

Microsoft Azure Chat Bot with LUIS (Cognitive Service)

CIS5850 – GROUP 2

Ashish Solanki
Vinayak Gaur
Shwetha Seetharam
Neel Savla

Introduction

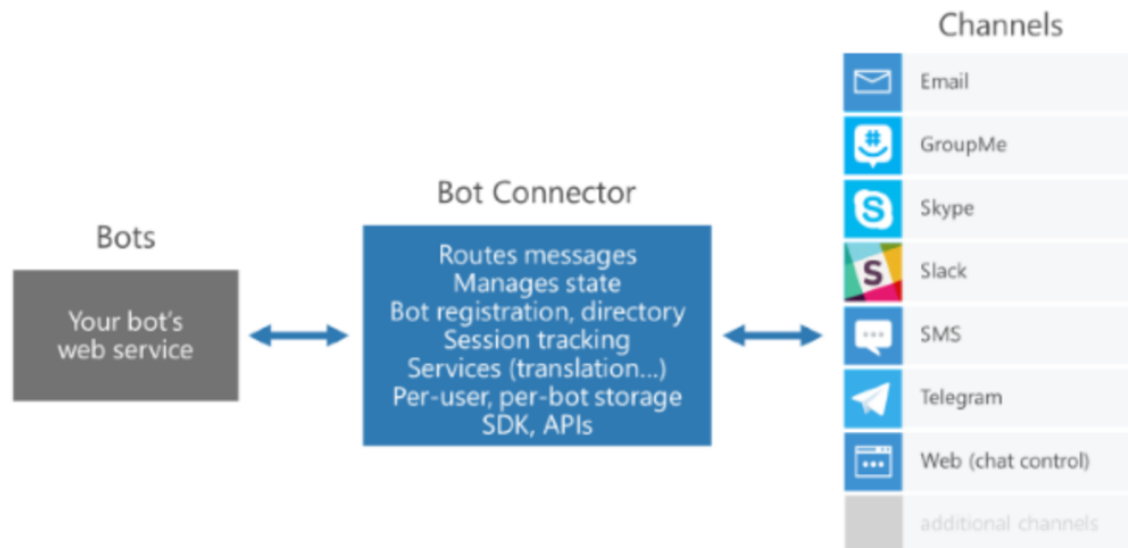


Azure Bot Service is Microsoft's artificial intelligence (AI) chatbot offered as a service on the Azure cloud service marketplace.

It offers the ability to add intelligent agents that are capable of conversation without having to commit the resources to develop one's own AI. The service can be added to websites, apps, email, GroupMe, Facebook Messenger, Kik, Skype, Slack, Microsoft Teams, Telegram, SMS, Twilio, Cortana and Skype for Business.

Along with cognitive services, Azure Bot service offers some excellent services including Language translation, recognizing users from pictures etc.

Bot Architecture



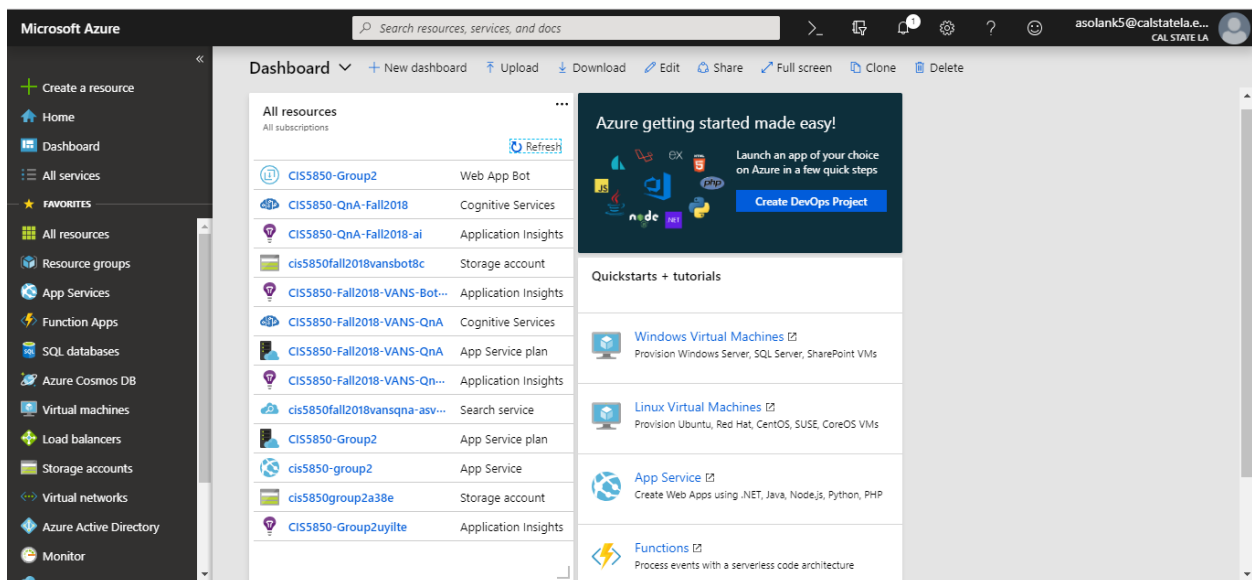
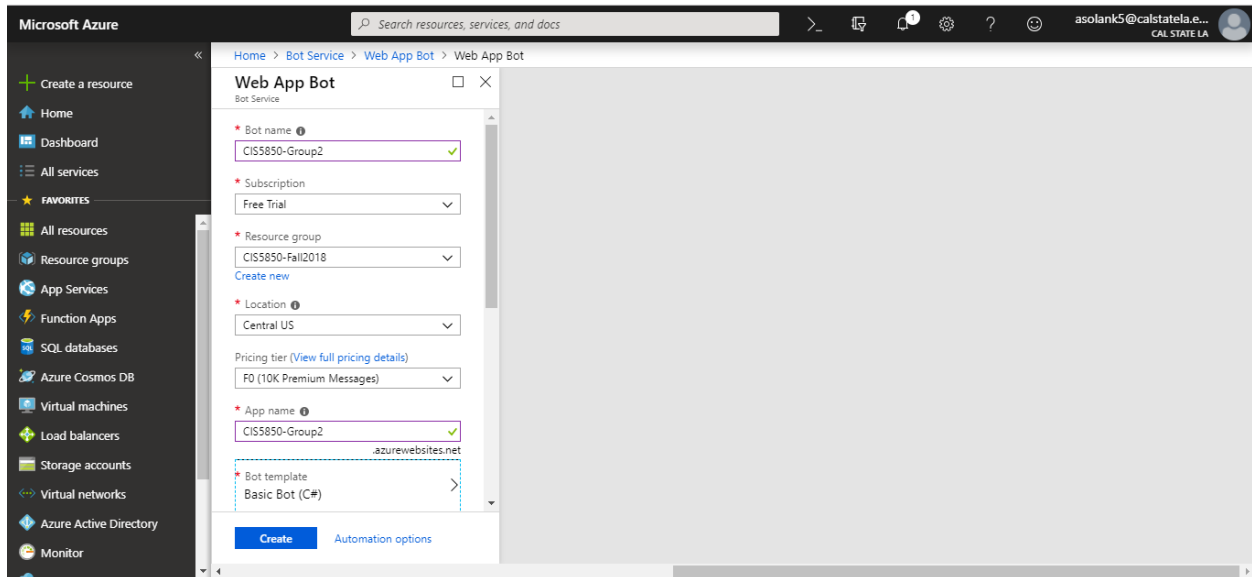
Practical Applications

As Bots provide an experience that feels less like using a computer and more like dealing with a person - or at least an intelligent robot, It has various practical applications such as.

- Health Care Assistant
- Multi-Media Marketing
- Smart Trip Assistant
- Customer Support
- Personal Assistance

Creating Azure Bot Service

Step 1: Creating Chat Bot from Microsoft Azure portal



CIS-Group2 is the name of Chat Bot created. Resource Group selected is Javascript and Resource Location is West US.

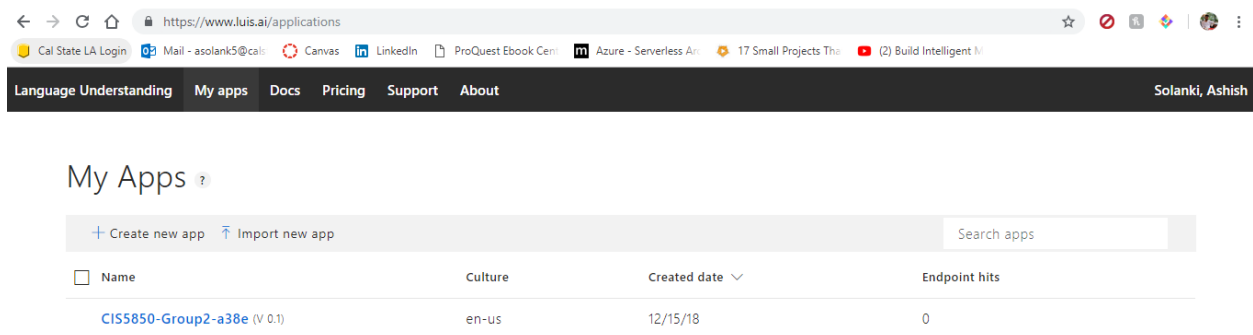
In this project we have created a Microsoft Azure Chat Bot which and a Cognitive Service (LUIS) which helps us to understand and create a conversational Chat Bot.

LUIS App

Step 2: Up until here, we have created successfully Azure Chat Bot. Now to make this bot interactive, we need an AI based Cognitive Service App. Language Understanding Integration App.

Create a Cognitive Service to interact with the Chat bot. This service primarily serves the purpose of storing utterances and answers to user based on the Questions asked in Chat bot

Designed to identify valuable information in conversations, LUIS interprets user goals (intents) and distills valuable information from sentences (entities), for a high quality, nuanced language model. LUIS integrates seamlessly with the Azure Bot Service, making it easy to create a sophisticated bot.



<input type="checkbox"/>	Name	Culture	Created date	Endpoint hits
<input type="checkbox"/>	CIS5850-Group2-a38e (V 0.1)	en-us	12/15/18	0

Step 3: Creating Intents and Utterances in LUIS App to create a conversational Bot with Azure

Utterances are input from the user that your app needs to interpret. To train LUIS to extract intents and entities from them, it's important to capture a variety of different inputs for each intent. Active learning, or the process of continuing to train on new utterances, is essential to machine-learned intelligence that LUIS provides.

The screenshot displays the LUIS application interface for an app named 'CIS5850-Group2-a38e (V0.1)'. The user is logged in as 'Solanki, Ashish'. The interface includes a top navigation bar with links for 'Language Understanding', 'My apps', 'Docs', 'Pricing', 'Support', and 'About'. Below this, there are tabs for 'DASHBOARD', 'BUILD' (which is active), and 'MANAGE', along with buttons for 'Train', 'Test', and 'Publish'. A green notification banner at the top states 'Publishing complete. Refer to the list of endpoints to access your endpoint URL'. On the left, a sidebar menu shows 'Intents' and 'Entities' under the 'Intents' section, and 'Improve app performance' with sub-options like 'Review endpoint utterances', 'Phrase lists', and 'Patterns'. The main area is titled 'Greeting' and contains a text input field with the placeholder 'Type about 5 examples of what a user might say and hit Enter'. Below the input field, there are 'Entity filters' and a toggle for 'Show All' and 'Entities View'. A table lists five utterances with their corresponding labeled intents:

Utterance	Labeled intent ?
userLocation	Greeting (0...)
my location is userLocation	Greeting (0...)
userLocation is my current city	Greeting (0...)
my name is userName	Greeting (1...)
you can call me userName	Greeting (1...)

Language Understanding My apps Docs Pricing Support About Solanki, Ashish

CIS5850-Group2-a38e (v 0.1) DASHBOARD BUILD MANAGE Train Test Publish

✓ Publishing complete. Refer to the list of endpoints to access your endpoint URL

Entity Inters Show All Entities view

Utterance	Labeled intent ?
i 'm lost	Help (0.931) ...
i don 't understand	Help (0.842) ...
confused	Help (0.905) ...
can you help me	Help (0.966) ...
why doesn 't this work	Help (0.874) ...
what can you help me with	Help (0.982) ...
how do i interact with you	Help (0.889) ...
frustrated	Help (0.905) ...

In this way numerous utterances can be created and tested and trained on LUIS portal before publishing it and using it on Chat Bot.

Step 4: Integrating LUIS app service to the Web Chat Bot using LUIS App ID

CIS 5850_Project_Report.docx x Settings - Microsoft Azure x LUIS: App General Settings x Integrating Microsoft LUIS into ti x +

https://www.luis.ai/applications/090bc756-d3b7-4542-bfe8-806783cbfbf5/versions/0.1/manage/general

Cal State LA Login Mail - asolank5@cali Canvas LinkedIn ProQuest Ebook Cent Azure - Serverless An 17 Small Projects Tha (2) Build Intelligent I

Language Understanding My apps Docs Pricing Support About Solanki, Ash

CIS5850-Group2-a38e (V 0.1) DASHBOARD BUILD MANAGE Train Test Publish

Application Settings

Application Information

Keys and Endpoints

Publish Settings

Versions

Collaborators

Application Information

Application ID

090bc756-d3b7-4542-bfe8-806783cbfbf5

Display Name (Required)

CIS5850-Group2-a38e

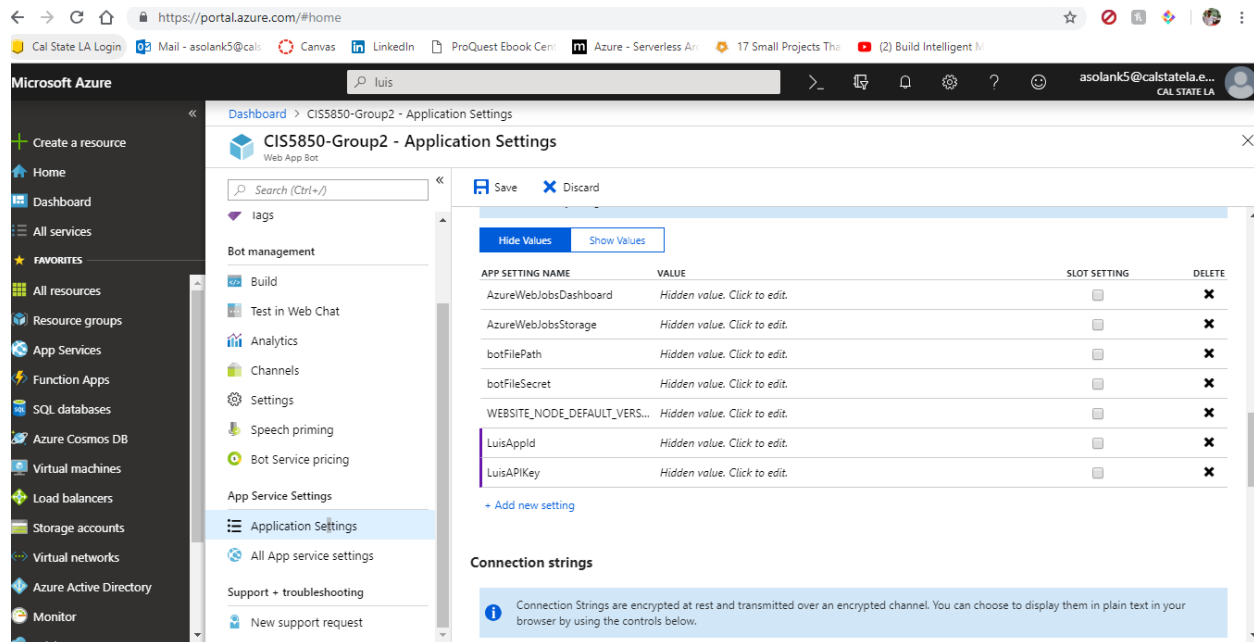
Description

A BotBuilder generated bot

Culture: en-us

Make this app public so that non-contributors can query this app with a valid key.

11:16 AM

Step 5: Enter the copied LUIS App ID key to the App settings of Chat Bot

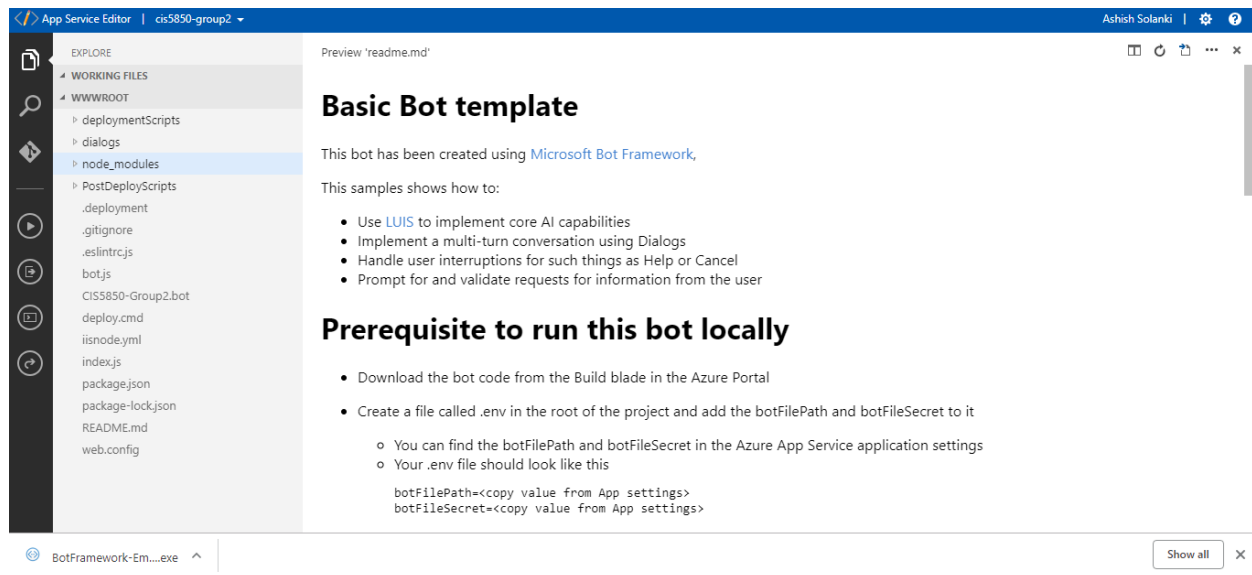
The screenshot shows the Microsoft Azure portal interface. The left sidebar contains navigation options like 'Create a resource', 'Home', 'Dashboard', 'All services', and 'FAVORITES'. The main content area is titled 'CIS5850-Group2 - Application Settings' and displays a table of application settings. The 'LuisApiKey' is highlighted in the table. Below the table, there is a section for 'Connection strings' with a note about encryption.

APP SETTING NAME	VALUE	SLOT SETTING	DELETE
AzureWebJobsDashboard	Hidden value. Click to edit.	<input type="checkbox"/>	✕
AzureWebJobsStorage	Hidden value. Click to edit.	<input type="checkbox"/>	✕
botFilePath	Hidden value. Click to edit.	<input type="checkbox"/>	✕
botFileSecret	Hidden value. Click to edit.	<input type="checkbox"/>	✕
WEBSITE_NODE_DEFAULT_VERSION	Hidden value. Click to edit.	<input type="checkbox"/>	✕
LuisAppId	Hidden value. Click to edit.	<input type="checkbox"/>	✕
LuisAPIKey	Hidden value. Click to edit.	<input type="checkbox"/>	✕

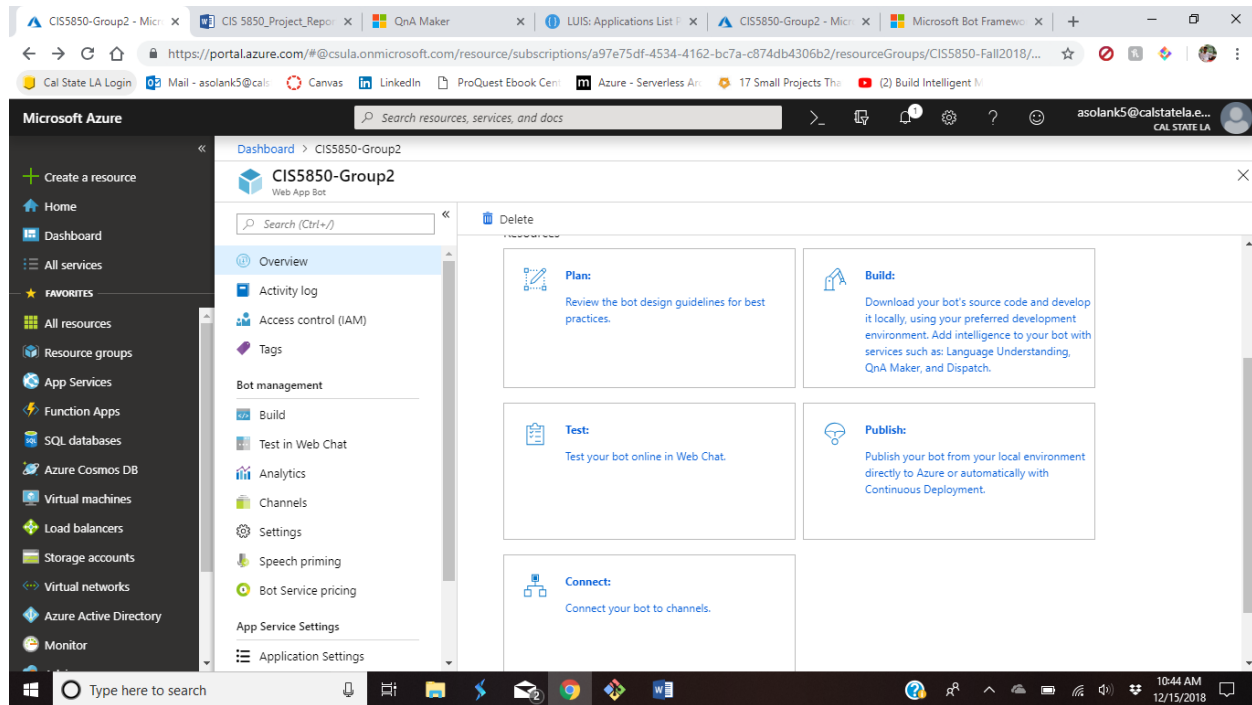
+ Add new setting

Connection strings

Connection Strings are encrypted at rest and transmitted over an encrypted channel. You can choose to display them in plain text in your browser by using the controls below.

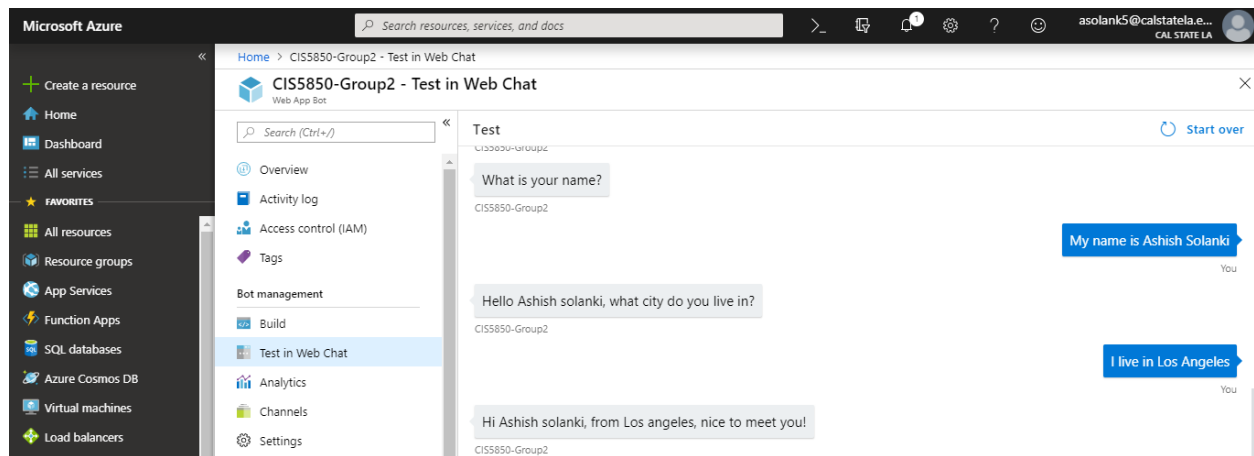
Step 6: Customize Chat Bot using Online Code Editor in Azure Chat Bot

Step 7: Testing the Chat Bot service



Step 8: Adding Utterances

We have added several utterances such as “What is your name?”, “What city do you live in?” in the chatbot for testing as well as for building it for practical implementation.



Limitations

Beside the successful testing of the chatbot, we still deal some of the limitations as follows:

- QnA service: To add more question and answer to support on the bot, we need to create the utterance for questions and answer but QnA maker on MS Azure is paid service, so we can not use this for further development.
- Continuous Development Project: As Chat bot is a continuous development project, it can be built smarter as per the need.