS API Gateway, AWS Lambda and AWS DynamoDB services developers can obtain a service-based application back-end that can used to track and control Activity and Messages with data flows in directional or bi-directional manner. Serverless Architecture is described below to illustrate the servicing and integration of various clients with this described backend infrastructure:



**MotherBot** helps by Sharing Calendars, Approved Contacts and provides a self-service administration platform for the *C.E.O. of the Home* to effectively schedule and organize. The AWS Chat Bot allows various processes to be made available to these household with the primary purpose of verifying the identity of the individual connections, formalizing an approval process and managing the ‘feasibility’ on the calendar.

## Uniqueness

Households are like mini-organizations, except their members or ‘***little users***’ often find themselves in a place of constant challenge and response. Seeking approval and then finding out how to facilitate getting it done is the world that ‘***Tweens’*** live in. The coordination required among Parents often requires effective communication practices regardless of your Work-life or Marital status. Technology can enable Parents or Parental guardians to manage the household workloads while providing an appropriate level of privacy and respect.

# Design Components

Reference a traditional 3 tiered Application stack (Web, Application and Data), various components are refactored to assume the functional responsibility.

There are 3 main functional aspects to **MotherBot**.

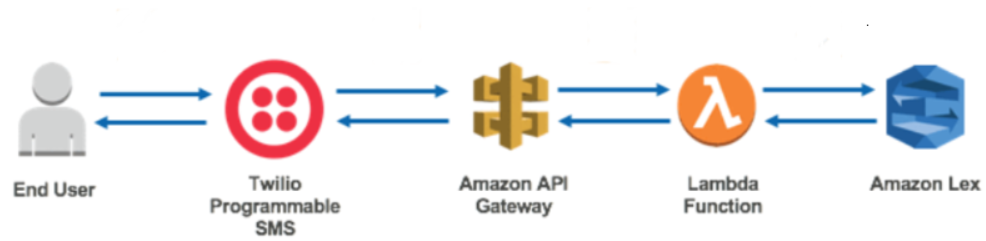
1. Friend’s Identity and Access Management - *Contact Management*
2. Activity Approval - *Workflow*
3. Calendar Feasibility – *Planning*

## Contact Management

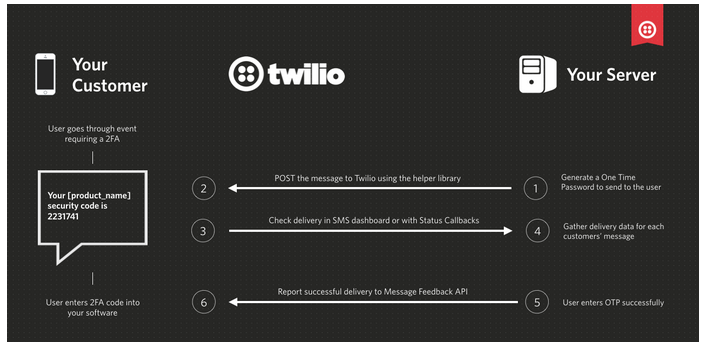
Managing the contacts that are made are very important in a digital age. This is for protection, but also as a measure of maintaining contact information. There is often a need to share contact information within the private networks that Households create.

**MotherBot** can be introduced to a friend which begins this Functional workflow. It is similar to performing a 2 form factor of authorization, by authenticating the new contact by gaining a phone number as a by-product. This allows for a contact number for each of the ‘friends’ that are included in the direct calls or attended events.

**Phone Verification**, using Twillio, is used to make sure people are who they say they are when signing up as a ‘***Friend****’*. It reduces risk and prevents fraudulent signups and increases trust. Phone verification events are best implemented when the user is giving you their number for the first time. When a user triggers that event, [generate a One Time Passcode (OTP)](https://en.wikipedia.org/wiki/One-time_password#Methods_of_generating_the_OTP) and send it via [SMS to the user](https://www.twilio.com/docs/api/rest/sending-messages) Present the user with a UI to enter the code and verify the code against the one you generated to see if they match.



<https://aws.amazon.com/blogs/ai/integrate-your-amazon-lex-bot-with-any-messaging-service/>



## Workflow

Information management about the households Who, Where and What is the primary goal of an activity approval workflow. It helps gather the details that are needed to make a decision. Once decisions are made Logistics can be considered, and sometimes the logistics dictates whether and Activity can be approved.

Activity Approval starts after approved events, locations and authenticated friends have been entered.

## Calendar Events

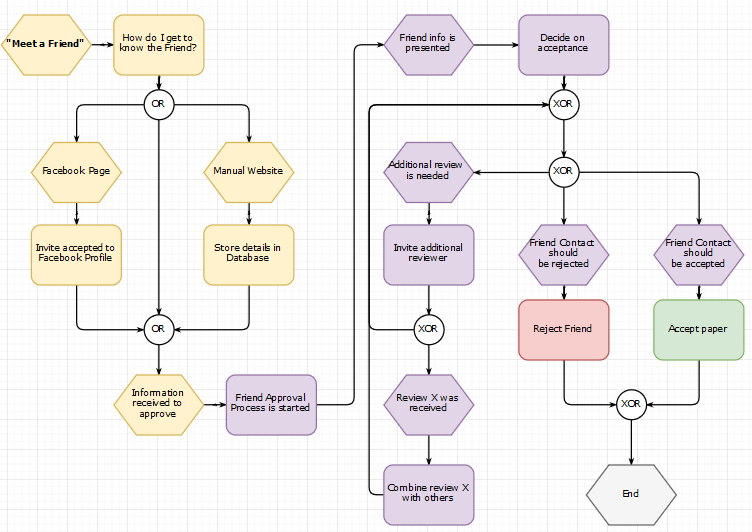
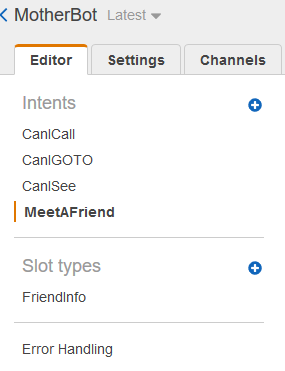
Coordinating the activity workflow is a major undertaking. Similar to Booking Hotels and Cars, accounting for events will help manage the household calendar.

* Arrival Alerts
* Appointment Reminders

# Models

## Intents

* MeetAFriend
  + **Slots:** FriendInfo
  + **Utterances:**
  + **Prompts:**
* CanICall
  + **Slots:** WhoCall
  + **Utterances:**
  + **Prompts:**
* CanIGoTo:
  + **Slots:** ApprovedFriendsHome - Available approved Friend's House
  + **Slots:** ApprovedPublicPlaces – Pool Club, Library, Bowling, Mall
  + **Utterances:**
  + **Prompts:**
* CanISee:
  + **Slots:** Events –
  + **Slots:** Approved Movie
  + **Slots:** Approved Concert
  + **Utterances:**
  + **Prompts:**



## Lambda

Re-using some examples from other blueprints, the attached ’lex-motherbot-python.py’ can be used to create a Lambda function. This function uses the lambda\_handler to route functions based on the intent.

* Blueprints
  + lex-make-appointment
* Response Cards
* Lambda Functions as Code Hooks
* Lex- motherbot -python

def lambda\_handler(event, context):

"""

Route the incoming request based on intent.

The JSON body of the request is provided in the event slot.

"""

# By default, treat the user request as coming from the America/New\_York time zone.

os.environ['TZ'] = 'America/New\_York'

time.tzset()

logger.debug('event.bot.name={}'.format(event['bot']['name']))

return dispatch(event)

|  |  |  |
| --- | --- | --- |
| * + **elicit\_slot**   + **confirm\_intent**   + **close**   + **delegate**   + **build\_response\_card**   + **parse\_int**   + **try\_ex** | * + **increment\_time\_by\_thirty\_mins**   + **get\_random\_int**   + **get\_availabilities**   + **isvalid\_date**   + **is\_available**   + **get\_duration**   + **get\_availabilities\_for\_duration** | * + **build\_validation\_result**   + **validate\_book\_appointment**   + **build\_time\_output\_string**   + **build\_available\_time\_string**   + **build\_options**   + **make\_appointment**   + **dispatch**   + **lambda\_handler** |

# Testing Framework

 Test the Lambda function. You invoke the Lambda function for each of the model uses, using sample data for Adding a Friend, Calling permission, GoTo permission and Going to Events.

1. Choose **Actions, Configure test event**.
2. Choose *Lex-MotherBot MeetAFriend* from the template list.

This sample event matches the Amazon Lex request/response model.

1. Choose **Save and test**.
2. Verify that the Lambda function successfully executed. The response in this case matches the Amazon Lex response model.
3. Repeat the step. This time you choose the *Lex- MotherBot GoTo* from the template list. The Lambda function processes the Event Appointment.

# Channels

Facebook**-**

MotherBot page: <http://fb.me/MotherBot01>

Facebook Messenger: <http://m.me/MotherBot01>

Slack**-**

**Added MotherBot to TEAMGSE**

Twillio**-**

An HTTPS endpoint was created on the AWS API Gateway to interact with Twilio. Text the **MotherBot’s** Twilio Phone Number: (201)431-7268 to access MotherBot by phone

1. Creating an Amazon Lex bot - **MotherBot**
2. Setting up an SMS-enabled Twilio phone number
3. Lambda–the preprocessing layer between Amazon Lex and Twilio
4. Using AWS API Gateway to create an HTTPS endpoint
5. Filling in the blanks on Twilio.

# Entry Requirements

1. A **demo video** that clearly shows your bot functioning on its intended platform.
2. **Share your code** repository publicly or privately through GitHub or BitBucket with testing@devpost.com.
3. **Access to a working bot**.
4. **Testing instructions** with anything we need to know to test your bot.
5. **Text description** - include a brief explanation of what the bot Application does and what makes it unique.
6. A **completed submission form** on Devpost.

## Demo Video

Created using Skype for business, and produce the publish location.

## Github

<https://github.com/bentowner/MotherBot>

## Access to Working Bot

Text Twilio Phone Number: **(201)431-7268**

Access Facebook Page: <http://fb.me/MotherBot01>

Access Facebook Messenger: <http://m.me/MotherBot01>

## Testing Instructions

Follow the workflow from this Document

## Text Description

This document will serve as the Text Description of MotherBot, the Challenge and the Submission.

## Devpost Submission