



2016 Banking and Financial Markets Industry Workshop

Implementing a chat bot using Watson Cognitive Services

Hands-on Lab

The lab is divided into three parts

Part A: Getting started

Estimated Duration: 10 minutes

Part B: Understand Watson Conversation and add features to the bot

Estimated Duration: 30 minutes

Part C: Use Watson Tone Analyzer to detect emotions in conversations

Estimated Duration: 10 minutes

Table of Contents

PART A: GETTING STARTED	3
1. WHAT YOU NEED.....	3
2. WHAT YOU WILL BE BUILDING	3
3. SETUP THE CHAT BOT USING WATSON CONVERSATION.....	4
PART B: UNDERSTAND WATSON CONVERSATION AND ADD NEW BOT FEATURES	16
1. INTERACT WITH THE CHAT APPLICATION.....	16
2. ADD INTENTS AND ENTITIES.	20
3. UNDERSTAND DIALOG FLOWS, NODES AND CONTEXT VARIABLES.....	25
PART C: USE WATSON TONE ANALYZER TO DETECT EMOTIONS IN CONVERSATIONS.....	32
1. OVERVIEW OF THE TONE ANALYZER SERVICE FOR EMOTION DETECTION	32
2. VIEW HOW THE SERVICE IS INVOKED IN THE CHAT APPLICATION	33
3. UNDERSTAND HOW TONE ANALYZER RESULTS ARE USED TO ALTER DIALOG FLOWS	34

Part A: Getting started

1. What you need

- a) **A Bluemix account** - to create a Watson Conversation service, Watson Tone Analyzer service and access JazzHub (DevOps)
- b) **A laptop with a web browser** - to work with Bluemix
Note: Google Chrome or Firefox browser is recommended
- c) **Internet connectivity**

Pre-requisites setup:

The pre-requisite steps will guide you through the setup of a Bluemix account, setup of a JazzHub ID and deployment of the Chat bot application with the associated Watson services.

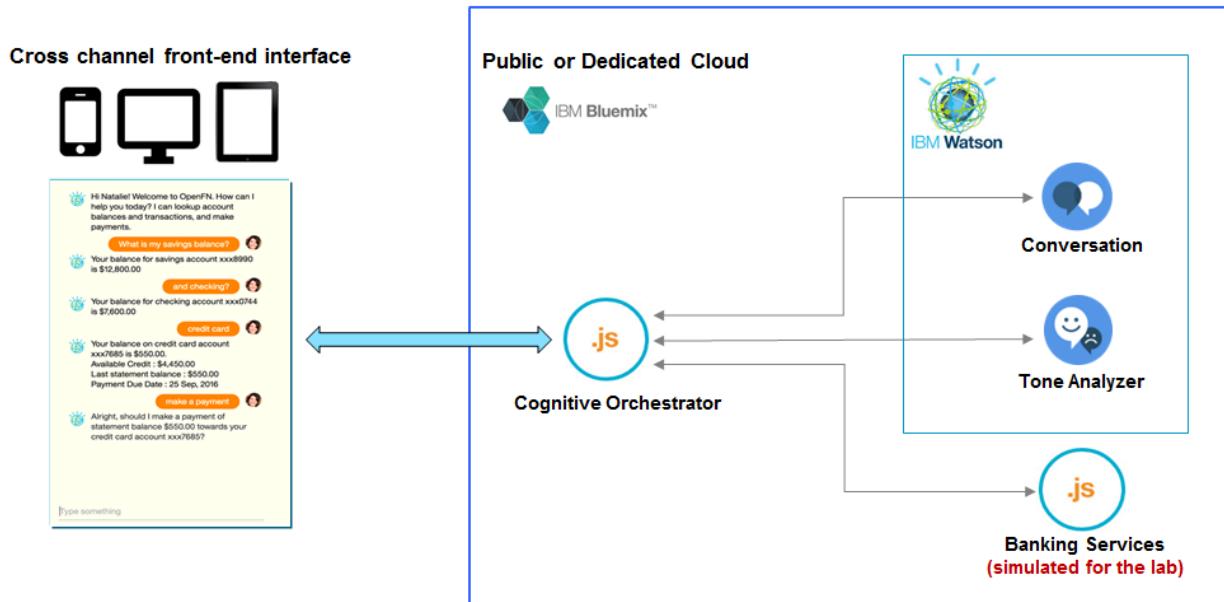
If you have not completed the pre-requisites, then ask your lab instructors for availability of temporary accounts to use during this lab (**limited availability**).

If you have completed the pre-requisites for this lab OR you have received a temporary Bluemix account/password from your lab instructors, then proceed to **step 2 (What you will be building)**.

Link to the pre-requisite : <http://ibm.biz/cognitiveprep>

2. What you will be building

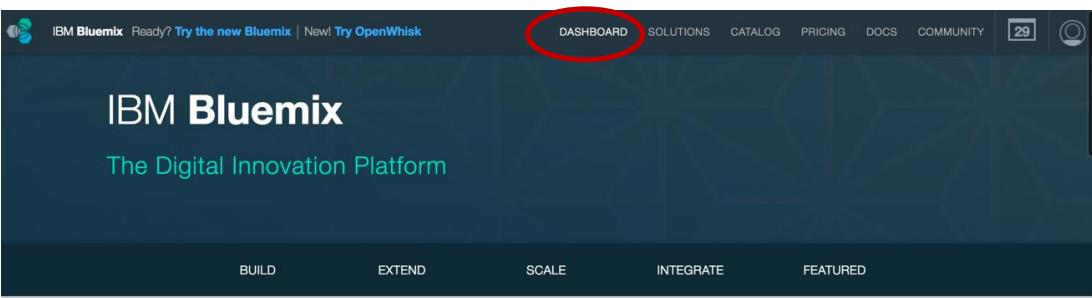
You will be building a chat application for retail banking that recognizes and manages common interactions between the bank's customer and the back-end systems via natural language conversations. The application personalizes conversations using customer profile and account information, defines responses in a Watson Conversation dialog and detects emotions like 'anger' with Watson Tone Analyzer, to direct the consumer from the chat application to a live agent if the need arises. The main components are depicted in the architecture diagram below.



The chat application's user interface (front end UI) interacts via REST APIs with a Cognitive Orchestrator service developed using NodeJS. Since the service is exposed via REST APIs, the front end interface can be implemented across multiple channels including smartphones, tablets and desktop browsers and applications. The purpose of the Cognitive Orchestrator is to request data from external systems such as Banking services (simulated for this lab) and Watson Tone Analyzer service, and act as a liaison between these systems and Watson Conversation. It retrieves data from external systems and provides this information to Watson Conversation to evaluate user input statements and personalize the responses. The Cognitive Orchestrator also complements the responses from Watson Conversation by providing data lookup functions when structured data needs to be returned to the front end interface of the chat application.

3. Setup the chat bot using Watson Conversation

- Login to your [Bluemix](#) account and click on the **Dashboard** tab.



The screenshot shows the top navigation bar of the new IBM Bluemix interface. The 'DASHBOARD' tab is circled in red. Other tabs include 'SOLUTIONS', 'CATALOG', 'PRICING', 'DOCS', and 'COMMUNITY'. Below the navigation, the text 'IBM Bluemix' and 'The Digital Innovation Platform' is displayed. A large banner at the bottom says 'Build your apps, your way.' with three options: 'Instant Runtimes', 'IBM Containers', and 'Virtual Servers'.

Note – If you land on the new Bluemix interface as shown below, please click on the icon on the top-right  and then click 'Switch to Classic'. You may be asked to login again. After login, you will land on the Bluemix Dashboard.

2016 Banking and Financial Markets Industry Workshop

The screenshot shows the IBM Bluemix New Console interface. At the top right, there's a user profile for 'gsclab 1234'. Below it, settings for 'Region: US South', 'Organization: gsclab1234', and 'Space: dev' are displayed. A 'Manage Organizations' link and a 'Create a Space' button are also present. On the left, a sidebar lists 'Cloud Foundry Applications' and 'Services'. The main area shows a table for 'Cloud Foundry Applications' with one entry: 'gscfss-lab-chatbot' with route 'gscfss-lab-chatbot.mybluemix.net', memory 256 MB, 1 instance, and 1 running. Below this is a section for 'Services' with 2/10 used. On the right, there's a 'Support' section with links for 'Get help' and 'Submit an Idea', and a 'Console Option' section with a 'Switch to Classic' button, which is circled in red.

- b) Click on the Chat application icon for an Overview of the Chat application **chatbot-xxxx-xxxx**. The Overview page displays the current application's runtime state and the services bound to the application (Watson Conversation and Tone Analyzer).
- Note – The Banking services component shown previously in the architecture diagram has been simulated in this lab's demo and is packaged within the same runtime Node SDK for simplification.**

The screenshot shows the IBM Bluemix Classic Dashboard. The top navigation bar includes 'IBM Bluemix', 'Ready? Try the new Bluemix | New! Try OpenWhisk', and tabs for 'DASHBOARD', 'SOLUTIONS', 'CATALOG', 'PRICING', 'DOCS', and 'COMMUNITY'. The left sidebar shows 'ORG: gsclab1234' and a dropdown for 'dev' space, with options for 'CF APPS (1)', 'SERVICES (2)', 'CONTAINERS (0)', and 'VIRTUAL SERVERS (0)'. The main area features sections for 'WORK WITH DATA' (Data Analytics) and 'USE SERVICES OR APIs' (Services API). Below these are sections for 'Applications' and 'Services'. The 'Applications' section contains a card for 'GSC-FSS-Lab-Chatbot' with the URL 'gscfss-lab-chatbot.mybluemix.net'. This card is circled in red. The 'Services' section shows two service instances. The entire dashboard has a dark theme.

2016 Banking and Financial Markets Industry Workshop

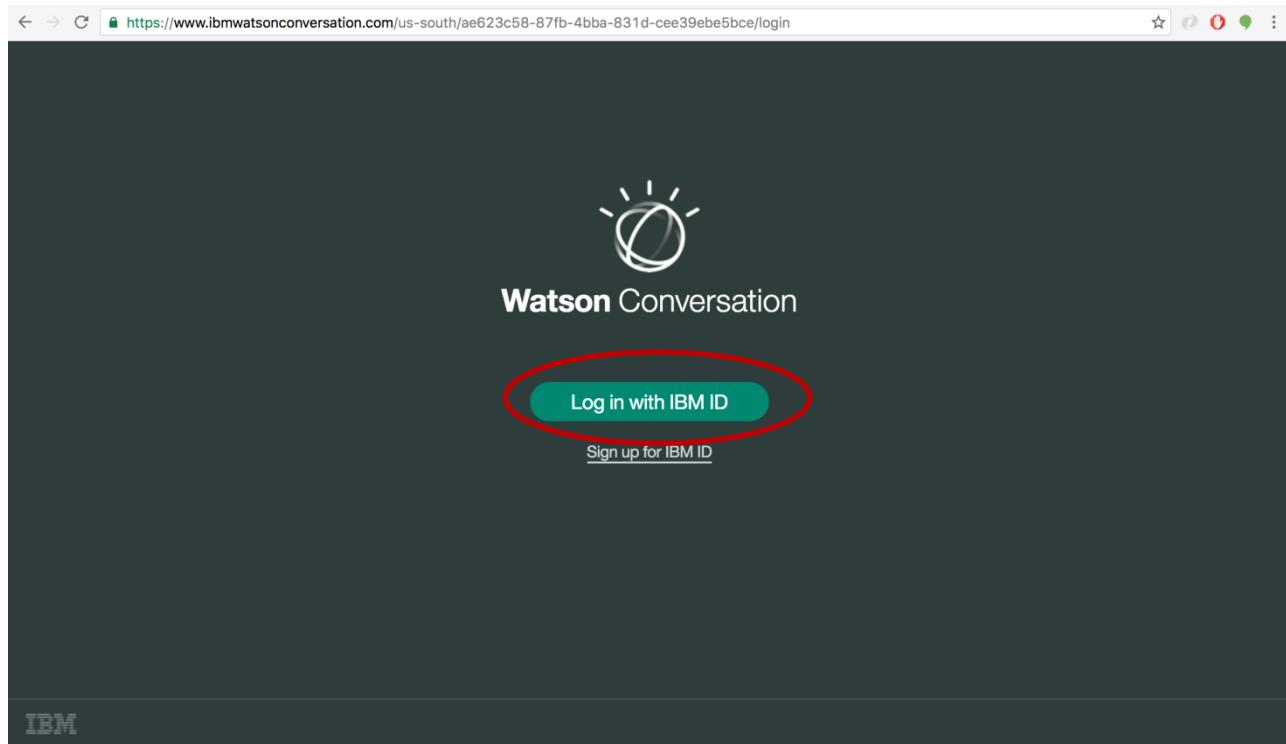
- c) Click on the Watson Conversation service tile and then click on the **Launch Tool** button.

The screenshot shows the IBM Bluemix dashboard for the 'GSC-FSS-Lab-Chatbot' application. On the left sidebar, under 'SERVICES', there are links for 'Conversation' and 'Tone Analyzer'. The main area displays the 'GSC-FSS-Lab-Chatbot' app details, including its icon, name, routes, instances (1), memory quota (256 MB), and available memory (1.750 GB). Below this, there are two sections: 'ADD A SERVICE OR API' and 'BIND A SERVICE OR API'. A red circle highlights the 'ADD A SERVICE OR API' section, specifically the 'Conversation' tile. To the right, there is an 'APP HEALTH' section showing the app is running, and an 'ACTIVITY LOG' section with recent logs from August 19, 2016.

The screenshot shows the Watson Conversation service page. The left sidebar includes 'Manage', 'Service Credentials', and 'Service Access Authorization'. The main content area features a large 'Conversation' heading with a purple speech bubble icon. It describes adding a natural language interface to applications for automating interactions with end users. Below this, there are sections for 'Conversation tooling' (with a red circle around the 'Launch tool' button) and 'Intended Use'. A sidebar on the right lists 'Developer resources' with links to 'Documentation' and 'Demo'.

2016 Banking and Financial Markets Industry Workshop

- d) If you get a page to Login shown below, then click **Log in with IBM ID** to proceed.



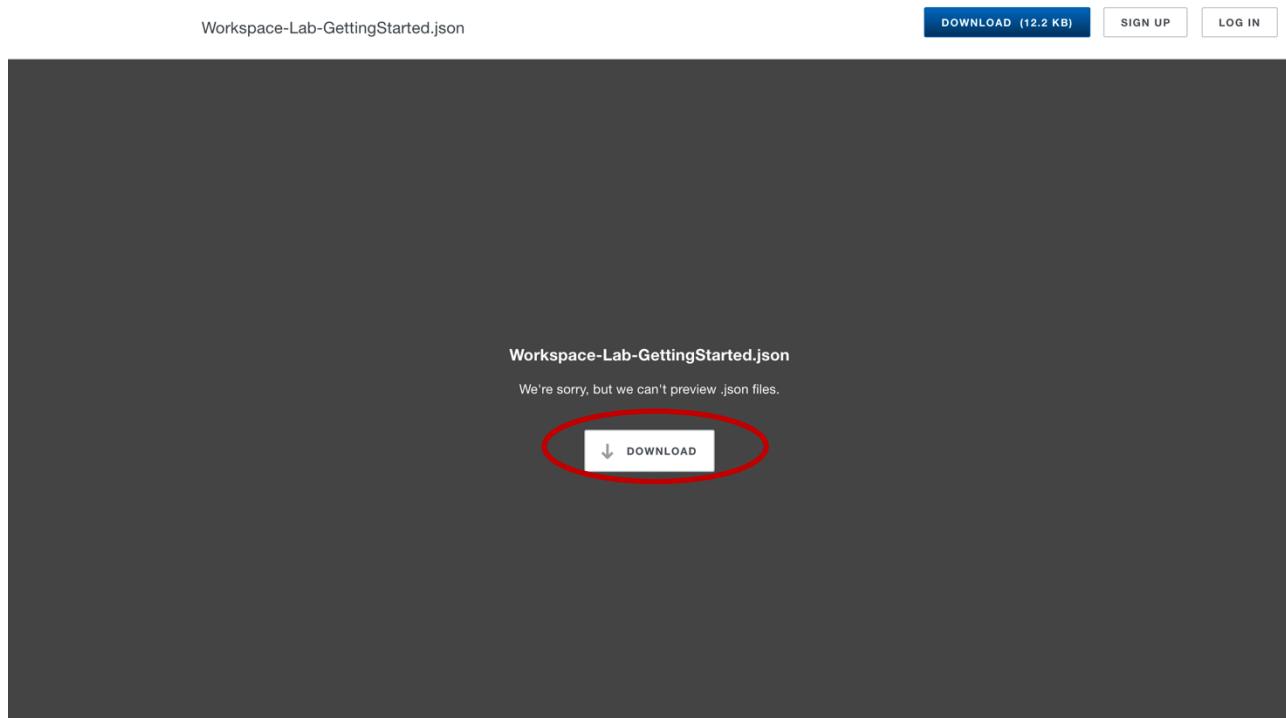
- e) The initial page displays an overview of workspaces in your Watson Conversation service. You can create new workspaces or import existing workspaces using JSON files.

A screenshot of the Watson Conversation service interface. At the top, it says 'Watson Conversation' and 'Instances: GSC-FSS-Lab-Conversation'. The main area shows a grid of workspace icons. One icon is highlighted with a purple dashed circle. Below the grid, there's a 'Create workspace' button and a description: 'Workspaces enable you to maintain separate intents, user examples, entities, and dialogs for each use or application.' At the bottom, there are 'Create' and 'Import' buttons.

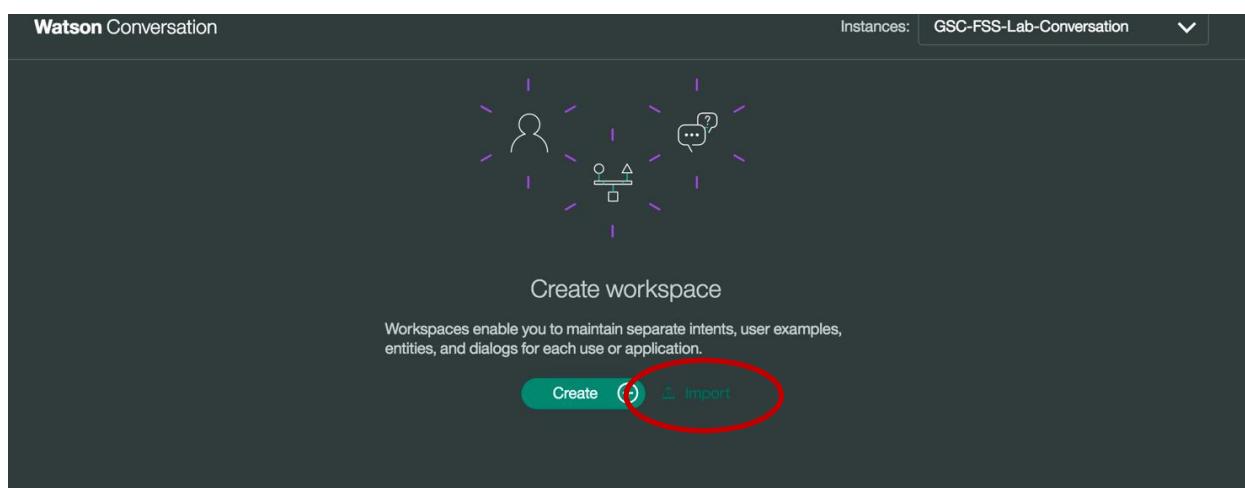
2016 Banking and Financial Markets Industry Workshop

- f) Click on the link below to download a JSON file that provides the initial setup of a Watson Conversation chat. Note the location where this file has been downloaded.

Link to download JSON : <http://ibm.biz/gscchatbot>

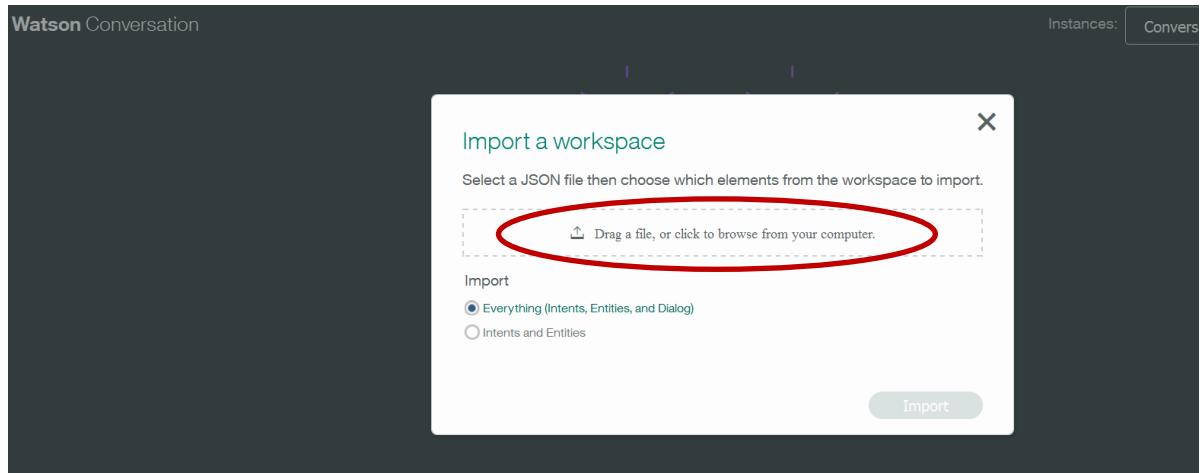


- g) Return to the Watson Conversation workspace overview and click the **Import** button.

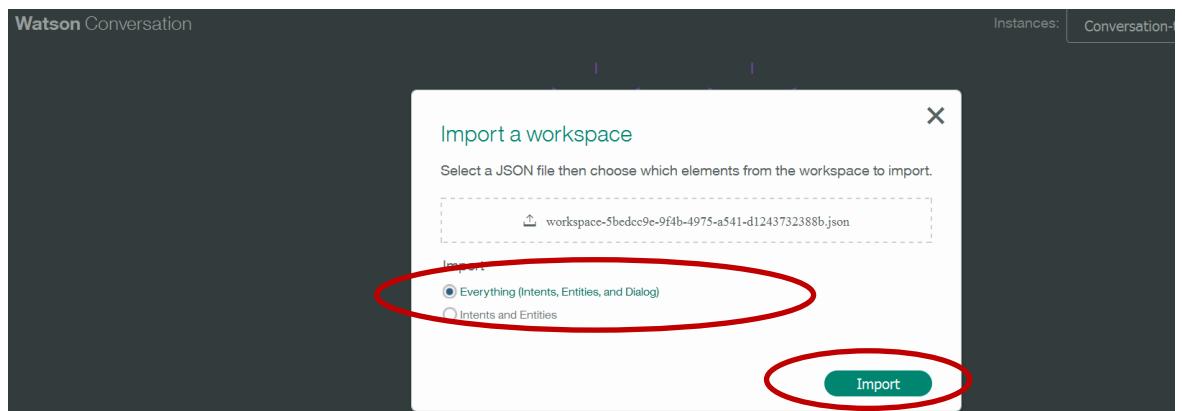


2016 Banking and Financial Markets Industry Workshop

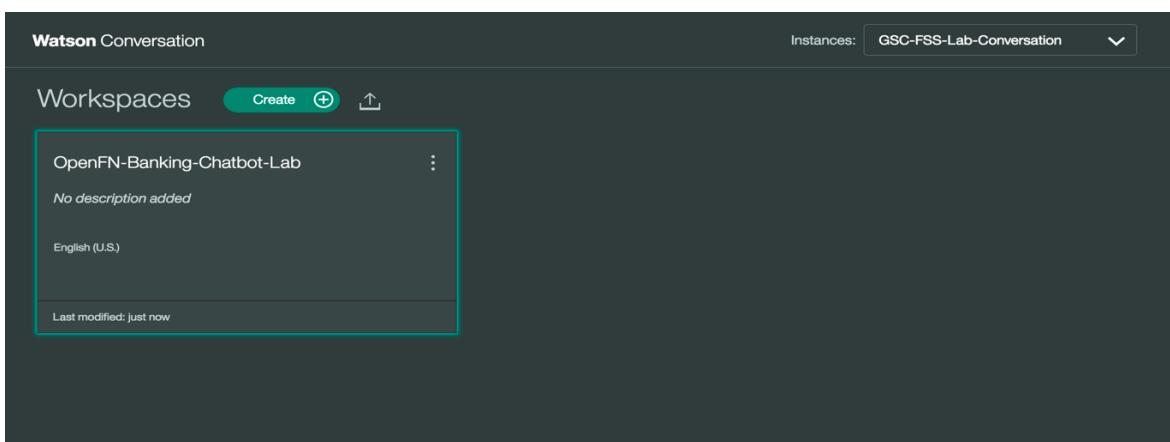
- h) In the **Import a workspace** popup, click the **browse from your computer** option and select the JSON file downloaded in the previous step.



- i) Ensure that the import **Everything (Intents, Entities and Dialog)** radio button is selected and then click the **Import** button.

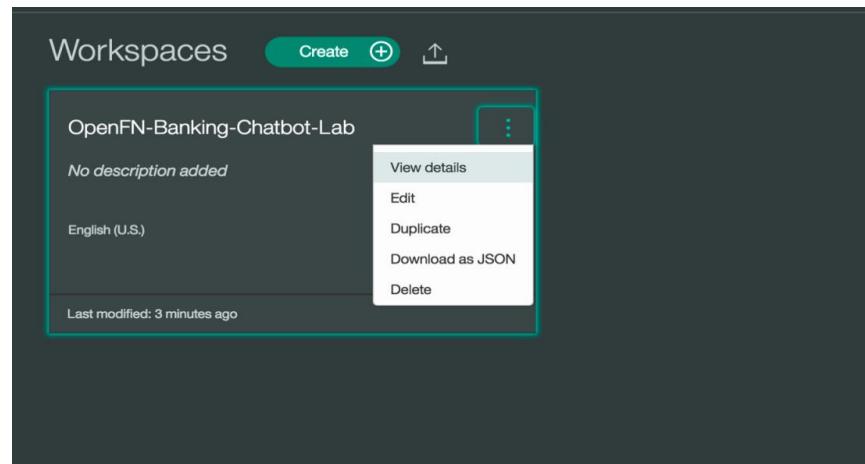


- j) When the import completes, the new workspace will be displayed.

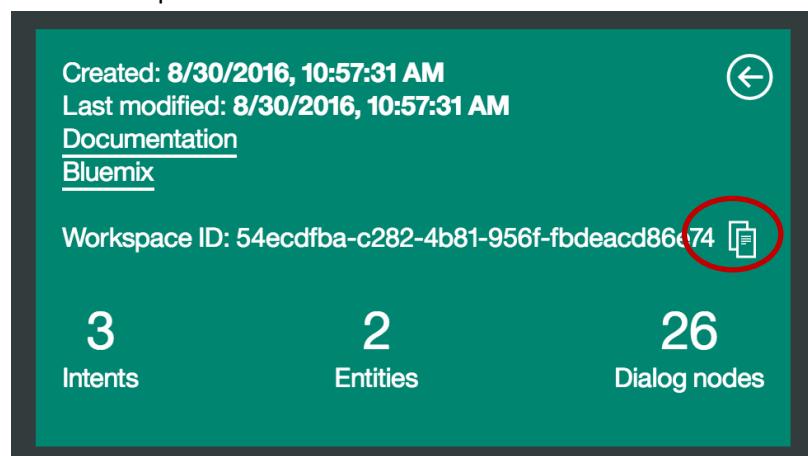


2016 Banking and Financial Markets Industry Workshop

- k) Click the options menu (three vertical dots) and select **View Details**. This will display the workspace ID that uniquely identifies the workspace instance within your Watson Conversation service.



- l) Copy the workspace ID by clicking on the button highlighted below OR select and copy (Ctrl+C or command+C) the 36 character workspace ID value.



2016 Banking and Financial Markets Industry Workshop

- m) Return to the Bluemix Dashboard and click on the chat application's tile to view the Overview page.

The screenshot shows the IBM Bluemix Dashboard. On the left, there is a sidebar with 'ORG: gsclab1234' and a dropdown menu for 'dev' containing 'CF APPS (1)', 'SERVICES (2)', 'CONTAINERS (0)', and 'VIRTUAL SERVERS (0)'. The main area is titled 'Data & Analytics' and has two buttons: 'WORK WITH DATA' and 'USE SERVICES OR APIs'. Below this, there is a section for 'Applications' which lists the 'GSC-FSS-Lab-Chatbot' application. The app card shows its icon (a robot head), name, URL ('gsclfss-lab-chatbot.mybluemix.net'), and status ('Not running'). A red circle highlights this application card. At the bottom of the dashboard, there are sections for 'Services' and some icons.

- n) Click on **Environment Variables**.

The screenshot shows the 'GSC-FSS-Lab-Chatbot' application overview page. On the left, there is a sidebar with 'Back to Dashboard...', 'Overview', 'SDK for Node.js™', 'Files', 'Logs', 'Environment Variables' (which is circled in red), and 'Start Coding'. Under 'SERVICES', there are 'Conversation' and 'Tone Analyzer'. The main area shows the application's configuration: 'INSTANCES: 1', 'MEMORY QUOTA: 256 MB per Instance', and 'AVAILABLE MEMORY: 1.750 GB'. It also features sections for 'ADD A SERVICE OR API' and 'BIND A SERVICE OR API'. Two service cards are shown: 'Conversation' (free) and 'Tone Analyzer' (standard). On the right, there is an 'APP HEALTH' section indicating the app is running, an 'ACTIVITY LOG' showing recent events, and a button to 'Estimate the cost of this app'.

2016 Banking and Financial Markets Industry Workshop

- o) Select **USER-DEFINED** tab

The screenshot shows the IBM Bluemix interface for a project named "GSC-FSS-Lab-Chatbot". On the left, there's a sidebar with options like Overview, SDK for Node.js™, Files, Logs, Environment Variables (which is currently selected), Start Coding, SERVICES (Conversation, Tone Analyzer), and a GitHub icon. The main area is titled "Environment Variables" and contains tabs for "VCAP_SERVICES" and "USER-DEFINED", with the latter being highlighted by a red oval. Below the tabs is a JSON configuration block:

```
{
  "conversation": [
    {
      "name": "GSC-FSS-Lab-Conversation",
      "label": "conversation",
      "plan": "free",
      "credentials": {
        "url": "https://gateway.watsonplatform.net/conversation/api",
        "password": "REDACTED",
        "username": "REDACTED"
      }
    }
  ],
  "tone_analyzer": [
    {
      "name": "GSC-FSS-Lab-Tone-Analyzer",
      "label": "tone_analyzer",
      "plan": "standard",
      "credentials": {
        "url": "https://gateway.watsonplatform.net/tone-analyzer/api",
        "password": "REDACTED",
        "username": "REDACTED"
      }
    }
  ]
}
```

- p) Click **ADD**.

The screenshot shows the same "Environment Variables" page as before, but now the "USER-DEFINED" tab is active. The main area displays a table with columns for "Name", "Value", and "Actions". A message at the bottom says "No items to display". At the bottom right, there are four buttons: "ADD", "SAVE", "RESET", and "EXPORT", with the "ADD" button being circled in red.

2016 Banking and Financial Markets Industry Workshop

- q) Enter **WORKSPACE_ID** in the **Name** field. Paste (Ctrl+V or command+V) the value of the workspace ID previously copied in step (l) into the **Value** field as shown below.

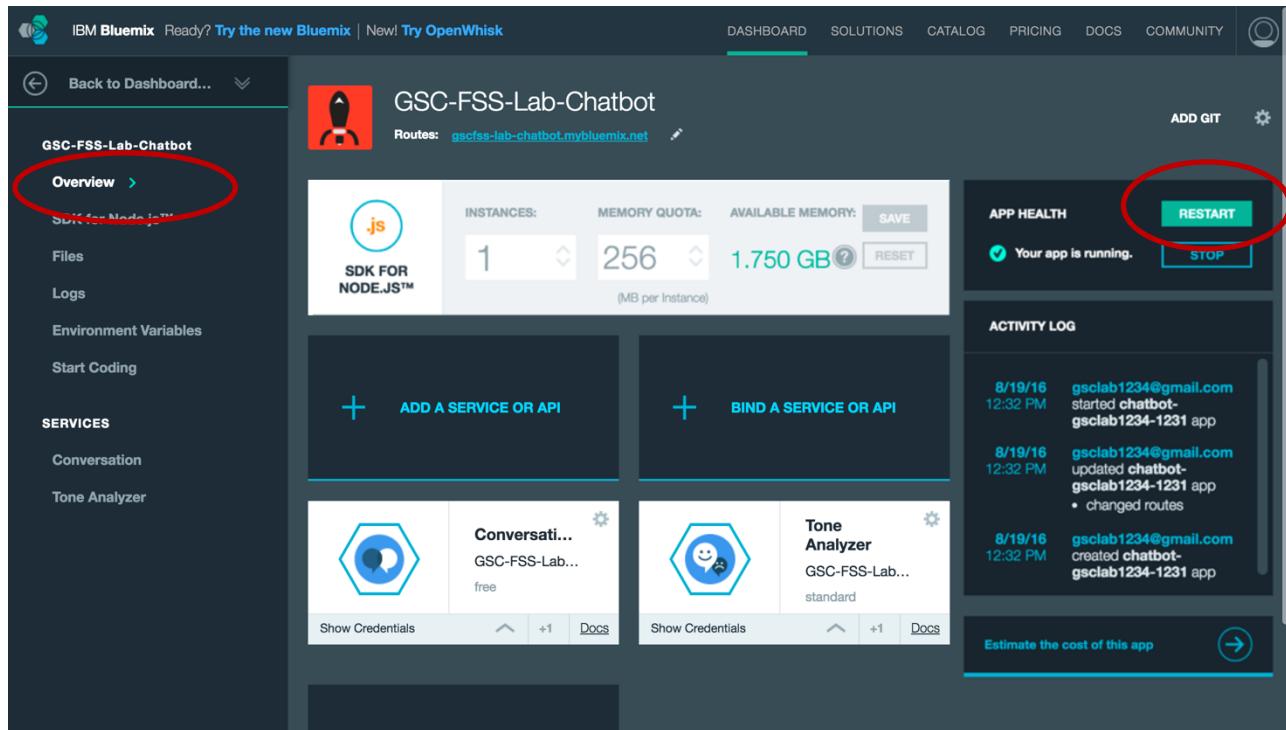
The screenshot shows a web-based interface for managing environment variables. At the top, there's a navigation bar with a 'js' icon and the title 'Environment Variables'. Below this is a table with columns for 'Name', 'Value', and 'Actions'. A single row is present in the table, containing the value 'WORKSPACE_ID' in the 'Name' column and 'e25291a5-4d6b-45b5-91fd-3ee7d328a2e5' in the 'Value' column. Both the 'Name' and 'Value' fields are circled in red. The 'Actions' column contains a delete icon. At the bottom of the table, a message says 'There are unsaved changes.' followed by four buttons: 'ADD', 'SAVE' (which is highlighted in blue), 'RESET', and 'EXPORT'.

- r) Click **SAVE**.

This screenshot shows the same environment variables page after the changes have been saved. The table now displays a single row with the 'Name' 'WORKSPACE_ID' and 'Value' 'e25291a5-4d6b-45b5-91fd-3ee7d328a2e5'. The 'Actions' column still has a delete icon. At the bottom, the message 'There are unsaved changes.' is still present, but the 'SAVE' button is no longer highlighted; instead, the 'RESET' button is circled in red. The other buttons ('ADD', 'EXPORT') are also visible.

2016 Banking and Financial Markets Industry Workshop

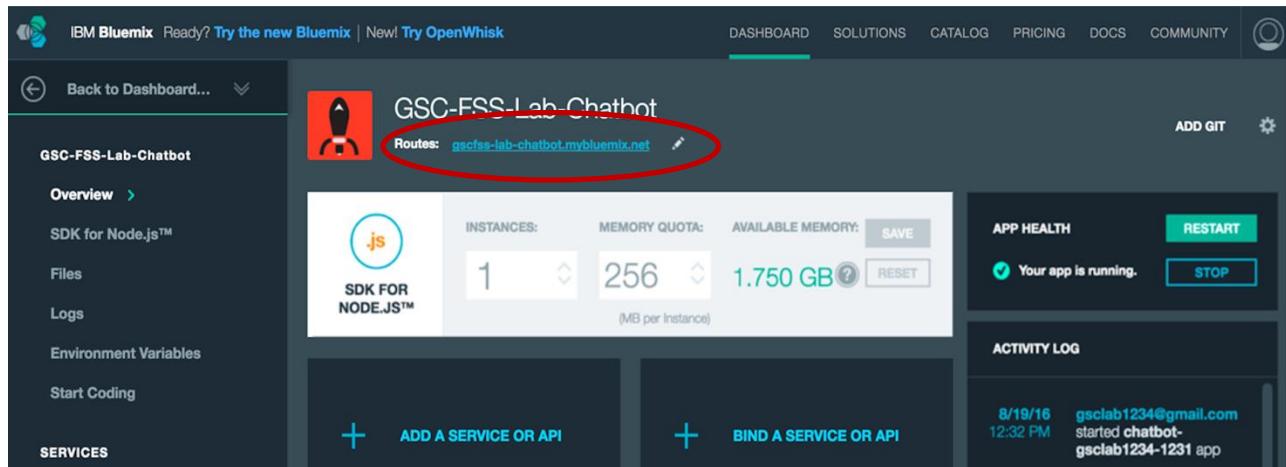
- s) Return to the **Overview** page of the chat application and click **RESTART**. The application may take a couple of minutes to finish restarting.



The screenshot shows the IBM Bluemix dashboard with the application 'GSC-FSS-Lab-Chatbot' selected. The left sidebar has a red circle around the 'Overview' link. The main panel displays the application's configuration, including instances (1), memory quota (256 MB), and available memory (1.750 GB). Below this are sections for adding services or APIs and binding services or APIs. Two service cards are shown: 'Conversation' (free) and 'Tone Analyzer'. The right side features an 'APP HEALTH' section with a green status message ('Your app is running') and a red circle around the 'RESTART' button. An 'ACTIVITY LOG' section shows recent events, and a footer button says 'Estimate the cost of this app'.

- t) After the restart has completed, your Watson Conversation service's workspace ID will be associated with your chat application instance. Click on the link besides **Routes** to open your chat application.

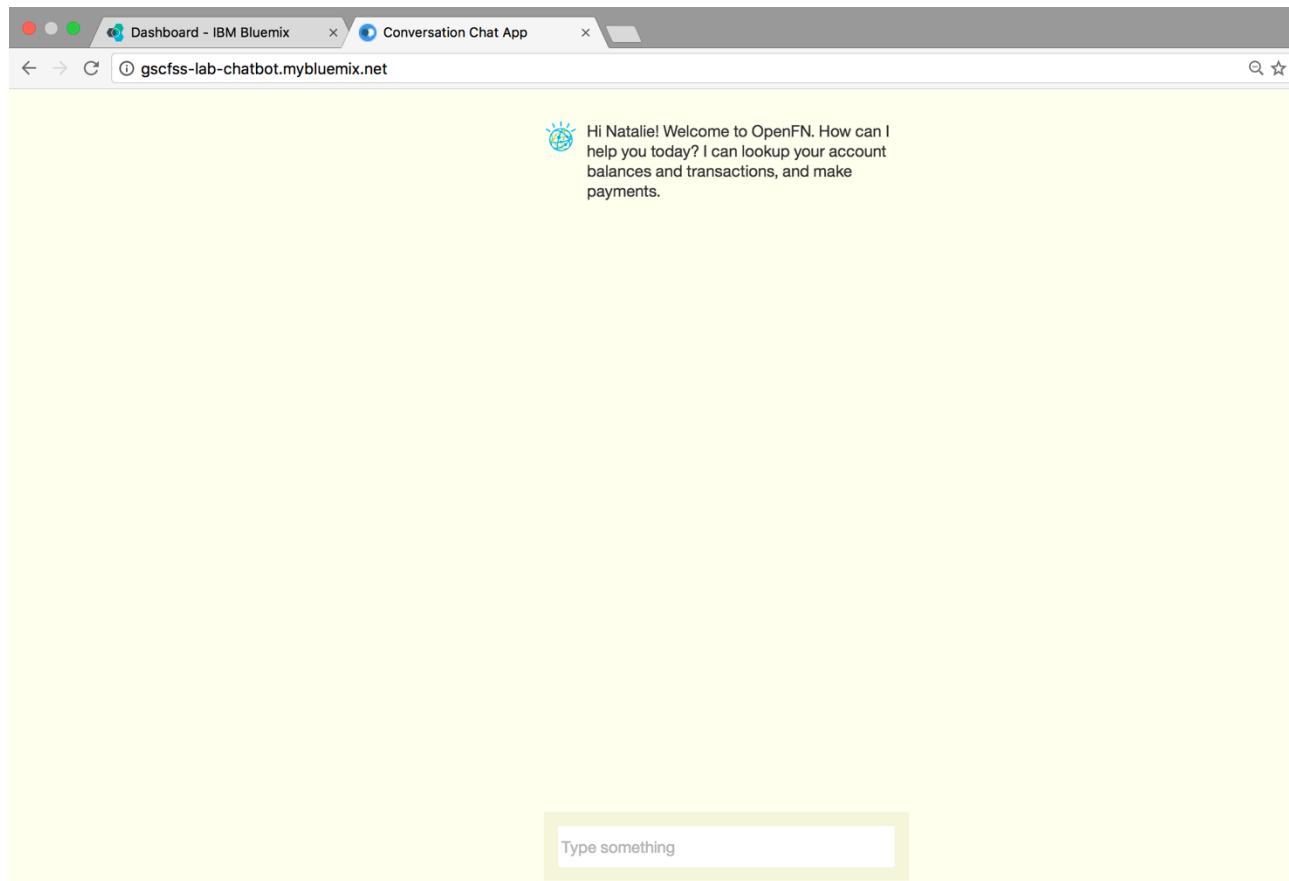
Note – If the Routes link cannot be clicked after the application restarts, then try refreshing the page in your browser and then click on the Routes link.



This screenshot is similar to the previous one but shows the 'Routes' link in the top right corner of the main panel highlighted with a red circle. The rest of the interface, including the sidebar, service cards, and activity log, appears identical to the first screenshot.

2016 Banking and Financial Markets Industry Workshop

- u) A welcome message should be displayed on the chat application.



Part B: Understand Watson Conversation and add new bot features

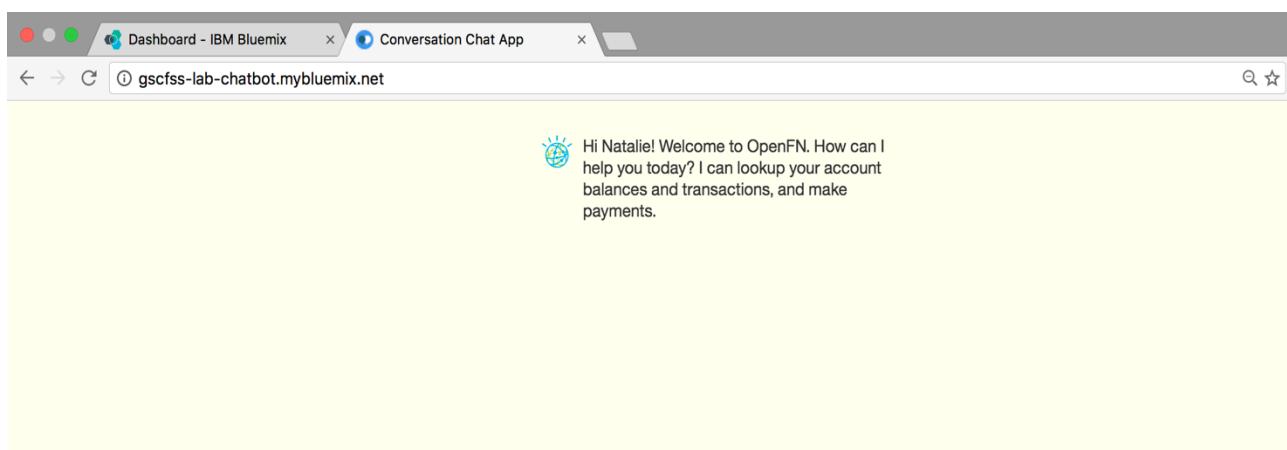
Overview: Watson Conversation provides an integrated environment to develop and train chat bot services that understand and respond to natural language conversations. This part of the lab explains intents, entities and dialog flow components. It also explains how dialog flows utilize intents, entities and context variables to communicate with external systems or services. The goals of this part of the lab are as follows:

- *Interact with the chat application.*
- *Add intents and entities.*
- *Understand dialog flows, nodes and context variables.*

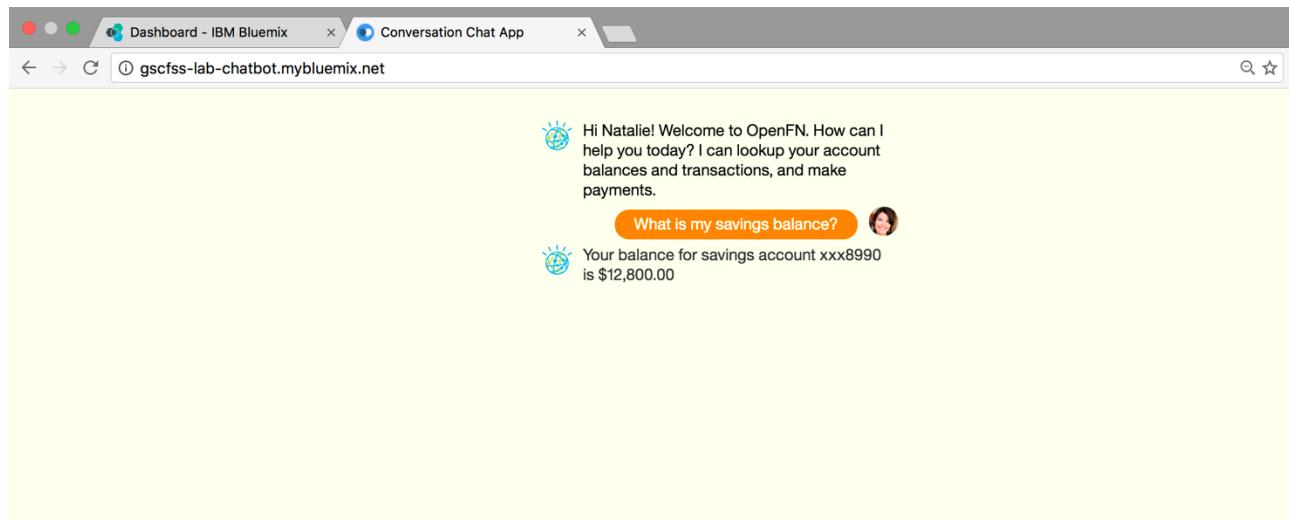
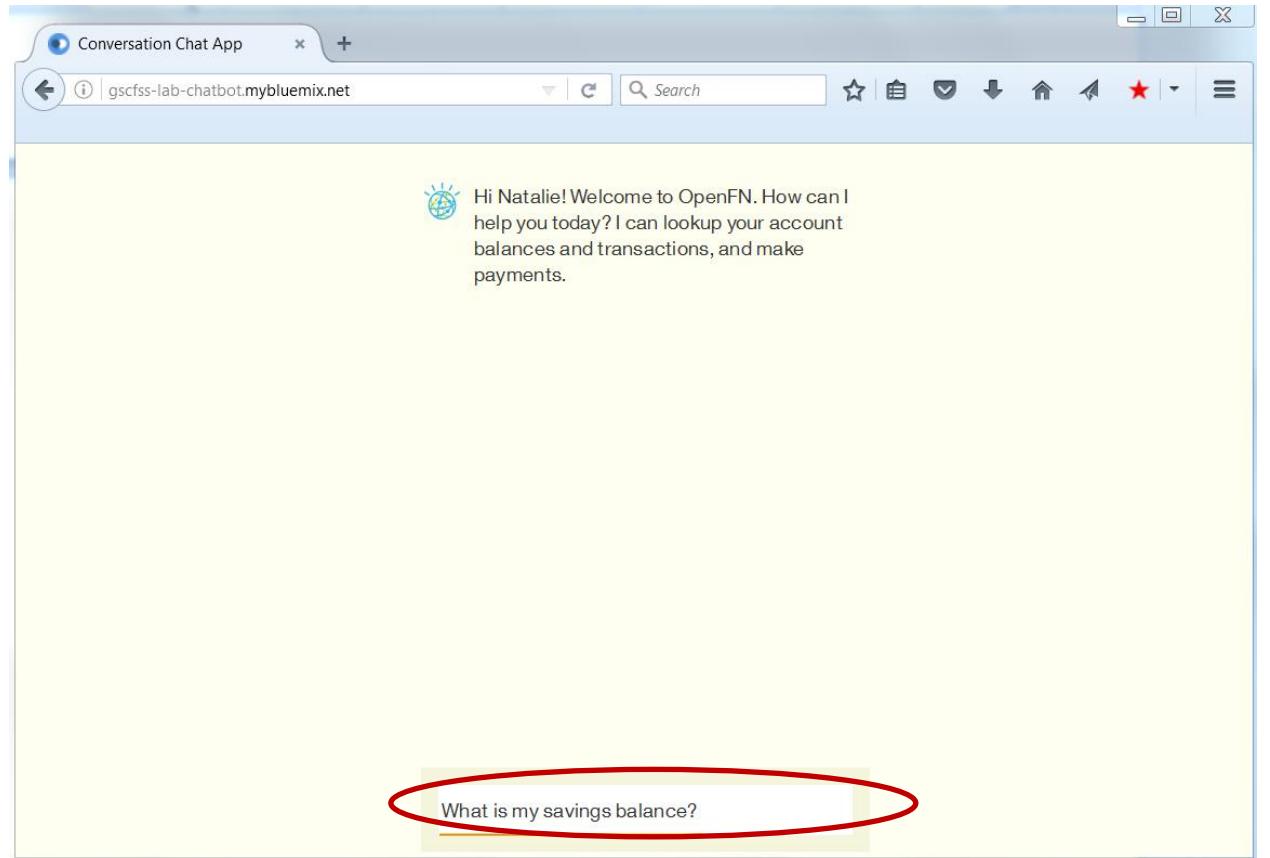
Estimated Duration: 30 minutes

1. *Interact with the chat application.*

- a) In the previous section, you have launched your chat application by clicking on the **Routes** link in the Bluemix Chat Application's Overview page and received a welcome message from Watson Conversation. The welcome message is customized for a specific banking customer – *Natalie Smith*. The customer's profile information is provided to Watson Conversation service from the NodeJS based Cognitive Orchestrator service (shown in the architecture diagram in Part A).



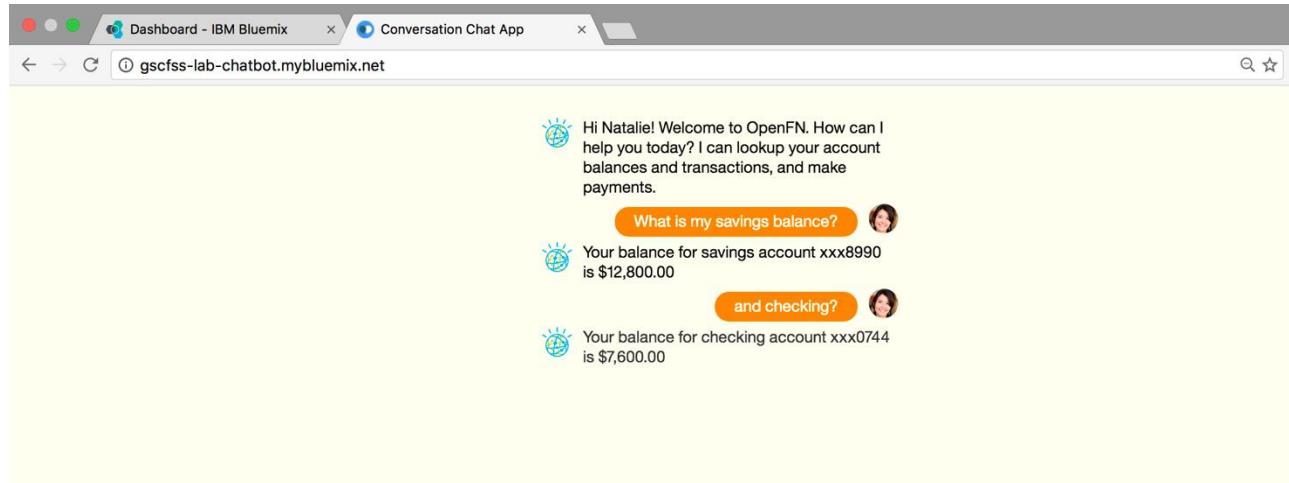
- b) Type in a question as **What is my savings balance?** and press Enter.



The response **Your balance for savings account xxx8990 is \$12,800.00** is returned from Watson Conversation. The Cognitive Orchestrator (NodeJS service) provides Watson Conversation the required data about Natalie's savings account information from the Banking Services (simulated).

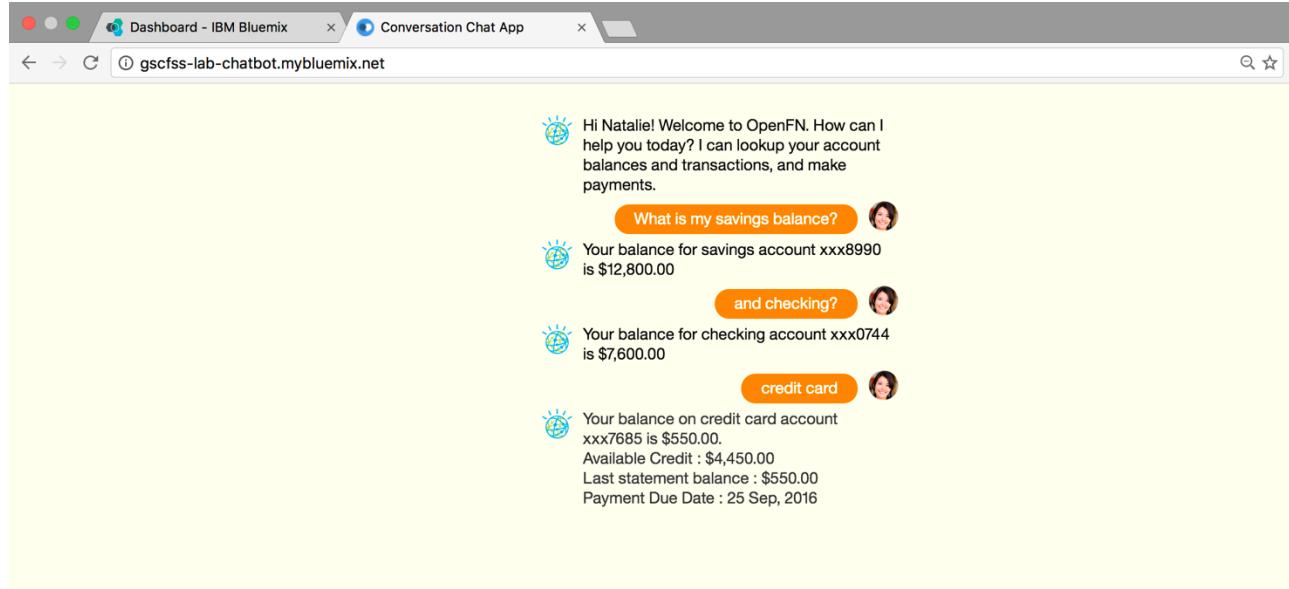
2016 Banking and Financial Markets Industry Workshop

- c) Type in the next input as ***and checking?*** and press Enter.



Notice that the input text did not specify that the customer was looking for an account balance. However, the response of ***Your balance for checking account xxx0744 is \$7,600.00*** shows that Watson Conversation retained the user's intention from the previous interaction (***What is my savings balance?***) and provides a response for the checking account balance in the same context.

- d) Next, enter the input as ***credit card*** and press Enter.



The credit card information is displayed with a set of additional fields such as last statement balance, payment due date, etc.

2016 Banking and Financial Markets Industry Workshop

- e) Type the input as ***make a payment*** and press Enter.

The screenshot shows a web browser window titled "Conversation Chat App" with the URL "gscfss-lab-chatbot.mybluemix.net". The chatbot, represented by a blue icon with a brain, starts by welcoming the user and asking how it can help. It then lists the user's account balances: savings account xxx8990 (\$12,800.00), checking account xxx0744 (\$7,600.00), and a credit card account xxx7685 (\$550.00). After displaying the credit card balance, the chatbot presents a "make a payment" button. The user inputs "make a payment" and the chatbot asks if they want to make a payment of \$550.00 towards the credit card account.

Hi Natalie! Welcome to OpenFN. How can I help you today? I can lookup your account balances and transactions, and make payments.

What is my savings balance?

Your balance for savings account xxx8990 is \$12,800.00

and checking?

Your balance for checking account xxx0744 is \$7,600.00

credit card

Your balance on credit card account xxx7685 is \$550.00.
Available Credit : \$4,450.00
Last statement balance : \$550.00
Payment Due Date : 25 Sep, 2016

make a payment

Alright, should I make a payment of statement balance \$550.00 towards your credit card account xxx7685?

You changed the user's intention from viewing account balances to making payments. Watson Conversation recognized that a credit card account was previously displayed and prompted the user to complete a payment on the credit card account.

- f) Type the input as ***yes*** or ***no*** and press Enter to complete or cancel the payment submission.

The screenshot continues the conversation from the previous step. The user has input "make a payment" and the chatbot has asked if they want to make a payment of \$550.00 towards the credit card account. The user now inputs "yes" to confirm the payment. The chatbot responds with a confirmation message: "Okay, your payment has been submitted."

Hi Natalie! Welcome to OpenFN. How can I help you today? I can lookup your account balances and transactions, and make payments.

What is my savings balance?

Your balance for savings account xxx8990 is \$12,800.00

and checking?

Your balance for checking account xxx0744 is \$7,600.00

credit card

Your balance on credit card account xxx7685 is \$550.00.
Available Credit : \$4,450.00
Last statement balance : \$550.00
Payment Due Date : 25 Sep, 2016

make a payment

Alright, should I make a payment of statement balance \$550.00 towards your credit card account xxx7685?

yes

Okay, your payment has been submitted.

2. Add intents and entities.

- Return to your Bluemix Chat application's dashboard overview and open the Watson Conversation Service. Click on the **Launch tool** button to view your workspace.

The screenshot shows the IBM Bluemix Dashboard with the URL <https://console.ng.bluemix.net/direct=classic#/resources/appGuid=690da4c1-c8d6-4e6c-bcbe-bd8f9dceb498&orgGuid=1e0f9150-6781-4d71-b>. The main content area is titled "Conversation" and describes adding a natural language interface to an application. A "Launch tool" button is highlighted with a red oval. The left sidebar shows the "GSC-FSS-Lab-Chatbot" project with "Conversation" selected under SERVICES.

- Click on the tile of your workspace that was previously imported.

The screenshot shows the "Watson Conversation" workspace list. A workspace titled "OpenFN-Banking-Chatbot-Lab" is highlighted with a red oval. The "Instances" dropdown at the top right is set to "GSC-FSS-Lab-Conversation".

2016 Banking and Financial Markets Industry Workshop

You will see three tabs in the navigation bar – Intents, Entities and Dialog. These tabs will be discussed in subsequent steps.

The screenshot shows a web-based interface for managing a chatbot. At the top, there's a dark header bar with the title "OpenFN-Banking-Chatbot-Lab". Below the header, a navigation bar has three tabs: "Intents" (which is highlighted with a red oval), "Entities", and "Dialog". Under the "Intents" tab, there's a list of intents. The first intent is "#account_balance" with a count of 6, followed by "account information". The second intent is "#payment" with a count of 5, followed by "bill payment". The third intent is "#representative" with a count of 5, followed by "can i speak to someone". There are buttons for "Create" and "Upload" at the top left, and a "Sort by: Newest" dropdown at the top right.

- c) Click on the **Intents** tab. This shows the intents that are recognized by this chat bot. Note that an Intent is referenced with the # symbol preceding its name. Click on the intents **#account_balance** and **#payment** to view the statements that help Watson Conversation understand the user's intention.

This screenshot shows the configuration for the "#account_balance" intent. At the top, it says "#account_balance" and has a button to "Add a new user example...". Below that is a list of user example statements, each with a checkbox to the left:

- account information
- balance in account
- details of account
- list my accounts
- show all accounts
- show me my account
- what is my account balance
- what's the balance on
- what's the current balance on my accounts

This screenshot shows the configuration for the "#payment" intent. At the top, it says "#payment" and has a button to "Add a new user example...". Below that is a list of user example statements, each with a checkbox to the left:

- bill payment
- do i have payments due
- make a payment
- make a payment on this
- pay my bills
- pay my monthly bill
- pay the bill on this
- pay the last statement balance on

An Intent defines a goal that the user is trying to achieve by interacting with a bot (view account balance or make payments). Intents are recognized by Watson Conversation based on user example statements used to train the service. While defining the statements for an intent, it is advisable to provide several variations of statements that cover common utterances so that Watson Conversation is able to determine the user's intention with better accuracy. Notice that the intent's user example statements do not specify the account type that the user is trying to view or the account that the user wants to make a payment on. Those details are provided by entities, which will be discussed later.

2016 Banking and Financial Markets Industry Workshop

For example, the statement **account information** is expected to handle utterances such as **savings account information**, **checking account information** and so on.

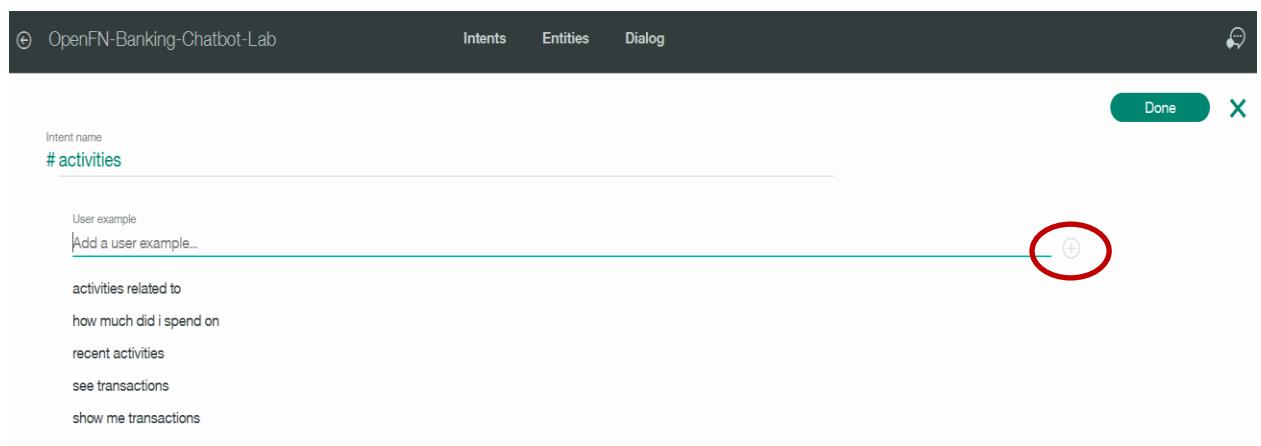
- d) Click on **Create**. You will be adding a new intent that recognizes user statements for viewing account transactions/activities.

Copy and Paste the *Intent name* and the *User example* statements shown below to define the new intent.

Intent name: **activities**

User example: **activities related to**
how much did i spend on
recent activities
see transactions
show me transactions

Hint: You can press the Enter key or click the (+) icon highlighted below after each user example statement that is entered in the textbox - *Add a user example....*

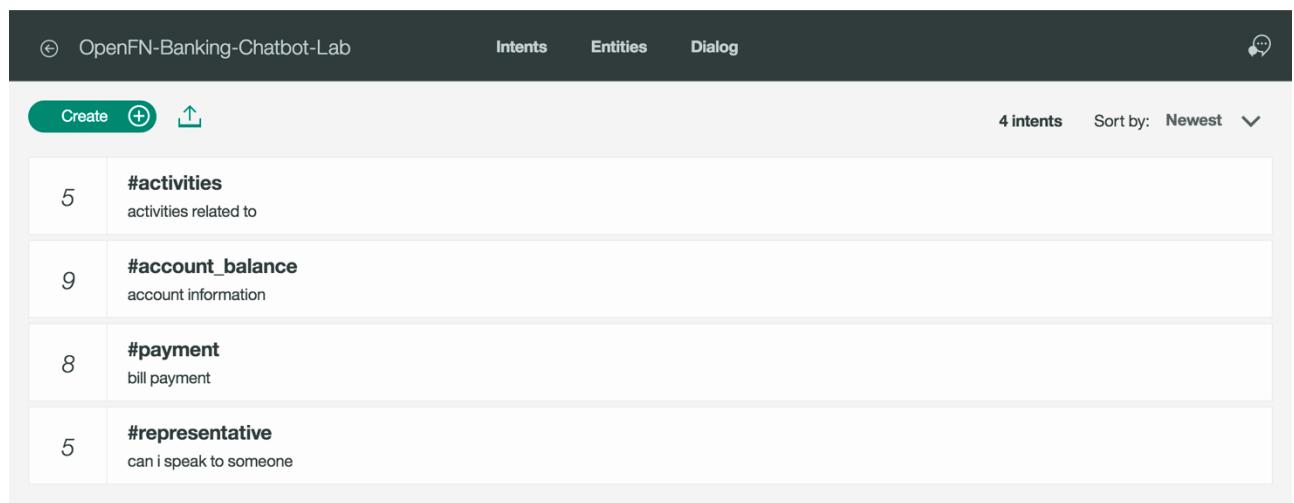


Intent name
#activities

User example
Add a user example...

activities related to
how much did i spend on
recent activities
see transactions
show me transactions

- e) Click **Done** to save the **#activities** Intent. Watson Conversation automatically trains the service when changes are made to the workspace.



ID	Intent Name	Description
5	#activities	activities related to
9	#account_balance	account information
8	#payment	bill payment
5	#representative	can i speak to someone

- f) Click on the **Entities** tab. An Entity is referenced with the @ symbol preceding its name.

The screenshot shows the 'Entities' tab selected in the top navigation bar. Below the tabs, there is a 'Create' button and a list of entities. The first entity listed is '@account_type' with values 'checking, savings, credit card'. The second entity listed is '@acknowledgement' with values 'no, yes'. A red circle highlights the 'Entities' tab.

Entity is the part of the user's input that provides specifics regarding the user's intent. For example, if the user's question is '**What is my savings balance?**', an entity would be used to identify the account type as **savings**.

- g) Click on **@account_type** entity to view its details. As the name implies, this entity is used to identify the type of an account. The values **checking**, **savings** and **credit card** list the types of accounts handled by this bot. Synonyms are alternate words that can be used to refer to each of the entity's values.

The screenshot shows the details for the '@account_type' entity. It includes a 'Create' button, a list of values ('checking', 'credit card', 'savings'), and a 'Add synonyms...' button. Below the values, it shows '(0 Synonyms)' for 'checking', '(2 Synonyms)' for 'credit card', and '(1 Synonym)' for 'savings'.

- h) Click **Create** to add a new entity. You will be defining a new entity that identifies the spending category (dining, travel, gas, grocery) of banking activities/transactions that the user wants to view.

Enter the name of the entity as **spending_category**

The screenshot shows a form for creating a new entity named '@spending_category'. It has fields for 'Value' (with placeholder 'Add a value, for example, Cat') and 'Synonyms' (with placeholder 'Add synonyms...').

2016 Banking and Financial Markets Industry Workshop

- i) Enter the values and synonyms for the entity as shown in the table below. Click the (+) icon to add a new **Value** (highlighted in the image).

Value Hint: Click the (+) icon to enter a new value	Synonyms Hint: Press the Tab key to add the next synonym.
dining	dinner eating out lunch meals
fuel	gas
grocery	market
travel	airline flight hotel rail rental tourism train

Entity
@spending_category

Done X

Value
dining

Synonyms
dinner X meals X Add synonyms... 

- g) Click **Done** to save the **@spending_category** entity.

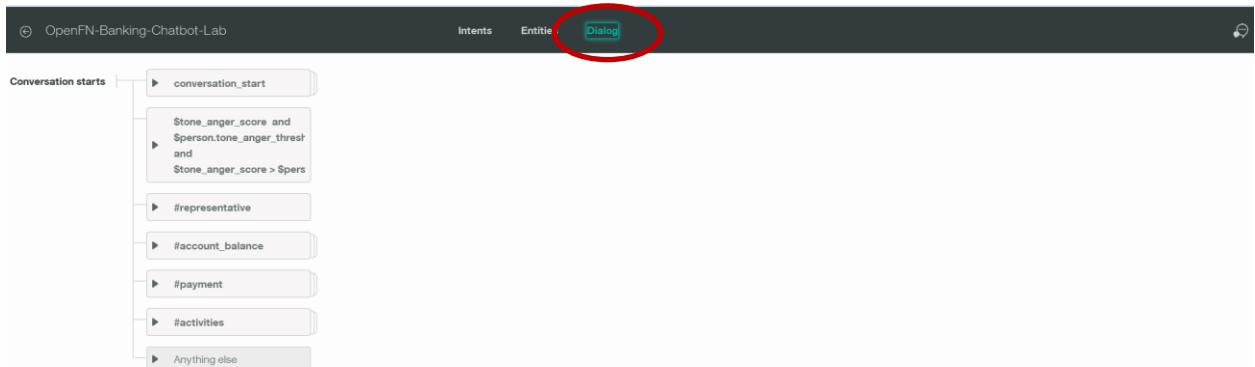
@spending_category 

+ Add a new value

Value	Synonyms	Count
dining	dinner eating out lunch meals	(4 Synonyms)
fuel	gas	(1 Synonym)
grocery	market	(1 Synonym)
travel	airline flight hotel rail rental tourism train	(7 Synonyms)

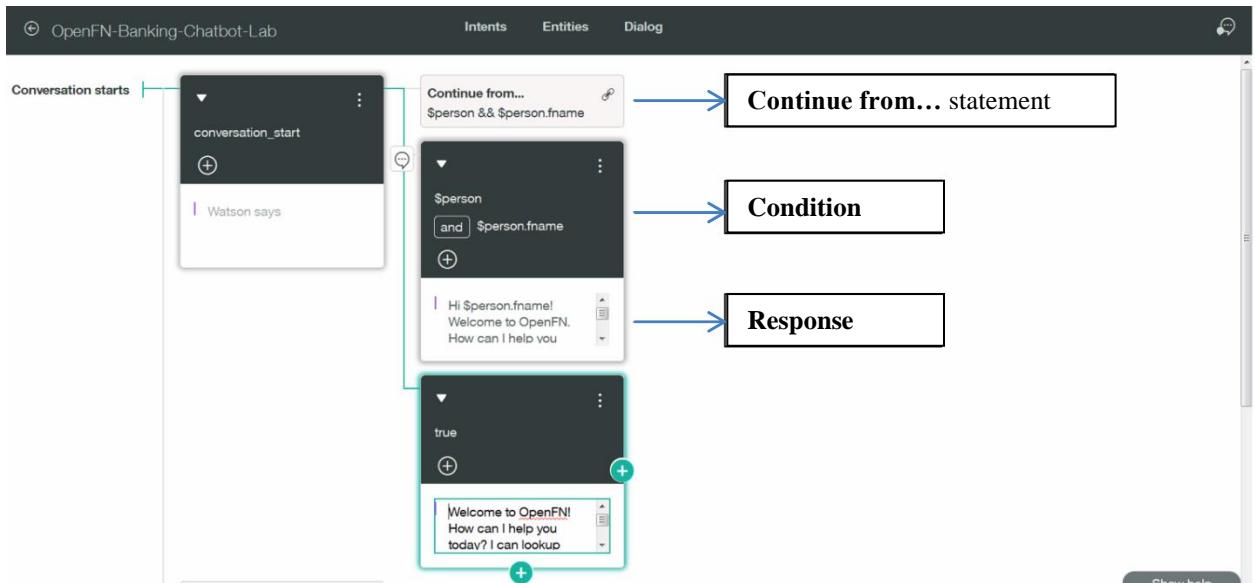
3. Understand dialog flows, nodes and context variables.

- Click on the **Dialog** tab in the navigation bar



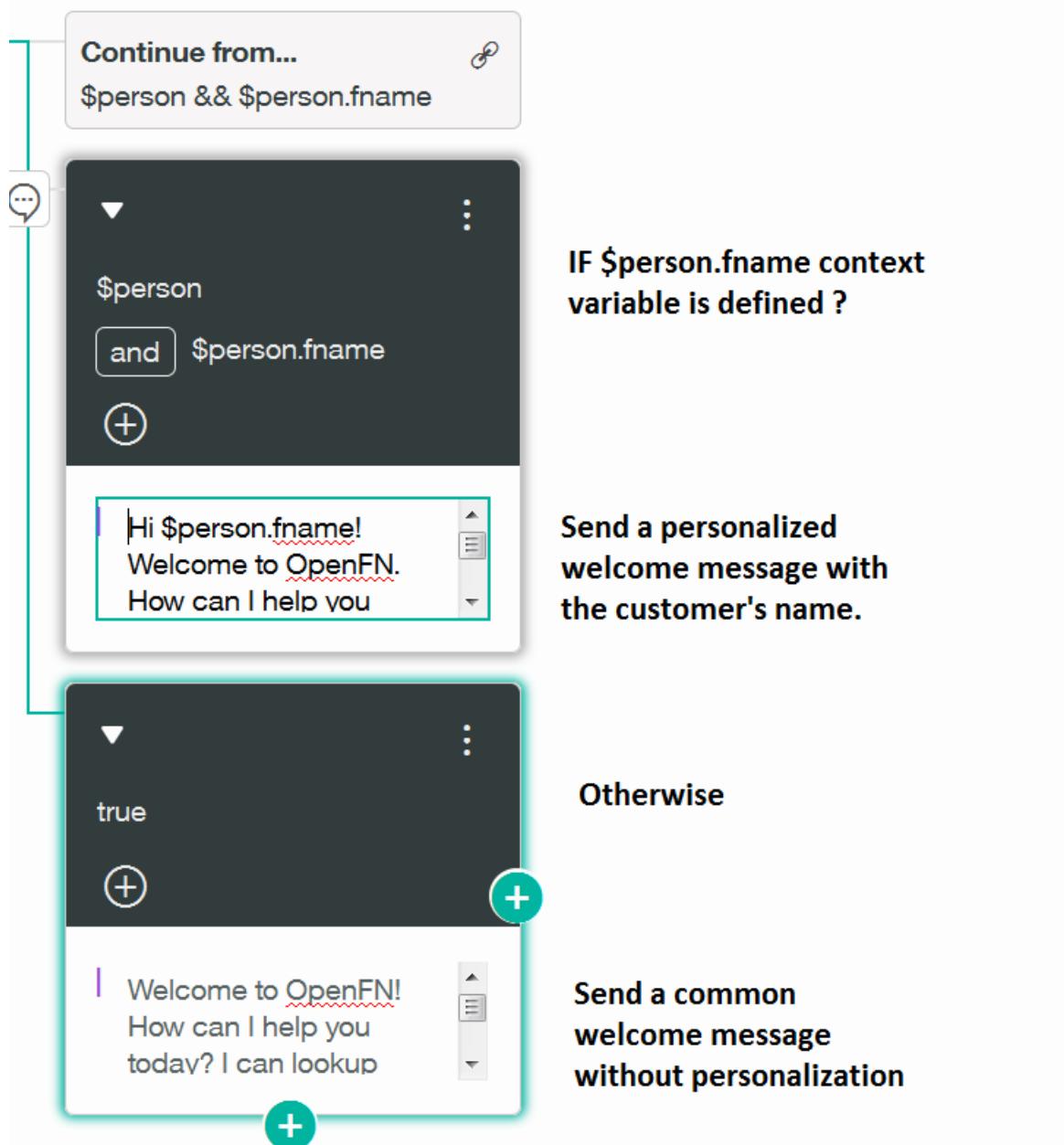
A Dialog defines the flow of the conversation and uses intents and entities to branch into a specific conversation flow and provide responses to the user's statements. Dialog nodes are evaluated from top to bottom (peer nodes) and left to right (child nodes). In this example, **conversation_start** node is evaluated first and **Anything else** node is evaluated last.

- Click on the **conversation_start** node to expand it. Additionally, click on the nodes attached to **conversation_start** node to expand the child nodes as shown in the image below. The **conversation_start** node is the first node in a dialog and provides a welcome message to the user. A dialog node has two sections – **condition** and **response**. The condition specifies rules that would permit the dialog response to be sent to the end user. The response section defines the text for the output.

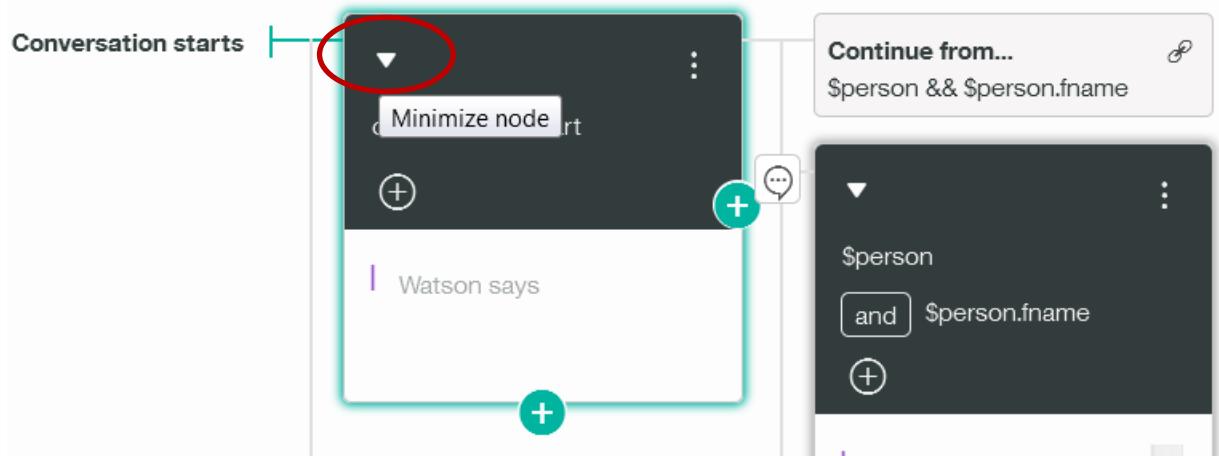


In this example, the **conversation_start** node does not define a response. It is linked to a **Continue from** statement. **Continue from** statements allow a dialog node to check subsequent nodes for a condition before sending a response. In this case, the dialog needs to check for the existence of a person's first name (customer *Natalie Smith*) in order to customize the welcome message. If the person's first name is not provided to the dialog flow, then the last node with a **true** condition is executed and a default welcome message is sent in the output response.

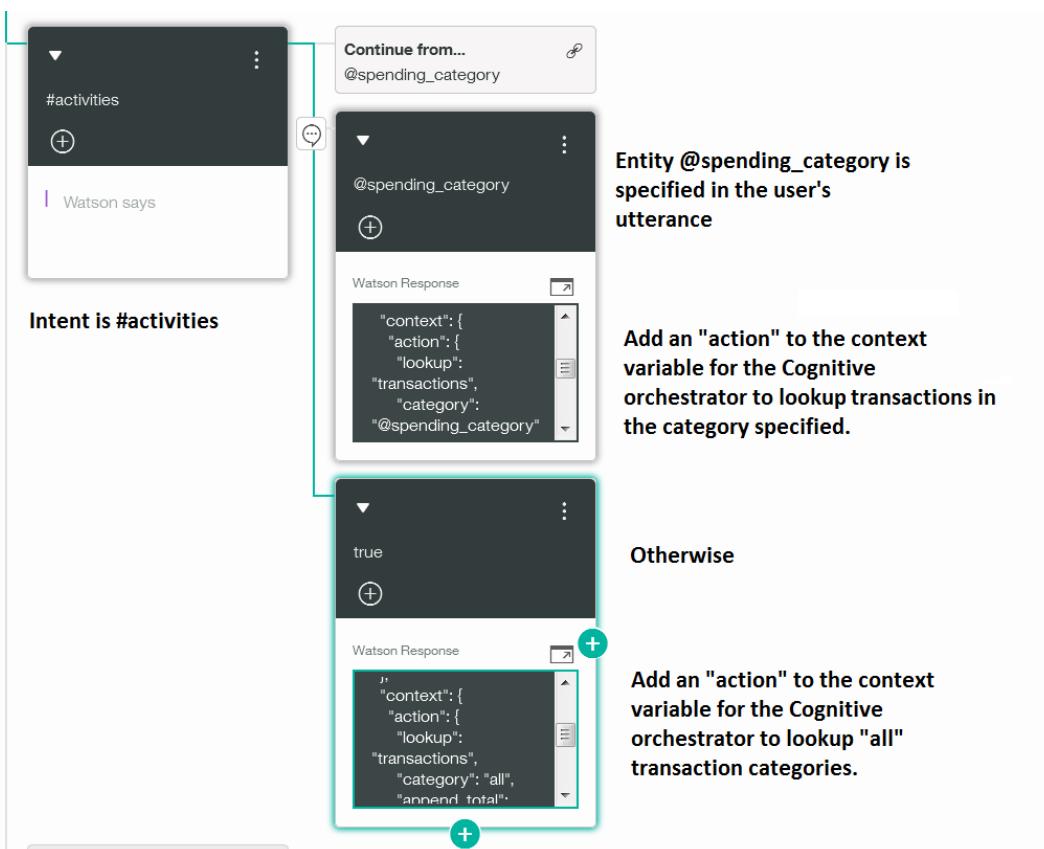
Watson Conversation's Dialog uses context variables identified with the \$ symbol preceding the variable name. Context variables enable Watson Conversation's dialog to communicate with other systems. In the chat application, the Cognitive Orchestrator NodeJS service adds the customer's profile information using a **\$person** context variable. This context variable is being referenced in the dialog node's condition and response.



- c) Minimize the **conversation_start** node by clicking on **Minimize node**.

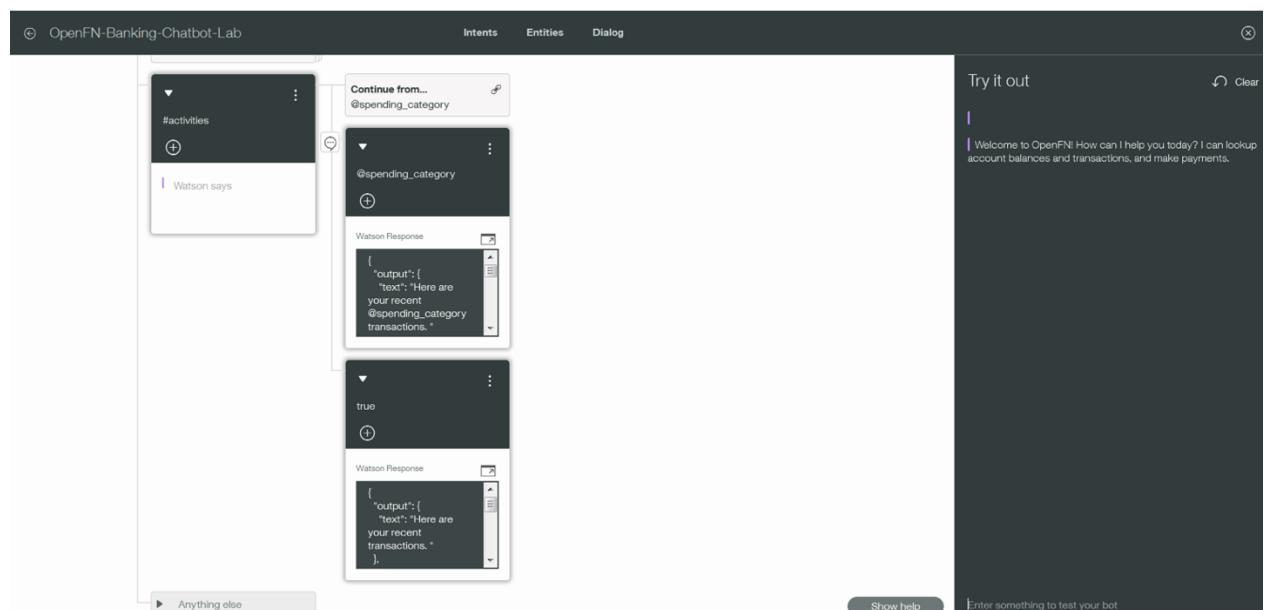
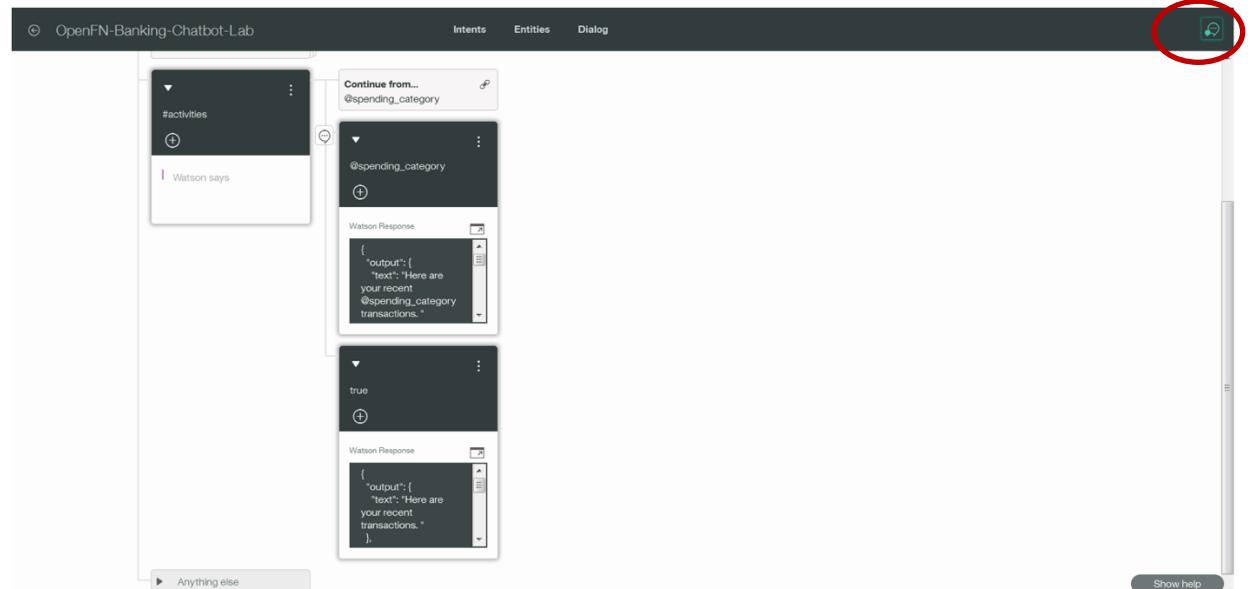


- d) Click on the **#activities** node to expand the dialog flow. Click on the subsequent nodes within **#activities** branch to expand the child nodes as shown below. This branch of the dialog will handle the intent **#activities** and the entity **@spending_category** that you have added in the previous steps. It follows a similar structure of a **Continue from** statement and node conditions, as discussed in the **conversation_start** node. A description of each node's condition and response is provided below for the **#activities** node's branch.

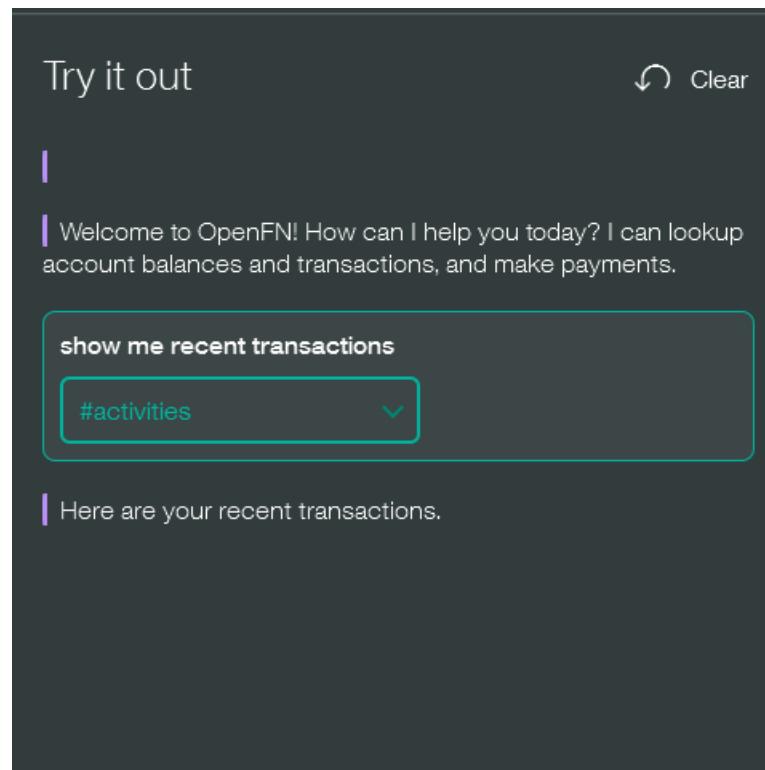
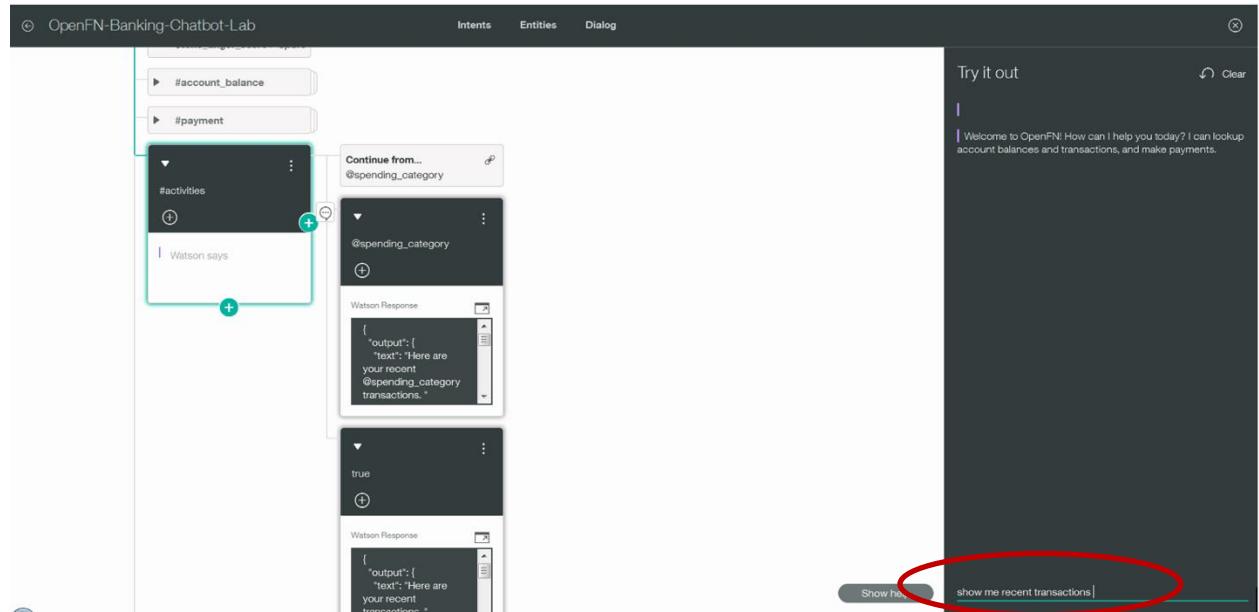


We have defined a custom **action** context variable in the response from Watson Conversation. The **action** context variable informs the Cognitive Orchestrator service to **lookup** user transactions before sending the response to the front end application. The Cognitive Orchestrator service performs additional data lookup functions by calling the Banking Services (simulated) based on this **action** variable and appends the account activities/transactions before returning the response.

- e) Test this dialog flow within Watson Conversation by clicking on the **chat icon** at the top right.

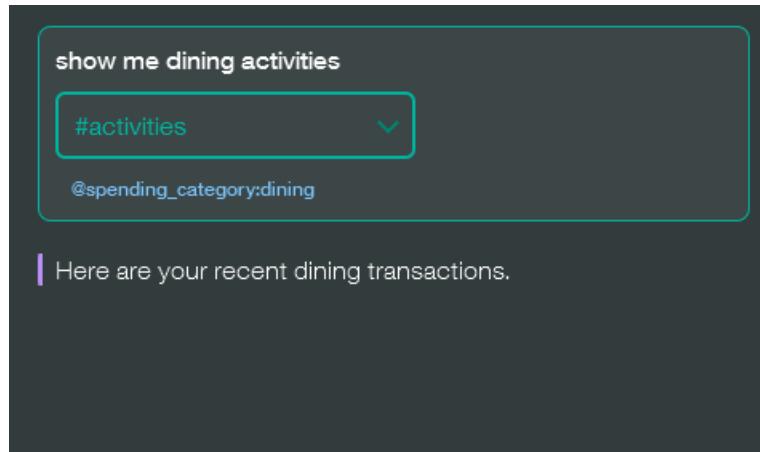


- f) Enter a statement as ***show me recent transactions*** and press Enter to submit the statement.



The response ***Here are your recent transactions*** is returned from the last node (**true** condition) executed in the dialog flow within the **#activities** branch, since you did not specify an entity for **@spending_category** in your question. The testing window also indicates the intent **#activities** that was recognized in the input statement.

- g) Next, type in a statement as ***show me dining activities*** and press Enter to submit the statement.



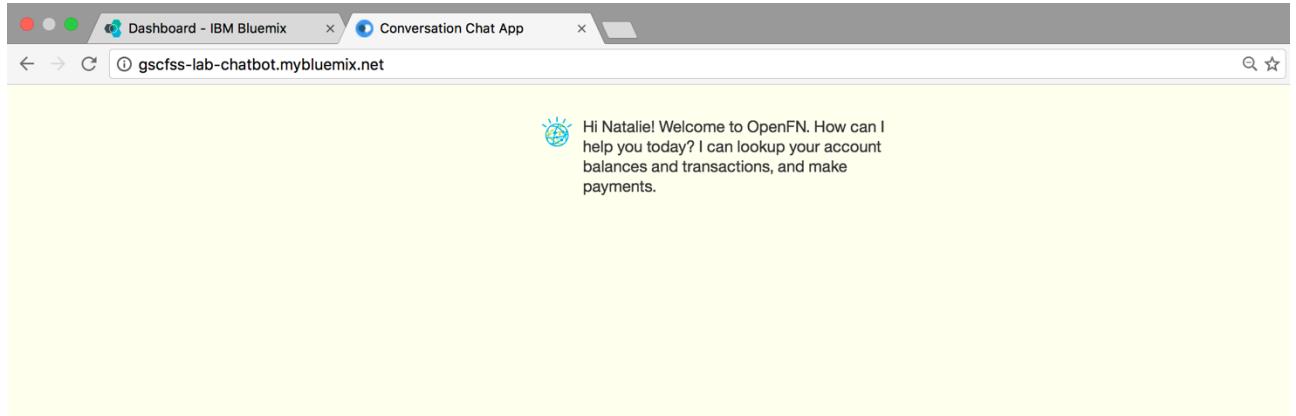
The response ***Here are your recent dining transactions*** is the first node (**@spending_category** condition) executed under the **Continue from** statement. Since the statement specified an intent as **#activities** and an entity as **@spending_category = dining**, Watson Conversation executed the node that met those conditions in the dialog flow. The testing window displays the intent as **#activities** and entity **@spending_category** as **dining** that were recognized in the user's input statement.

We do not see the list of transactions/activities yet, because they will be populated by the Cognitive Orchestrator NodeJS service (**action** context variable **lookup** functions). The NodeJS service will look up the account transactions for all categories, or for a specific category that Watson Conversation provides in the context variable while returning the output response.

- h) Now that the intent, entity and dialog flow for activities have been defined, we can verify the changes in the chat application. Click on the chat application's **Routes** link (found on your Bluemix chat application's overview page) and refresh the chat page to start a new conversation.

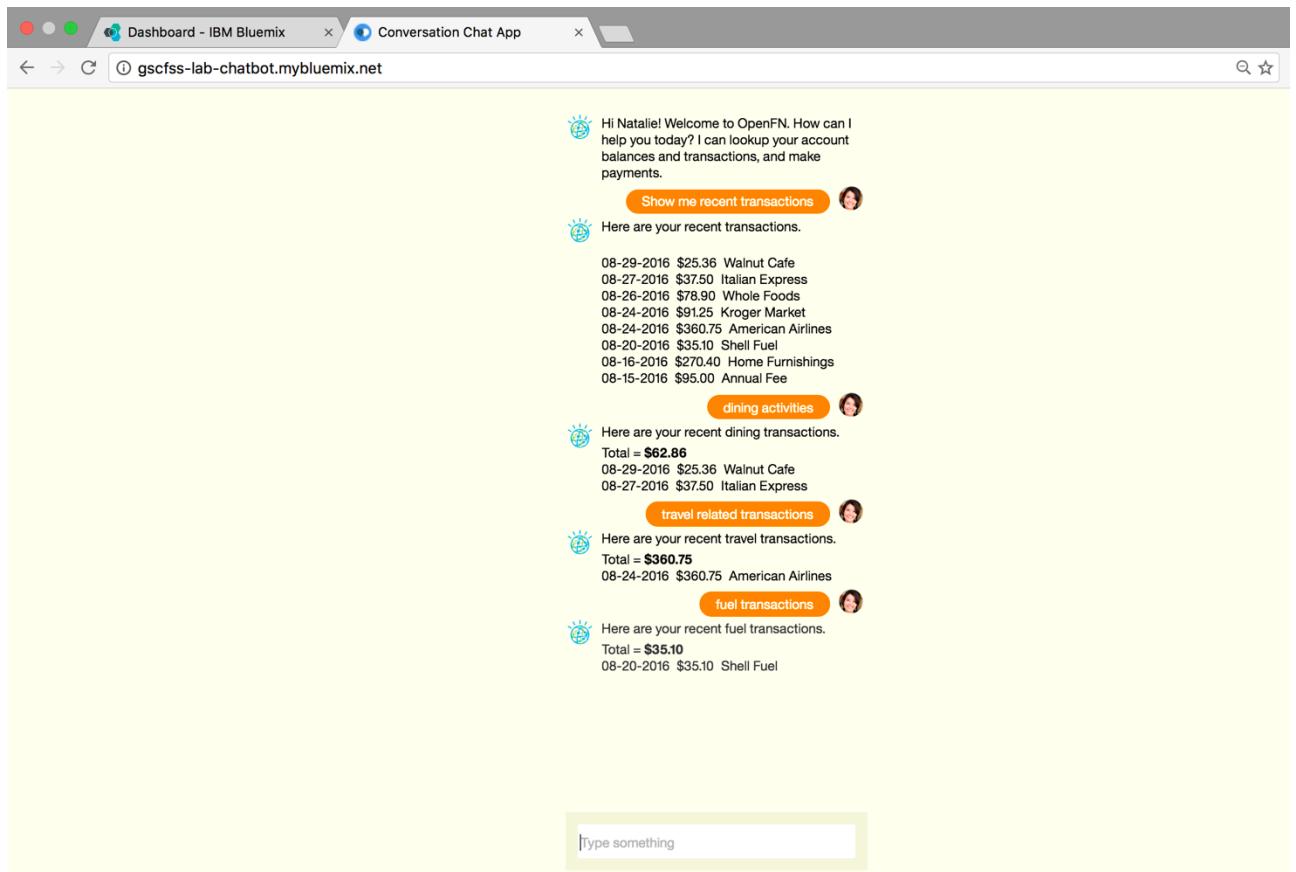
The screenshot shows the IBM Bluemix dashboard. On the left, there is a sidebar with links like "Back to Dashboard...", "Overview", "SDK for Node.js™", "Files", "Logs", "Environment Variables", "Start Coding", and "SERVICES". The main area shows the "GSC-FSS-Lab-Chatbot" application. It has a red circle highlighting the "Routes" link, which is located above the instance configuration. The instance configuration shows 1 instance, 256 MB memory quota, and 1.750 GB available memory. To the right, there is an "APP HEALTH" section showing "Your app is running." and a "RESTART" button, and an "ACTIVITY LOG" section showing a log entry for "gsclab1234@gmail.com" starting the "chatbot-gsclab1234-1231 app" at 12:32 PM on 8/19/16.

2016 Banking and Financial Markets Industry Workshop



- i) Enter (or Copy and Paste) the statements below into the chat application to see the transactions for different categories of spending.

Show me recent transactions
dining activities
travel related transactions
fuel transactions



Part C: Use Watson Tone Analyzer to detect emotions in conversations

Overview: This part of the lab will introduce the Watson Tone Analyzer service's emotion detection capabilities and describe techniques used to drive the dialog in Watson Conversation based on the emotions detected by Tone Analyzer. The goals of this section of the lab are as follows:

- Overview of the Tone Analyzer service for emotion detection
- View how the service is invoked in the chat application
- Understand how tone detection is used to alter dialog flows

Estimated Duration: 10 minutes

1. Overview of the Tone Analyzer service for emotion detection

The Watson Tone Analyzer Service uses linguistic analysis to detect three types of tones from written text: emotions, social tendencies, and writing style. Emotions identified include things like anger, fear, joy, sadness, and disgust. Identified social tendencies include things from the Big Five personality traits used by some psychologists. These include openness, conscientiousness, extraversion, agreeableness, and emotional range. Identified writing styles include confident, analytical, and tentative.

The chat application uses Tone Analyzer to detect anger among the emotion tone results. Emotional tone is inferred from different types of emotions and feelings that people express in their language. For each of these emotions, the service outputs a score that lies between 0 to 1 that indicates the probability that the emotion came across in the text.

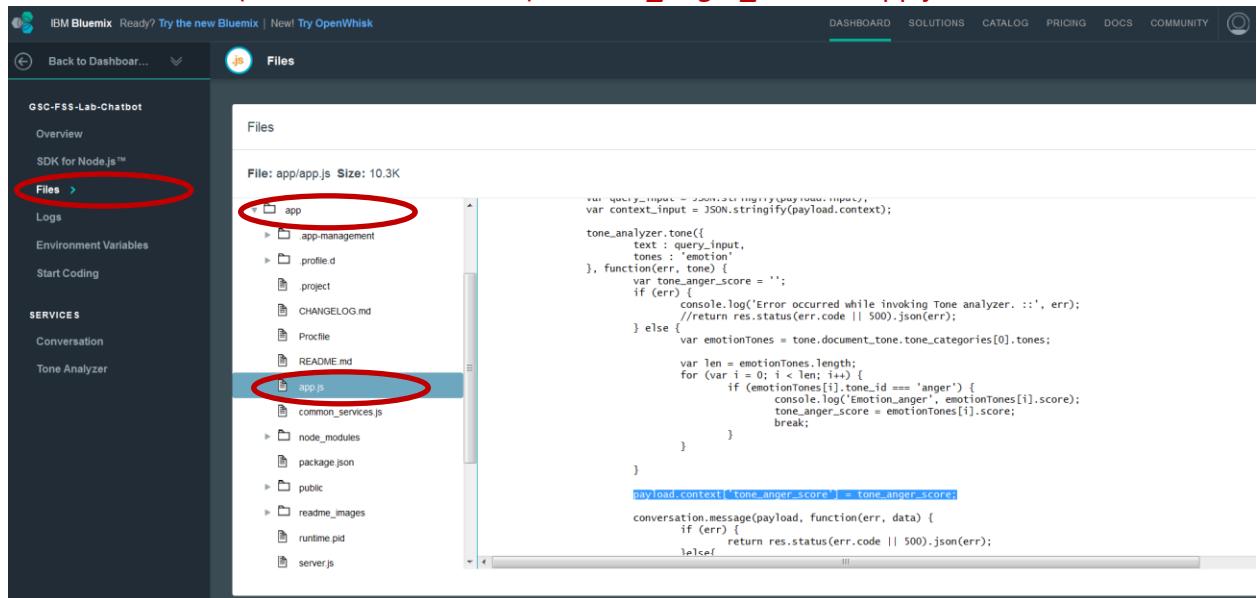
Emotion	Description	Low value	High Value
joy	Joy or happiness has shades of enjoyment, satisfaction and pleasure. There is a sense of well-being, inner peace, love, safety and contentment.	Less than 0.5 - less likely to be perceived as joyful.	More than 0.75 - Highly likely to be perceived as joyful.
fear	A response to impending danger. It is a survival mechanism that is a reaction to some negative stimulus. It may be a mild caution or an extreme phobia.	Less than 0.5 - less likely to be perceived as scared.	More than 0.75 - Highly likely to be perceived as scared.
sadness	Indicates a feeling of loss and disadvantage. When a person can be observed to be quiet, less energetic and withdrawn, it may be inferred that sadness exists.	Less than 0.5 - less likely to be perceived as sad.	More than 0.75 - Highly likely to be perceived as sad.
disgust	An emotional response of revulsion to something considered offensive or unpleasant. It is a sensation that refers to something revolting.	Less than 0.5 - less likely to be perceived as disgusted.	More than 0.75 - Highly likely to be perceived as disgusted.
anger	Evoked due to injustice, conflict, humiliation, negligence or betrayal. If anger is active, the individual attacks the target, verbally or physically. If anger is passive, the person silently sulks and feels tension and hostility.	Less than 0.5 - less likely to be perceived as angry.	More than 0.75 - Highly likely to be perceived as angry.

2. View how the service is invoked in the chat application

The NodeJS application invokes the Tone Analyzer service for every user input that it receives. The result of the anger score is then passed into Watson Conversation using a context variable. For the purpose of this lab, the customer profile information has an anger threshold provided as 0.49 (standard threshold value). If the anger emotion score results from Tone Analyzer exceed this threshold, then Watson Conversation's Dialog would recognize this increase in the anger score and take appropriate action, such as transfer the user to a live agent for further assistance. The anger threshold can be adjusted to adapt to the way an individual user converses with a chat bot. For example, a user may be accustomed to chatting with a bot with a high anger score on a regular basis. The threshold for such users should be adjusted accordingly by considering previous interactions and outcomes.

- (Optional) Return to your Bluemix dashboard and open the chat application's overview page. Click on Files and then expand the **app** folder. Look for a file named **app.js** and scroll to the code highlighted in the image below.

Hint – Search (Ctrl+F or command+F) for *tone_anger_score* in app.js



The screenshot shows the IBM Bluemix dashboard with the "GSC-FSS-Lab-Chatbot" application selected. On the left, the "Files" section is expanded, showing a tree view of files and folders. The "app" folder is expanded, and within it, the "app.js" file is highlighted with a red oval. The right pane displays the contents of the "app.js" file. A portion of the code is highlighted with a red oval, specifically the line: `payload.context['tone_anger_score'] = tone_anger_score;`. The code block is as follows:

```

tone_analyzer.tone({
  text : query_input,
  tones : 'emotion'
}, function(err, tone) {
  var tone_anger_score = '';
  if (err) {
    console.log('Error occurred while invoking Tone analyzer. :::', err);
    //return res.status(err.code || 500).json(err);
  } else {
    var emotionTones = tone.document_tone.tone_categories[0].tones;

    var len = emotionTones.length;
    for (var i = 0; i < len; i++) {
      if (emotionTones[i].tone_id === 'anger') {
        console.log('Emotion_anger', emotionTones[i].score);
        tone_anger_score = emotionTones[i].score;
        break;
      }
    }
  }
}

payload.context['tone_anger_score'] = tone_anger_score;

conversation.message(payload, function(err, data) {
  if (err) {
    return res.status(err.code || 500).json(err);
  } else {
    res.end();
  }
});
  
```

```

tone_analyzer.tone({
  text : query_input,
  tones : 'emotion'
}, function(err, tone) {
  var tone_anger_score = '';
  if (err) {
    console.log('Error occurred while invoking Tone analyzer. :::', err);
    //return res.status(err.code || 500).json(err);
  } else {
    var emotionTones = tone.document_tone.tone_categories[0].tones;

    var len = emotionTones.length;
    for (var i = 0; i < len; i++) {
      if (emotionTones[i].tone_id === 'anger') {
        console.log('Emotion_anger', emotionTones[i].score);
        tone_anger_score = emotionTones[i].score;
        break;
      }
    }
  }
}

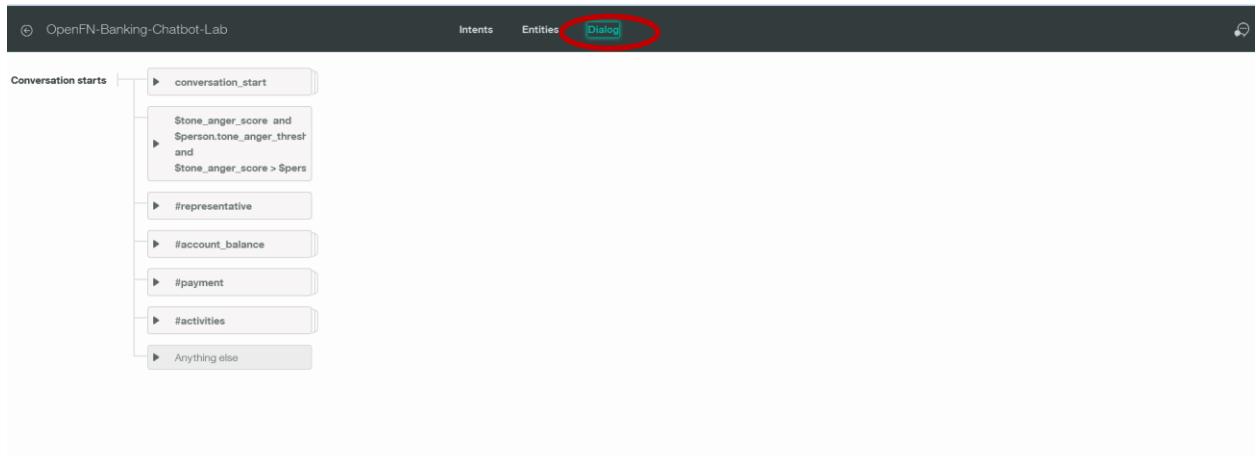
payload.context['tone_anger_score'] = tone_anger_score;

conversation.message(payload, function(err, data) {
  if (err) {
    return res.status(err.code || 500).json(err);
  } else {
    res.end();
  }
});
  
```

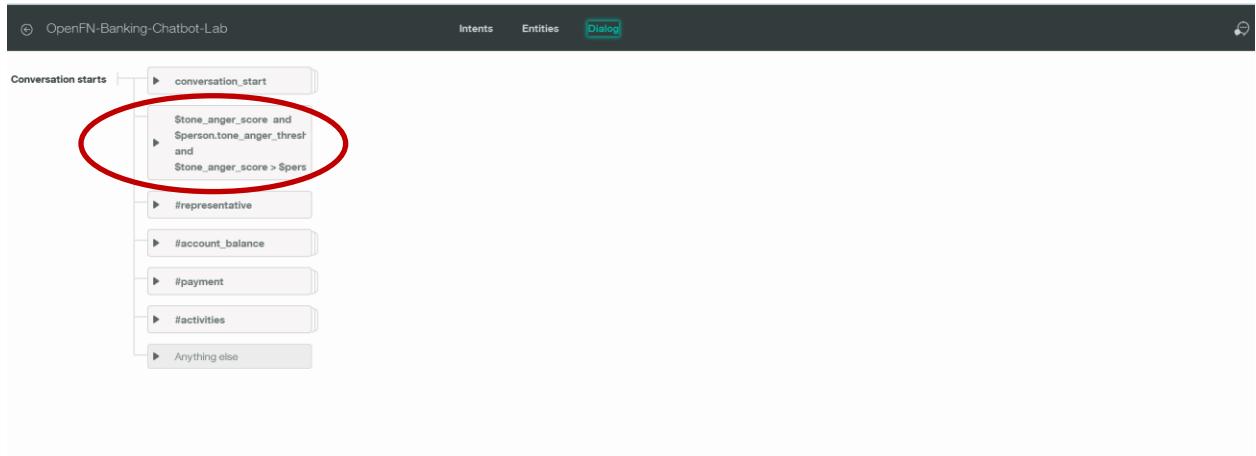
The code in the example above calls Tone Analyzer service for **emotion** detection and extracts the anger sub-tone from the results. It then sets the anger score as a context variable (encircled in red) before invoking Watson Conversation.

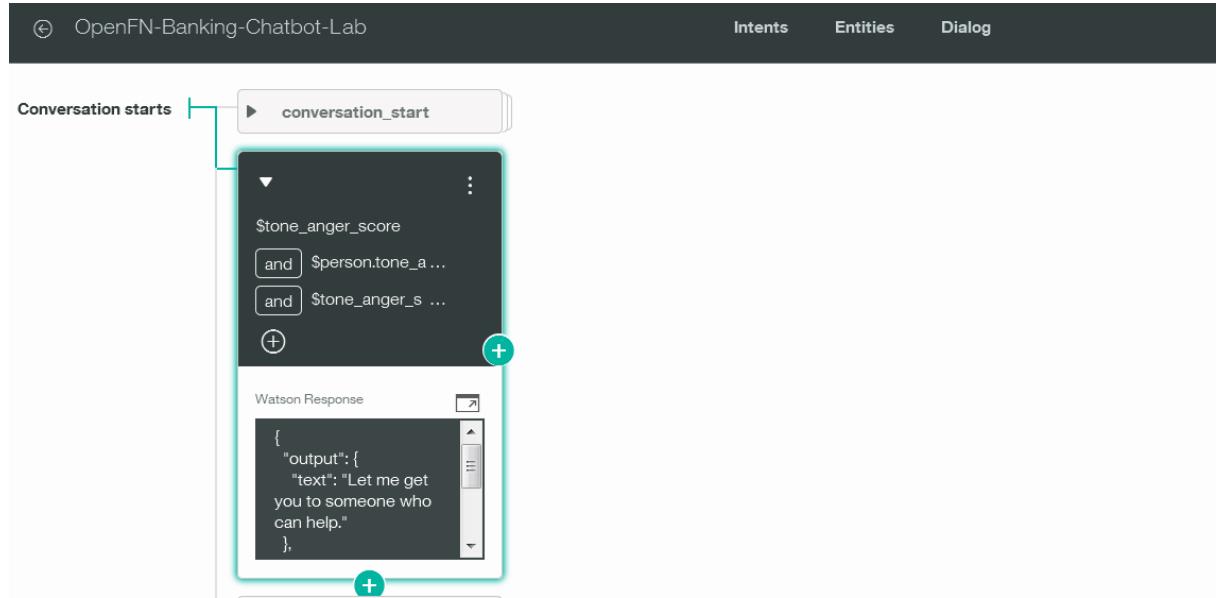
3. Understand how Tone Analyzer results are used to alter dialog flows

- Return to your Watson Conversation workspace and click on the Dialog tab in the navigation bar.



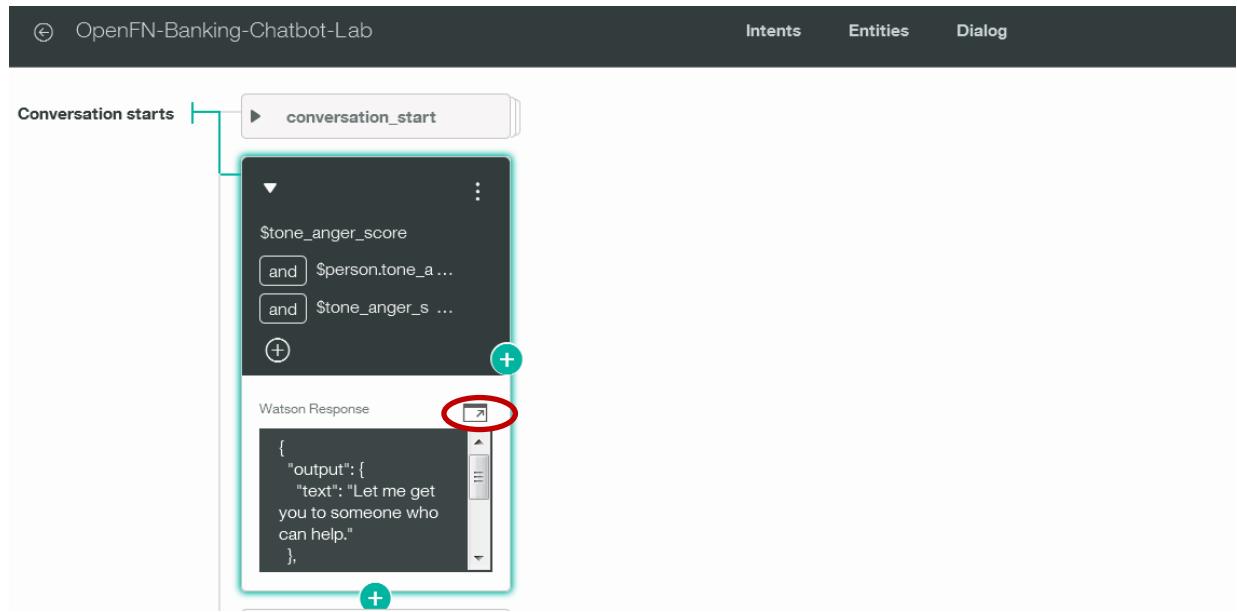
- Expand the node with **\$tone_anger_score** in the condition.





This node looks for the existence of context variables **\$Stone_anger_score** and **\$person.tone_anger_threshold** and compares their values (**\$Stone_anger_score > \$person.tone_anger_threshold**). If the anger tone score exceeds the defined threshold for the user, then the output response is configured to send an **action** to the calling chat application to be re-directed to a live agent.

- c) To see the advanced JSON response view, click on the expand icon highlighted below.



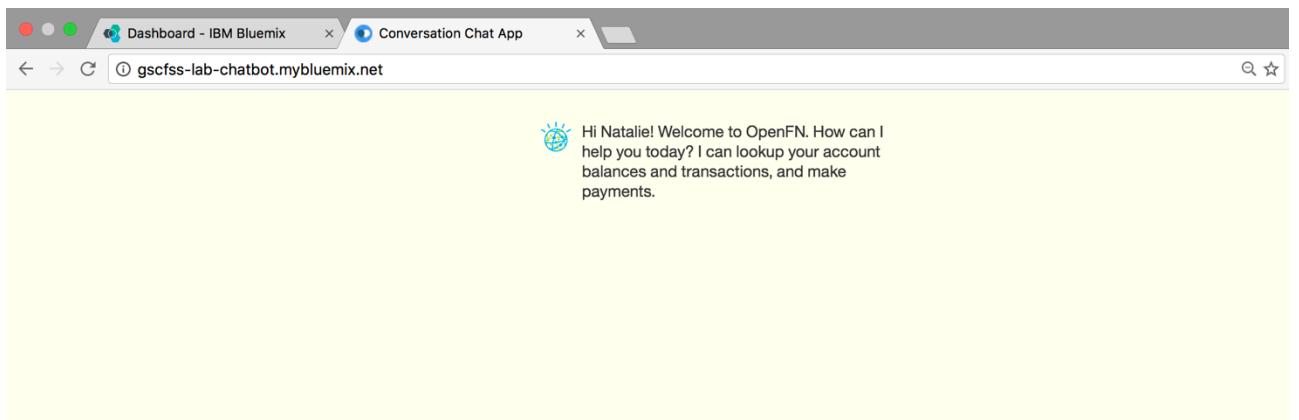
2016 Banking and Financial Markets Industry Workshop

- d) The context variable's action defined here is **directToLiveAgent**, that acts as an indicator to the calling application to transfer the chat to a live agent.

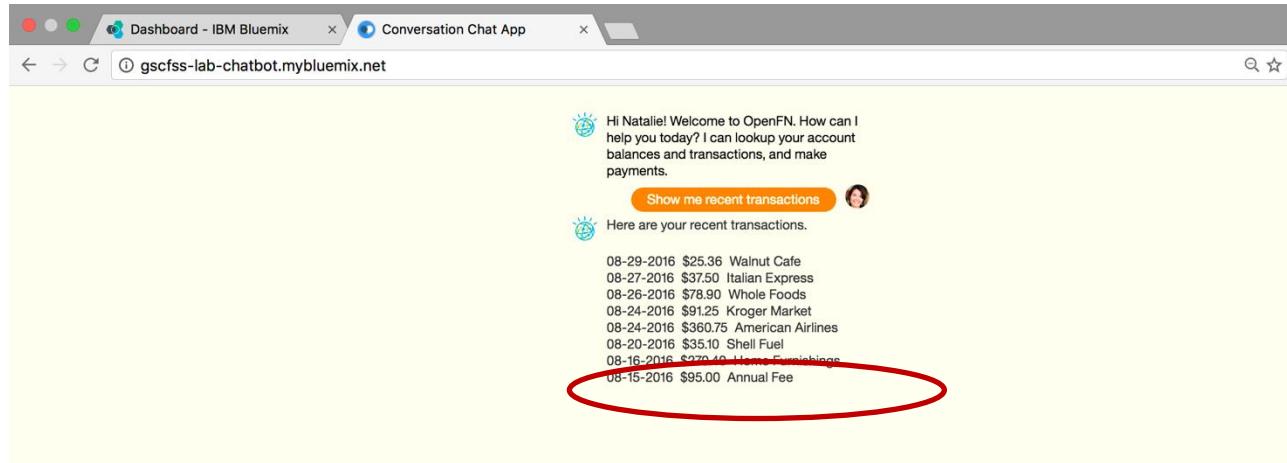
Advanced response

```
{  
  "output": {  
    "text": "Let me get you to someone who can help."  
  },  
  "context": {  
    "action": {  
      "directToLiveAgent": true  
    }  
  }  
}
```

- e) Open the chat application's **Routes** link and refresh the page to get the welcome message.



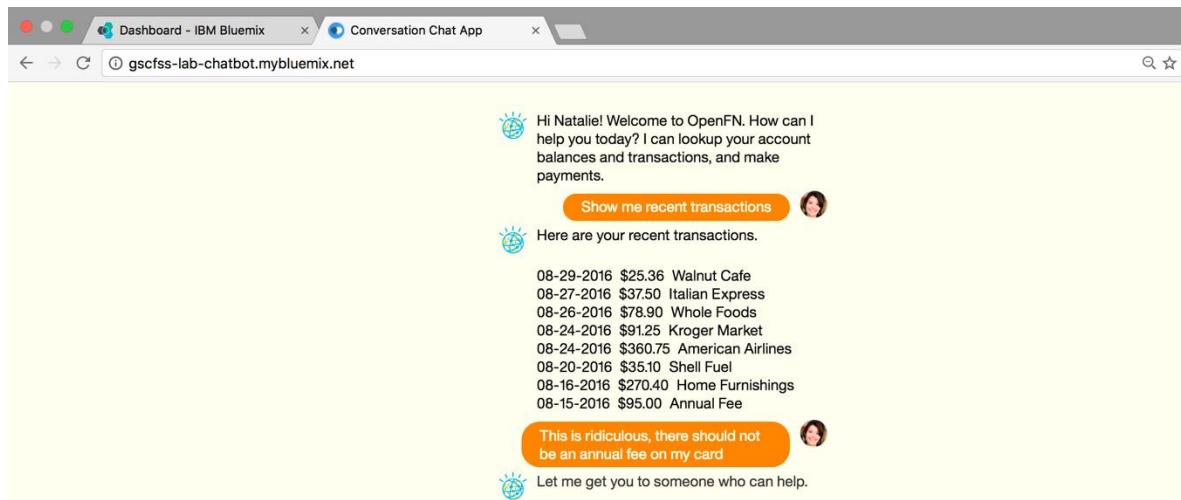
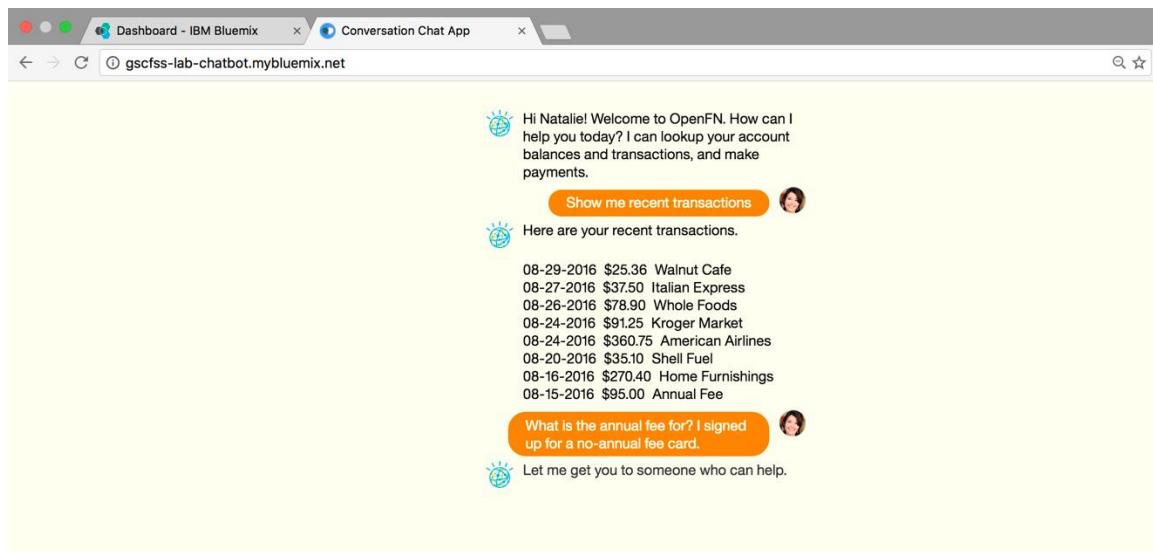
- f) Type (or copy paste) the statement **Show me recent transactions**. There is an **Annual Fee** transaction shown in the list.



2016 Banking and Financial Markets Industry Workshop

- g) Assume that the user is not pleased with the annual fee charges and types in a statement that may be recognized as an expression of anger. You can copy and paste examples provided below that shows the corresponding scores returned from Tone Analyzer. If the score result returned from Tone Analyzer is above the threshold (0.49), you will receive a response as ***Let me get you to someone who can help.*** The receiving application would then use the context of **directToLiveAgent** to redirect the user's chat to a representative (beyond the scope of this lab exercise).

Statement examples	Anger Score
What is the annual fee for? I signed up for a no-annual fee card.	0.68
This is ridiculous, there should not be an annual fee on my card	0.59



Congratulations! You have successfully completed this Lab.