Course link: <https://www.udemy.com/course/microsoft-bot-framework-v4-nodejs-with-real-life-example/>

Graphical user interface, text, application

Description automatically generated

Diagram

Description automatically generated

Resources required:

Graphical user interface

Description automatically generated

**Use case description below:**

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Tools required:

Graphical user interface, application, table

Description automatically generated with medium confidence

**Steps summary**

1: Create a "Language Understnding" authoring instance on Azure portal

2. Create LUIS app on LUIS portal (where we can define intents, entities etc)

3. Create a QnA instance on Azure portal(Knwledge base, which will expose an endpoint URL).

4. Next on QnA.io portal we create a actual KB where we define our Quesitons and Answeers

5. Next we create a Bot via 'yo' and 'generator-botbuilder'

Aftre installation run command : 'yo botbuilder'

This will create a Node based Bot solution (Boilerplate)

6. To work with dailogs you need **botbuilder-dialogs** pn package

npm i botbuilder-dialogs

As a first step: In Azure Go to “Language Understanding” and create a new instance:

Herre we are creating a LUIS app on Azure

Language understanding (LUIS) is a natural language processing service that enables you to build your own custom model to understand human language programmatically **or through the UI in the LUIS portal**. After you are satisfied with your LUIS model, you publish it and query it's prediction endpoint through your client application for an end to end conversational flow. **To build, manage, train, test and publish your LUIS Model, you will need to create the below Authoring Resource**. This also gives you 1,000 requests/month endpoint requests. If you want your client app to request **beyond the 1,000 requests provided by the authoring, create the below Prediction Resource**. If you know from the start you will be needing more than 1000 prediction requests as well as the authoring experience, create using the "Both" option. This will create two resources, one for each type.

A screenshot of a computer

Description automatically generated with medium confidence

Once app is created, we can use its keys:

A screenshot of a computer

Description automatically generated

LUIS has a separate portal too - <https://www.luis.ai/>

Graphical user interface, text, application, email

Description automatically generated

Creating a new LUIS app on LUIS portal

Graphical user interface, application

Description automatically generated

Above one is the app created on LUIS portal (NOT azure portal). Here you will define Intents, Entities which are used to create NLP Model.

Prediction resource created on Azure portal will reflect here in LUIS portal automatically. Choose the same Prediction resource created earlier while creating LUIS instance on Azure portal

* **Guidline to create LUIS model below:**

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

* Now after creating LUIS service on LUIS portal, we will now **create QnA maker** **service** on Azure portal

Graphical user interface, text

Description automatically generated

We are creating a QnA maker. Notice we are using the same Resource grp here.

Once created it will expose an API end point and keys to access

A screenshot of a computer

Description automatically generated with medium confidence

Now go to : <https://www.qnamaker.ai/> and sign in

On selecting the subscription, this portal will reflect the Knowledge Base created earlier on Azure portal

Graphical user interface, text, application, Word

Description automatically generated

Below are the steps on <https://www.qnamaker.ai/> to create a Knowledge base

Graphical user interface, text, application, email

Description automatically generated

Since we have already created a QnA service on Azure portal, we will skip Step 1

Now in Step 3, it will ask for Knowledge Base name you want to keep and also will to **populate the KB.**

**For sample,** You can use Google policy link: [**https://policies.google.com/faq**](https://policies.google.com/faq)as **shown below:**

Graphical user interface, text, application, email

Description automatically generated

Nest you create your KB (Knowledge base). Final screen looks like below:

Graphical user interface, text, application, email, website

Description automatically generated

Next press “**Save and train**” button

You can add more QnA from the Edit page as shown below:

Graphical user interface, text, application, website

Description automatically generated

Now you can test the QnA

Graphical user interface, application

Description automatically generated

Next you can Publish - once you publish the KB, in  **settings**  it will give you an endpoint to use to use Knowledge base. See below:

Graphical user interface, text, application, email

Description automatically generated

We have got the endpoint on publishing.

**Azure Bot Service**- Tools and Services which helps in interating the different components provided by Bot framework to test, build and manage your bot.

Bot service Procvides quick start templates to start building Bots quickly. Also provides template for

1. Cognitive service like LUIS and QnA maker
2. To deploy Bot to Azure

Graphical user interface, text, application, email

Description automatically generated We will be using Yeoman

**Creating EchoBot**

Step 1:We will install ‘yo’ and ‘generator-botbuilder’

<https://www.npmjs.com/package/generator-botbuilder>

Step 2: Use the botbuilder to create a new Echo Bot

Command: **yo botbuilder**

It will create a folder which will have entire code for Bot and deployment templates too that you can directly use in Azure as shown below:

Table

Description automatically generated with medium confidence

Now to run the bot solution you can run command :  **nom run start**  from the root of the new directly if above created bot

Text

Description automatically generated

Now go to Bot Emulator (installed previosuly)

And create a new bot configuration as shown below:

Graphical user interface, application

Description automatically generated

In the configuration window, give any name to the bot and in the URL, use the same port as it was displayed in the above screen when we run the source code i.e. 3978 with a default route of **api/messages/**  as shown below.

A screenshot of a computer

Description automatically generated with medium confidence

If your bot is hosted on Azure provide Azure UserId and Pasword:

Graphical user interface, application

Description automatically generated

Above file will give you a configuration file(**\*.bot**) which you can resuse.

One configuration file is set and saved, it will apear automatically in Emulator Home page like below:

Graphical user interface, application

Description automatically generated

**Our Echo bot is ready.**  Echoing the same message what we are typing

Graphical user interface, application, Teams

Description automatically generated

* **Bot Terminologies**

Diagram

Description automatically generated

**Channels-** like WebChat, facebook messagner, Skype, etc.

**Activity** can be text message, speech message, attachment, etc

**How it works**

Activity’s json payload is received by Bot which runs a **restify** **web server**(npm package in Node solution). **Restify deserailizes the JSON payload** and send the deserialized payload to **Adapter**. See how adaptor is used in Node solution.

A screenshot of a computer

Description automatically generated with medium confidence

Adaptor will than process the payload (by calling ‘processActivity’ method of adapter) and that creates a ‘**Turn Context’**out of it and than sends this activity to **ActivityHandler.** See picture below

Adapter will then call Main Dailog’s run method (Main Dialog is the Bot object in our case). This Bot object extends ‘ActivityHanlder’ class. See picture below

**Turn Context** gives the information like ‘Sender Info’, ‘Receiver Info’, Channel Info to the bot. Bot needs to know where the info is coming from and where it needs sent the respond to , etc.

Turn is just ow people talk to each other to make his/her own argument. E.g. in above diagram A1-A6 is turn

A screenshot of a computer

Description automatically generated with medium confidence

Text

Description automatically generated

Actvity Handler is a class provided by Bot Builder and uses an EventEmitter and listener pattern. E,g. onMessage() to register an event listener for a message type activity.

All activity types are listed here:

<https://docs.microsoft.com/en-us/azure/bot-service/bot-activity-handler-concept?view=azure-bot-service-4.0&tabs=csharp>

* **Understanding ActiivtyHandler**

Diagram

Description automatically generated

Now we first modiify MemberAdded activity type method in bot class to change the welcome message and also send some **suggestedActions** as shown below:

Text

Description automatically generated

**OnMembersAdded** is called when the chat window is first opened.See how it looks in action in emulator below:

**Check above in code base**

Graphical user interface, application

Description automatically generated

Notice that the username is coming as ‘User’ and it can be checked, it is coming from Activity JSON in Emulator Window as shown below:

Graphical user interface, text, application

Description automatically generated

* **Types of Dialogs**

We have a few different dialogs: **Prompts**, **Waterfall dialogs**and**Component dialogs**

**Prompts**

Prompts are used to ask the user for input and wait until the user enters input.

There are different prompts available, and among those:

* Choice Prompt: Asks for a choice from a set of options
* Text prompt: Asks for general text input
* Confirm prompt: Asks for a confirmation.
* Number prompt: Asks for a number.
* OAuth Prompt: Asks the user to sign in using the Bot Frameworks Single Sign On (SSO) service

**Waterfall dialogs**

A waterfall dialog is composed of a sequence of steps. Each step of the conversation is implemented as an asynchronous function that takes a waterfall step context parameter. Usually, in each step, we prompt the user for input(for instance a question), which then the user can respond to.

**Component dialogs**

ComponentDialogs are containers that encapsulate multiple sub-dialogs, but can be invoked like normal dialogs. This is useful for re-usable dialogs, or creating multiple dialogs with similarly named sub-dialogs that would otherwise collide. A **component** could be added as a **dialog** to another **ComponentDialog** or **DialogSet**.

**What is a Dialog Set**

DialogSet is a container for multiple dialogs. Once added to a DialogSet, dialogs can be called and interlinked.

* **Working with Component Dialogs**

We are going to create a Reservation dialogue next which will be Component Dialog.

Waterfall dialog is a type of Componnet Dialog

Our Waterfall dialog will contain a number of stepswhich will run in a sequence when the dialog is triggered and each step will contain some prompt. We are going to use a number prompt, text prompt and a confirmation prompt. Than we will also validate our prompt. We will take one example of number prompt and than try to validate the value entered by the user if the value entered by the user is in a particular range.

* To work with dailogs you need **botbuilder-dialogs** pn package

npm i botbuilder-dialogs

A picture containing diagram

Description automatically generated

**Dialogs are key concepts in Bot Framework SDK** and is provided by **botBuilder-dialog** module

Dialogs can be thought of as structures in your bot that can act as function in your Bot’s program and each dialog is designed to perfrm a specific task.In this case we are going to make reservation for a user.

There can be many dialogs and you can specfy the order of dialogs to guide the conversation and you can invoke them in different ways.

* **How Dialog States are stored**

Graphical user interface, diagram

Description automatically generated

When messageActivity lands on our bot from Azure Bot service, the bot will check our Storage layer for already saved DialogState for that particlar conversation and if a Dialog Stateis present it will be sent to the Bot and than process the content as per the content of Dialog State which was saved from the previous step.

InMemory is not recommended for Production since this type of Stprage is volatile and has limitation of **32KB of data.**

* **Making Hotel Reservation Bot**

**Step 1: Index.ts file changes**

* For in-memory, we have added **MemoryStorage class** in Index.js file

const { BotFrameworkAdapter, MemoryStorage } = require('botbuilder');

const memoryStorage = new MemoryStorage();

We also add **ConversationState** and **UserState** in **Index.ts** file

const { BotFrameworkAdapter, MemoryStorage, ConversationState, UserState } = require('botbuilder');

**Why User and Conversation State?**

Without ConversationState, bot wont know what to do when a new activity comes in. When a user sends in message, our dialog wont know what to do, where to send that message and how to process without a Conversation State.

* Then we create objects of all 3 in **Index.ts** file only as shown below:

const memoryStorage = new MemoryStorage();

const conversationState = new ConversationState(memoryStorage);

const userState = new UserState(memoryStorage);

* Now we pass them to mainBot in index.ts file:

// Create the main dialog.

const restaurantBot = new RestaurantBot(conversationState, userState);

**Step 2: Bot File code** (Code is explained via comments in Bot file (myRestaurantBot.js), so refer it)

Here we created propertyAccessors(named ‘dialogState’) and then in onMessage() handler we handled the routing logic(which Dialog to be invoked based on the incomng message)

Diagram

Description automatically generated

We are going to present a Welcome Dialog to the user and user joins the conversation, If user selects ”Make Reservation” , then we will start MakeReservation Waterfall dialog and user will redirected to it.If the waterfall dialog has ended than again welcome message will be displayed and this goes in a loop.

**Step 3: Added Component Dialog** (file named ‘makeReservationDailog’ in solution)