Internship project report on

Intelligent Customer Help Desk with Smart Document Understanding

---

By
Name: - Abhijeet Behera
Email: - bbgiabhijeet12@gmail.com

Category: Artificial Intelligence

Internship under: -

SMARTBRIDGE
Let's Bridge the Gap

TABLE OF CONTENTS

1.Problem Definition	
2.Objectives	
3.Introduction	
4.Software Requirement Specification (SRS)	
4.1.Introduction	
4.1.1Purpose	
4.1.2Scope	
4.1.3 Overview	
4.2 The Overall Description	
4.2.1 System Environment	
4.2.2 Interfaces	
4.3 Specific Requirements	
4.3.1 Functionality of the Web-app(Chatbot)	
4.4 Usability	
4.4.1 Graphical User Interface	
4.4.2 Accessibility	
5. System Design and Analysis	
5.1 Watson Assistant	
5.1.1 Intents	
5.1.2 Entities	
5.1.3 Dialog	
5.2 Watson Discovery	
5.2.1 Smart Document Understanding	
5.3 Node-RED	
5.4 IBM Cloud Functions	
5.4.1 Working with Actions	
6. Limitations	
7. Future Scope	
8. Conclusion	
9. References	
10. Appendix	
10.1 Project Code	
10.2 Sample Output	

1. Problem statement

The typical customer care chatbot can answer simple questions, such as store locations and hours, directions, and maybe even making appointments. When a question falls outside of the scope of the pre-determined question set, the option is typically to tell the customer the question isn't valid or offer to speak to a real person.

In this project, there will be another option. If the customer question is about the operation of a device, the application shall pass the question onto Watson Discovery Service, which has been pre-loaded with the device's owner's manual. So now, instead of "Would you like to speak to a customer representative?" we can return relevant sections of the owner's manual to help solve our customers' problems.

To take it a step further, the project shall use the Smart Document Understanding feature of Watson Discovery to train it on what text in the owner's manual is important and what is not. This will improve the answers returned from the queries.

2. Objective

The main purpose of this project is to develop a chatbot that can solve simple and basic queries posed by many customers or users. AI-powered chatbots have become widely available and let you do what was once impossible: help customers 24/7, automatically resolve questions without any human intervention, and provide support to multiple customers at once.

If the customers face some technical issues regarding a product, then instead of calling for technical support, an AI powered chatbot will be there to provide any kind of assistance. The chatbot is pre-loaded with user manuals and other documents about the operation of the device, so if the customers asks a question about the operation of the device we can return specific or relevant sections from the owner's manual to the customer.

So until and unless the customer specifically asks for customer representative the bot will try to solve all your queries.

3. Introduction

Chatbots are self-help tools for improving communications. Organisations can use it to improve their customer’s experience, to generate more sales and build a deeper rapport with customers. They allow your customers to easily interact with your brand through stimulated conversations. Customers want immediate response to their query. They want to be able to handle challenges on their own until they’re no longer able to do it. Only then will they require assistance from a live chat agent.

At the very least, customers will be happy to receive a welcome message from a chatbot telling them the exact time they will receive solutions to their challenges — especially when the solution has not been programmed to the bot yet. Chatbots are cost-effective customer service solutions for brands in any capacity with a high or low budget.

4. Software Requirement Specification

4.1 Introduction

4.1.1 Purpose

The purpose of the document is to collect and analyze all assorted ideas that have come up to define the web-app, its requirements with respect to the users. Also, we shall predict how we hope this web-app will be used in order to gain a better understanding of the project, outline concepts that may be developed later.

In short, the purpose of this SRS document is to provide a detailed overview of our chatbot (Integrated in a web-app), its parameters and goals. This document describes the project's target audience and its user interface, hardware and software requirements. It defines how our client, team and audience see the product and its functionality. Nonetheless, it helps any designer and developer to assist in software delivery lifecycle (SDLC) processes.

4.1.2 Scope

Primarily, this chatbot is designed to run on the IBM cloud. The First and only thing the user is required to do is to at post their respective queries to the chatbot. The chatbot will answer their queries about the operation of the product.

This chatbot can only provide assistance to problems about the operations of the product Ecobee3 Smarter Wi-Fi Thermostat. In future some more products will be added to the chatbot.

In summary: -

- Create a customer care dialog skill in Watson Assistant
- Use Smart Document Understanding to build an enhanced Watson Discovery collection
- Create an IBM Cloud Functions web action that allows Watson Assistant to post queries to Watson Discovery
- Build a web application with integration to all these services & deploy the same on IBM Cloud Platform

4.1.3 Overview

The remaining sections of this document provide a general description, including characteristics of the project, the product's module, and the functional and data requirements of the system. It also gives the user viewpoint of product. It lists all the functions performed by the system.. The third chapter, specific requirements section, of this document is written primarily for the developers and describes in technical terms the details of the functionality of the system.

4.2 Overall Description

4.2.1 System Environment

The Chatbot is deployed on the IBM cloud which is integrated with different services provided by the IBM cloud such as: -

1. Watson Assistant

   Watson Assistant is a conversation AI platform that helps you provide customers fast, straightforward and accurate answers to their questions, across any application, device or channel
2. Watson Discovery

   Watson Discovery applies the latest breakthroughs in machine learning, including natural language processing capabilities, and is easily trained on the language of your domain
3. Node-Red (To build the web-app UI)

   Node-Red is a flow-based development tool for visual programming developed originally by IBM for wiring together hardware devices, APIs and online services as part of the Internet of Things. Node-RED provides a web browser-based flow editor, which can be used to create JavaScript functions

4. IBM cloud function

4.2.2 Interfaces

The user interface for the software shall be compatible to any browser such as Internet Explorer, Mozilla or Google Chrome by which user can access to the system or the web-app.

This project provides only one UI interface developed in Node-Red. Node-red runs on the platform Node.js and hence basic HTML, CSS and vanilla JavaScript shall be used to build a very simple UI.

4.3.1 Specific Requirements

4.3.1 Functionality

1. The chatbot is integrated in a web-app and it provides a very simple UI.
2. The chatbot allows user to confirm the completion of current configuration.
3. The chatbot allows user to post different queries to the chatbot.
4. The chatbot is integrated with Watson discovery which allows the developers to use a graphical user interface called Smart Document Understanding in order to easily extract specific, clean answers from complicated business documents.
5. The chatbot is also able to refer the customer to a human customer representative if the customers ask for such assistance.

4.4 Usability

4.4.1 Graphical User Interface

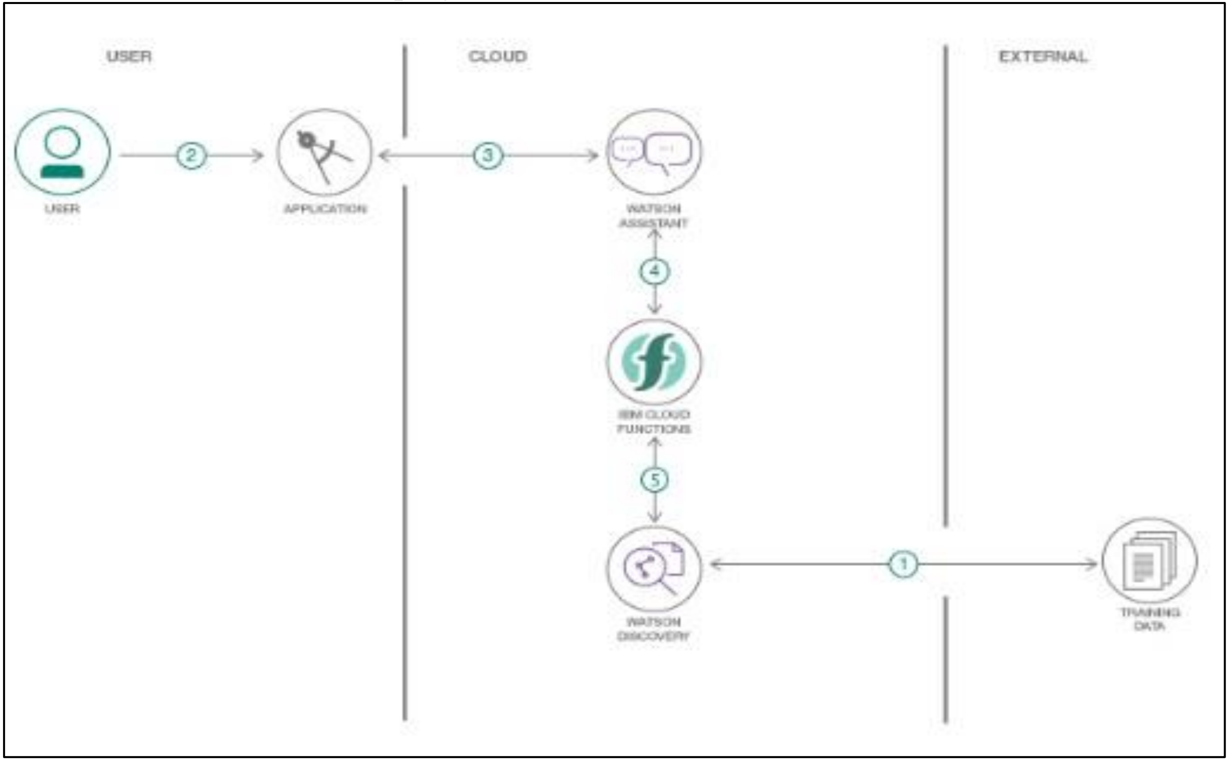
The chatbot shall provide a uniform look and feel. The system shall provide a simple, easy, quick access to the chatbot. The system shall provide use of icons and toolbars.

4.4.2 Accessibility

The system shall provide handicap access. Anyone who has access to internet can have access to this web-app.

5. System Design and Analysis

The basic overflow of the project is shown below: -



1. The document is annotated using Watson Discovery SDU
2. The user interacts with the backend server via the app UI. The fronted app UI is a chatbot that engages the user in a conversation.
3. Dialog between the user and backend server is coordinated using a Watson Assistant dialog skill.
4. If the user asks a product operation question, a search query is passed to a predefined IBM Cloud Functions action.
5. The Cloud Functions action will query the Watson Discovery service and return the results

As the above block diagram shows that the application is connected to the Watson Assistant and Watson Discovery which are IBM cloud services.

So the creation of following services is in the order: -

1. Watson Assistant
2. Watson Discovery
3. IBM cloud action
4. Node-Red

5.1 Watson Assistant

Choose the Watson assistant service from the catalogue, launch the Watson assistant tool and create a new dialog skill. Use sample skill option as your starting point.

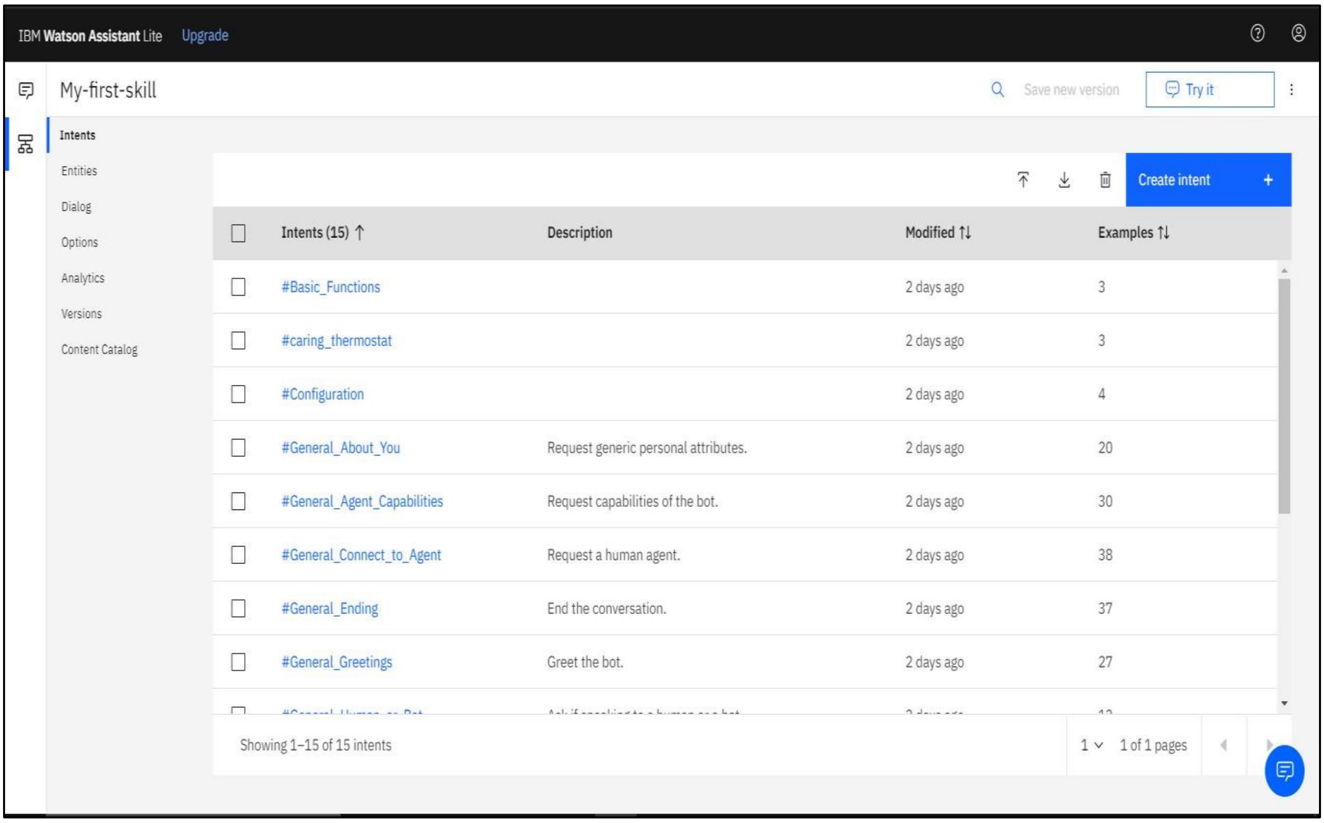
With IBM Watson Assistant, you can build conversational interfaces into any application, device, or channel. Most virtual assistants try to mimic human interactions, but Watson Assistant is more. Watson Assistant knows when to search for an answer from a knowledge base, when to ask for clarity, and when to direct you to a human.

Some basic terminology about using IBM Watson Assistant: -

Term	**Definition**
Skills	A container for the artificial intelligence that enables an assistant to help your customers.
Assistant	Directs requests down the optimal path for solving a customer problem. Add skills so that you’re assistant can provide a direct answer to a common question or reference more generalized search results for something more complex.
Dialog	Defines what your assistant says in response to customers, based on what it believes the customer wants. The dialog flow is represented graphically in the tool as a tree.
Intents	A goal that you anticipate your users will have when they interact with your assistant. For example, you might define an intent that is named store hours that answers questions about store hours.
Entities	A term or object that provides context for intent. For example, an entity might be a city name that helps your dialog to distinguish which store the user wants to know store hours for.
Content Catalog	An easy way to add common intents to your Watson Assistant dialog.

5.1.1 Intents

Intents are purposes or goals that are expressed in a customer's input, such as answering a question or processing a bill payment. By recognizing the intent expressed in a customer's input, the Watson Assistant service can choose the correct dialog flow for responding to it.



☐	Intents (15) ↑	Description	Modified ↑↓	Examples ↑↓
☐	#Basic_Functions		2 days ago	3
☐	#caring_thermostat		2 days ago	3
☐	#Configuration		2 days ago	4
☐	#General_About_You	Request generic personal attributes.	2 days ago	20
☐	#General_Agent_Capabilities	Request capabilities of the bot.	2 days ago	30
☐	#General_Connect_to_Agent	Request a human agent.	2 days ago	38
☐	#General_Ending	End the conversation.	2 days ago	37
☐	#General_Greetings	Greet the bot.	2 days ago	27

Showing 1–15 of 15 intents

1 ⌄ 1 of 1 pages

5.1.2 Entities

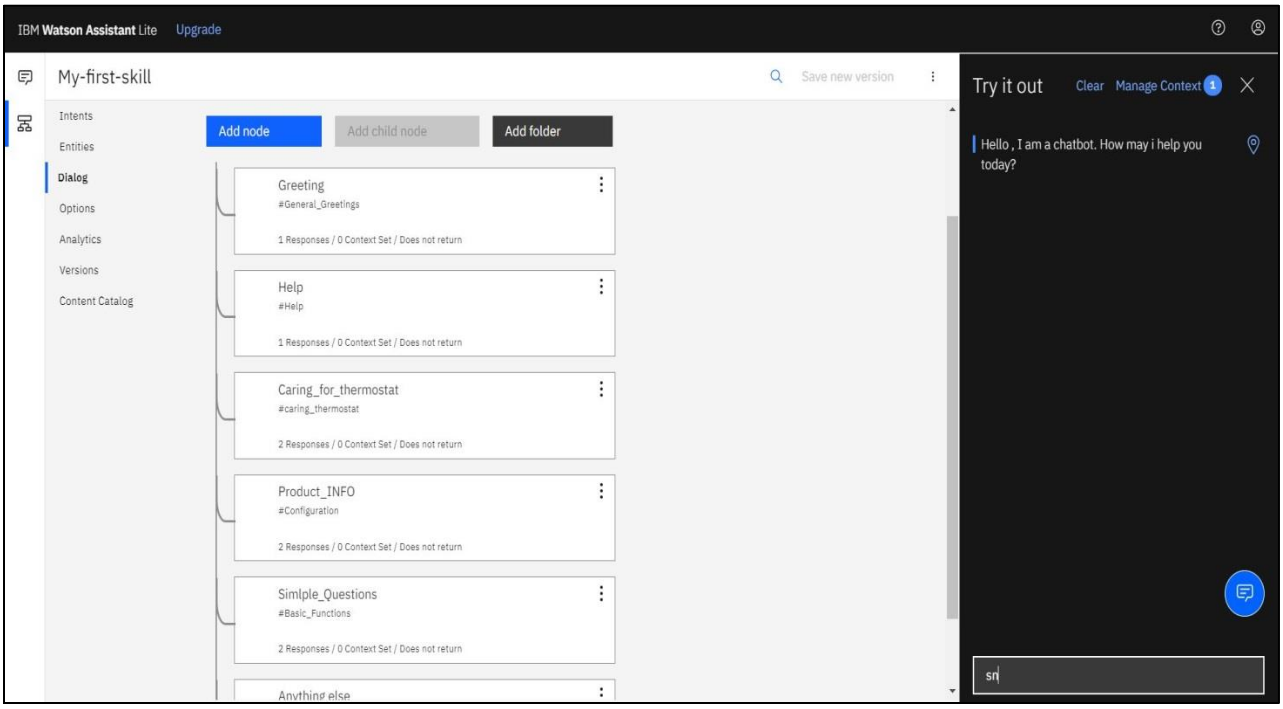
Entities represent information in the user input that is relevant to the user's purpose.

If intents represent verbs (the action a user wants to do), entities represent nouns (the object of, or the context for, that action). For example, when the *intent* is to get a weather forecast, the relevant location and date *entities* are required before the application can return an accurate forecast.

Recognizing entities in the user's input helps you to craft more useful, targeted responses. For example, you might have a `#buy_something` intent. When a user makes a request that triggers the `#buy_something` intent, the assistant's response should reflect an understanding of what the *something* is that the customer wants to buy. You can add a `@product` entity, and then use it to extract information from the user input about the product that the customer is interested in. (The `@` prepended to the entity name helps to clearly identify it as an entity.)

5.1.3 Dialog

The dialog defines what your assistant says in response to customers.



Hello , I am a chatbot. How may i help you today?

sn

A node condition determines whether that node is used in the conversation. Response conditions determine which response to return to a user.

The dialog response defines how to reply to the user. You can reply in the following ways:

- Simple text response
- Rich responses
- Conditional responses

5.2 Watson Discovery

With IBM Watson Discovery, you can ingest, normalize, enrich, and search your unstructured data (JSON, HTML, PDF, Word, and more) with speed and accuracy. It packages core Watson APIs such as Natural Language Understanding and Document Conversion along with UI tools that enable you to easily upload, enrich, and index large collections of private or public data.

A common way to use Discovery is by accessing the Discovery APIs from your application. The Watson team releases SDKs that support many programming languages so that you can use Discovery easily in a web or mobile application. All of the data content is stored and enriched within a Discovery collection. The data does not require any specific structure and can come from multiple public and private data sources. Every Discovery environment comes with a pre-enriched data collection named Watson Discovery News.

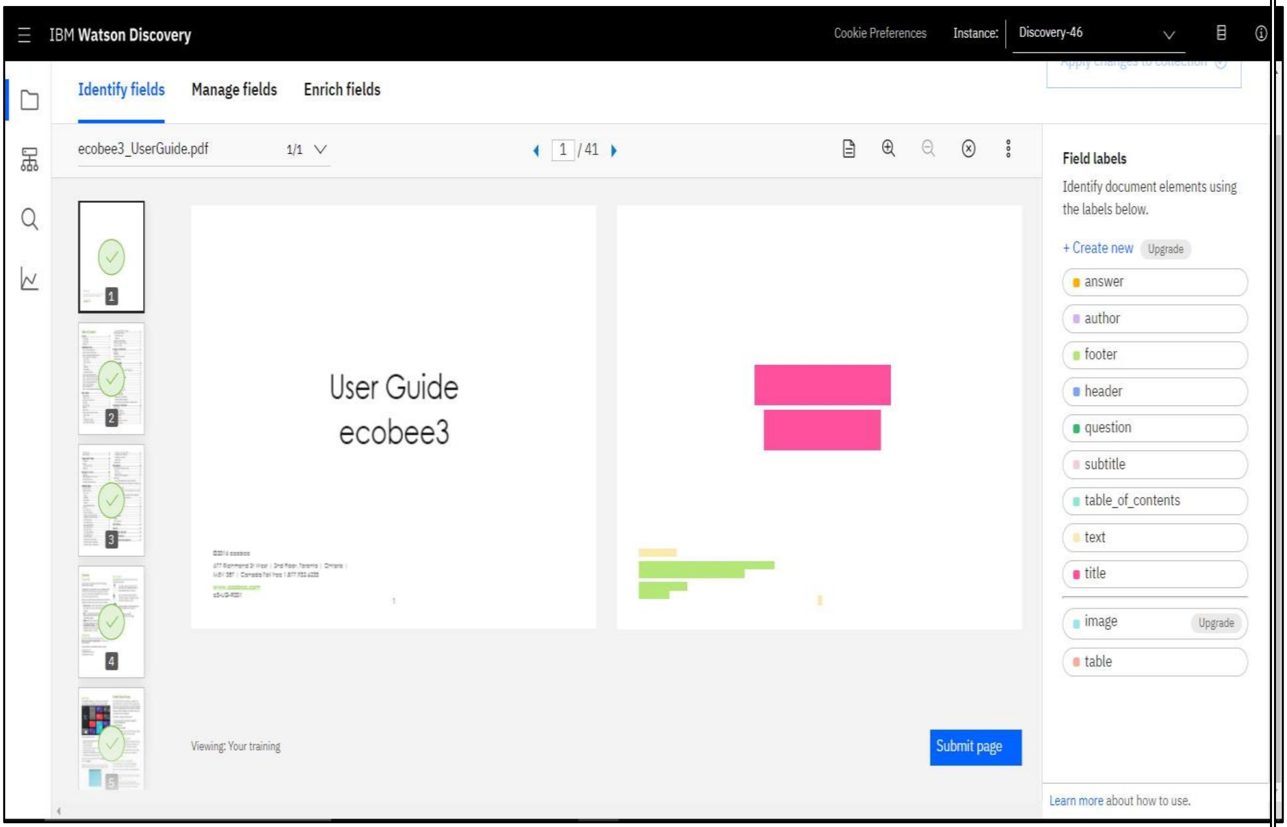
5.2.1 Smart Document Understanding

We will use the Smart Document Understanding feature of Watson Discovery to train it on what text in the owner’s manual is important and what is not. This will improve the answers returned from the queries.

Import the document (Ecobee3 thermostat manual) into the discovery.

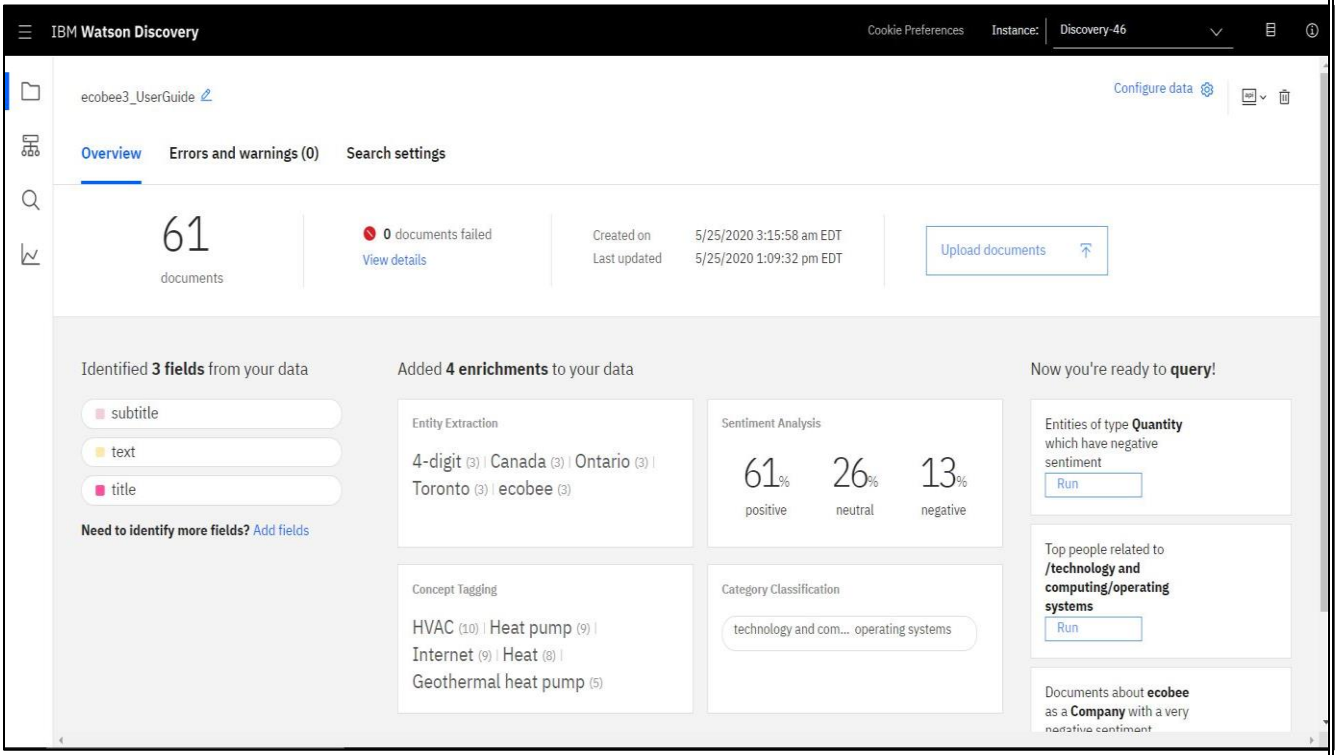
1. The document is annotated using Watson Discovery Smart Document Understanding.
2. The user interacts with the back-end server via the app UI. The front-end app UI is a chatbot that engages the user in a conversation.
3. Dialog between the user and back-end server is coordinated using a Watson Assistant dialog skill.
4. If the user asks a product operation question, a search query is passed to a predefined IBM Cloud Functions action.
5. The IBM Cloud Functions action will query the Watson Discovery Service and return the results.

The goal is to annotate all of the pages in the document so Discovery can learn what text is important, and what text can be ignored.
As you go though the annotations one page at a time, Discovery is learning and should start automatically updating the upcoming pages. Once you get to a page that is already correctly annotated, you can stop, or simply click Submit to acknowledge it is correct. The more pages you annotate, the better the model will be trained



Identify document elements using the labels below.

Learn more about how to use.



61 documents

0 documents failed
View details

Created on 5/25/2020 3:15:58 am EDT
Last updated 5/25/2020 1:09:32 pm EDT

Identified **3 fields** from your data

- subtitle
- text
- title

Need to identify more fields? Add fields

Added **4 enrichments** to your data

Entity Extraction
4-digit (3) | Canada (3) | Ontario (3) | Toronto (3) | ecobee (3)

Sentiment Analysis
61% positive
26% neutral
13% negative

Concept Tagging
HVAC (10) | Heat pump (9) | Internet (9) | Heat (8) | Geothermal heat pump (5)

Category Classification
technology and com... operating systems

Now you're ready to **query**!

Entities of type **Quantity** which have negative sentiment

Top people related to **/technology and computing/operating systems**

Documents about **ecobee** as a **Company** with a very negative sentiment

5.3 Node-RED

Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways.

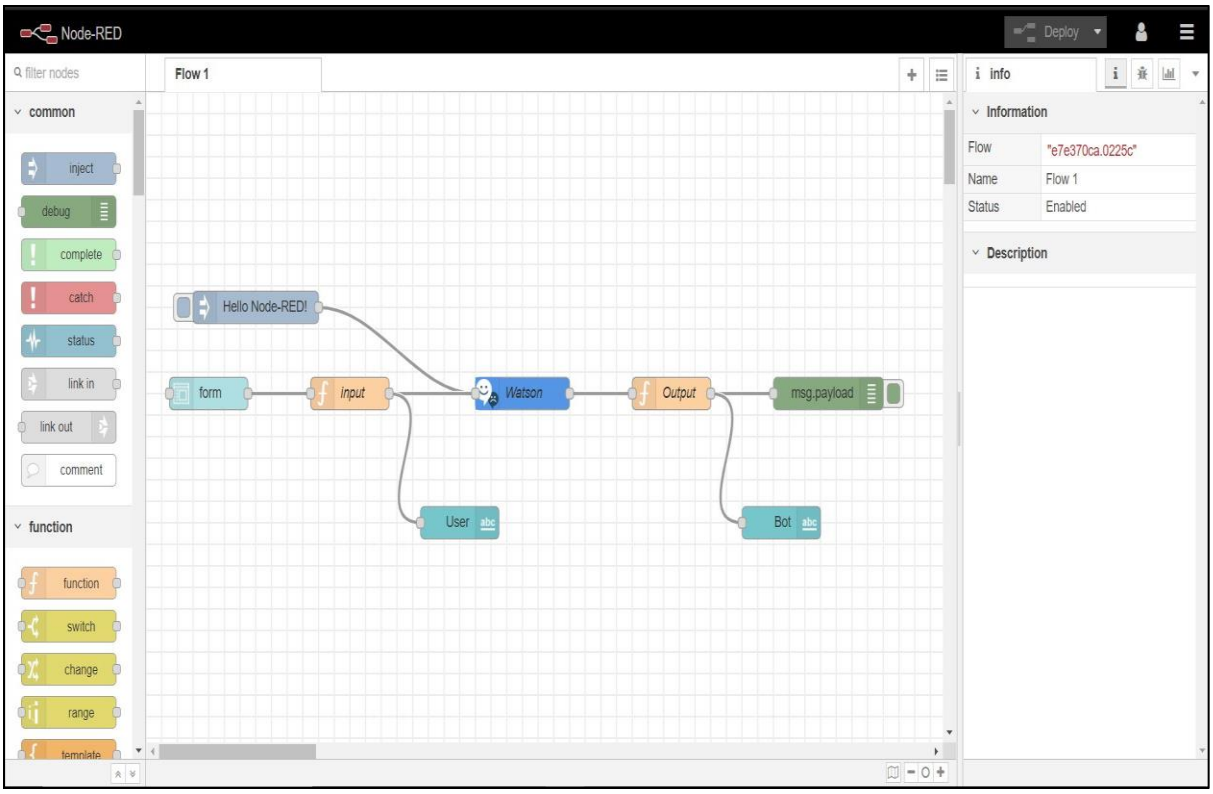
It provides a browser-based editor that makes it easy to wire together flows using the wide range of nodes in the palette that can be deployed to its runtime in a single-click.

Node-RED provides a browser-based flow editor that makes it easy to wire together flows using the wide range of nodes in the palette. Flows can be then deployed to the runtime in a single-click. JavaScript functions can be created within the editor using a rich text editor. A built-in library allows you to save useful functions, templates or flows for re-use. Node-RED is built on Node.js, taking full advantage of its event-driven, non-blocking model. This makes it ideal to run at the edge of the network on low-cost hardware such as the Raspberry Pi as well as in the cloud.

Steps

1. Find the Node-RED Starter in the IBM Cloud catalog
2. Create your application

   Insert the following nodes into the flow in Node-RED.
   - Inject
   - Debug
   - ui_Form
   - ui_Text
   - ui_Button
   - Function
   - Switch
   - Assistant
3. Enable the Continuous Delivery feature
4. Open the Node-RED application
5. Configure your Node-RED application
6. Add extra nodes to your Node-RED palette



Flow "e7e370ca.0225c"
Name Flow 1
Status Enabled

5.4 IBM Cloud Functions

With IBM Cloud™ Functions you can use your favourite programming language to write lightweight code that runs app logic in a scalable way. You can run code on-demand with HTTP-based API requests from applications or run code in response to IBM Cloud services and third-party events. The Function-as-a-Service (Faas) programming platform is based on the open source project Apache OpenWhisk.

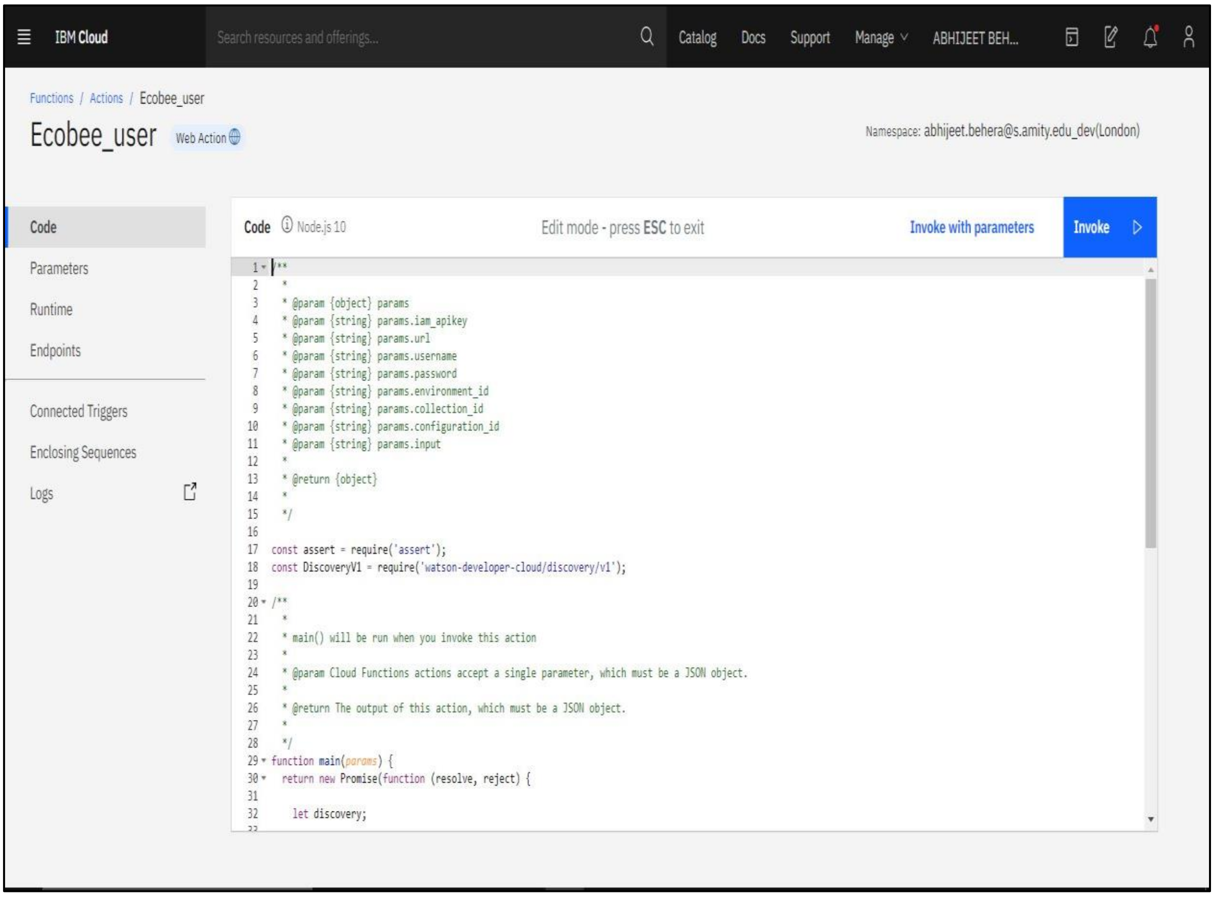
5.4.1 Working with Cloud Functions

With IBM Cloud Functions, you can create stateless code snippets, called actions that are set to perform one specific task.

Creating an action in the GUI .To get started with Cloud Functions, try creating the Hello World quickstart template.

1. Create an IBM Cloud account or log in to an existing account.
2. Navigate to the Cloud Functions dashboard.
3. Click Start Creating > Quickstart Templates and select the Hello World template.
4. Create a package for your actions by inputting a unique name in the Package Name field.

5. Select a runtime from the drop-down menu in the Action helloworld section. You can preview the code for the sample action in each available runtime before you deploy the template.
6. Click Deploy. You created an action. Great job!
7.Run the action by clicking Invoke. Invoking an action manually runs the app logic that the action defines. In the Activations panel, you can see the "Hello stranger!" greeting produced by the action.

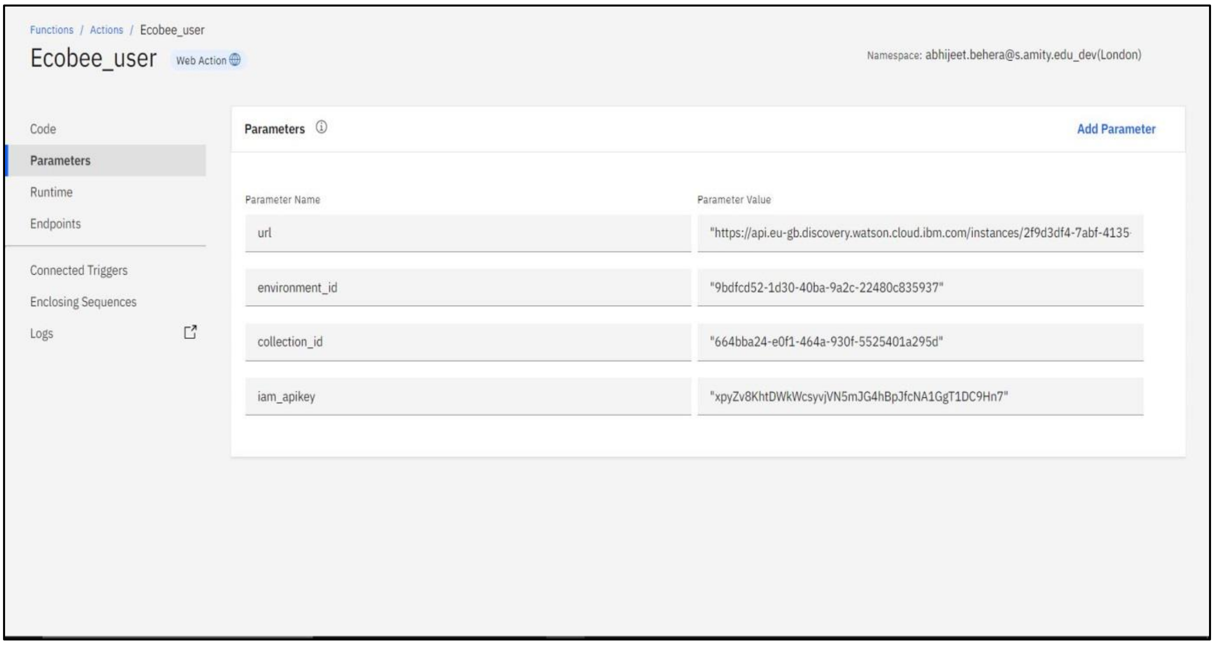


Namespace: abhijeet.behera@s.amity.edu_dev(London)

```
/**
  *
  * @param {object} params
  * @param {string} params.iam_apikey
  * @param {string} params.url
  * @param {string} params.username
  * @param {string} params.password
  * @param {string} params.environment_id
  * @param {string} params.collection_id
  * @param {string} params.configuration_id
  * @param {string} params.input
  *
  * @return {object}
  *
  */

const assert = require('assert');
const DiscoveryV1 = require('watson-developer-cloud/discovery/v1');

/**
  *
  * main() will be run when you invoke this action
  *
  * @param Cloud Functions actions accept a single parameter, which must be a JSON object.
  *
  * @return The output of this action, which must be a JSON object.
  *
  */
function main(params) {
  return new Promise(function (resolve, reject) {

    let discovery;
```

Add the following keys:

- url
- environment_id
- collection_id
- iam_apikey



Namespace: abhijeet.behera@s.amity.edu_dev(London)

Parameter Name	Parameter Value
url	"https://api.eu-gb.discovery.watson.cloud.ibm.com/instances/2f9d3df4-7abf-4135
environment_id	"9bdfcd52-1d30-40ba-9a2c-22480c835937"
collection_id	"664bba24-e0f1-464a-930f-5525401a295d"
iam_apikey	"xpyZv8KhtDWkWcsyvjVN5mJG4hBpJfcNA1GgT1DC9Hn7"

6. Limitations

1. Chatbots cannot fully replace humans, but they should complement each other. So an agent should interact with a customer in second instance but already having all necessary information getting from the bot.
2. It is often considered that chatbot are complicated and need a lot of time to understand what you want in customer. Sometimes, it can also annoy the client about their slowness, or their difficulty in filtering responses.
3. If a query doesn’t relate to something you’ve previously « taught » it, you won’t understand it. This can lead to a frustrated customer and the loss of the sale. Other times they do understand you, but they need double (or triple) as many messages as one person, which spoils the user experience.
4. Chatbots are installed with the aim of speeding up responses and improving customer interaction. However, due to the limited availability of data and the time needed for self-updating, this process can be slow and costly.

7. Future Scope

Linguistic and conversational ability of the chatbot can be improved by increasing the no of intents and entities and also updating the dialog. This will allow the bot to mimic human interactions which will make the customers feel at ease.

If the future demands advanced chatbots that do more than use scripted, single-turn exchanges, then their method of interface will also have to advance. A voice interface can assist users with disabilities or those who are skeptical of technology, but it also requires another layer of NLP development.

We can include Watson studio text to speech and speech to text services to access the chatbot handsfree. This is one of the future scopes of this project

8. Conclusion

With the help of services such as Watson assistant, Watson discovery, node-RED and IBM cloud functions an Intelligent Customer Help Desk (Chatbot) with smart document understanding is created.

The chatbot was successfully tested and deployed on the IBM cloud.

9. References

1. Nick O' Leary published February 4, 2020 "*Node-Red Starter Application*"- https://developer.ibm.com/tutorials/how-to-create-a-node-red-starter-application/
   Node-Red Documentation: - https://nodered.org/docs/user-guide/
2. Scott Dangelo published August 6, 2019 "*Introduction to Watson Assistant*" - https://developer.ibm.com/articles/introduction-watson-assistant/
3. Rima iBRAHIM published Nov 8, 2019 "*Node-Red Watson Discovery Chatbot Telegram*" - https://medium.com/@rimaibrahim/node-red-watson-discovery-chatbot-telegram-ce616ddcd0d9
4. Y Chen, JDE Argentinis, G Weber - Clinical therapeutics, 2016 – Elsevier "*IBM Watson: How Cognitive Computing Can Be Applied to Big Data Challenges in Life Sciences Research*" https://doi.org/10.1016/j.clinthera.2015.12.001

10. Appendix

10.1 Project Code

Index.js

```
/**
         * Copyright 2017, 2019 IBM Corp.
         *
         * Licensed under the Apache License, Version 2.0 (the "License");
         * you may not use this file except in compliance with the License.
         * You may obtain a copy of the License at
         *
         * http://www.apache.org/licenses/LICENSE-2.0
         *
         * Unless required by applicable law or agreed to in writing, software
         * distributed under the License is distributed on an "AS IS" BASIS,
         * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
         * See the License for the specific language governing permissions and
         * limitations under the License.
         **/
        var bcrypt = require("bcrypt");
        var util = require("util");
        var path = require("path");
        util.log("Starting Node-RED on IBM Cloud bootstrap");
        util.log("Loading bluemix-settings.js");
        var settings = require("./bluemix-settings.js");
        if (!settings.adminAuth) {
            // No user-defined security
            var storage;
            if (settings.storageModule) {
                storage = settings.storageModule;
            } else {
                storage = require('./node_modules/@node-
        red/runtime/lib/storage/localfilesystem');
            }
            util.log("Loading application settings");
            storage.init(settings).then(storage.getSettings).then(runtimeSettings => {
                if (process.env.NODE_RED_USERNAME && process.env.NODE_RED_PASSWORD) {
                    util.log("Enabling adminAuth using
        NODE_RED_USERNAME/NODE_RED_PASSWORD");
                    var config = {
                        adminAuth: {
                            username: process.env.NODE_RED_USERNAME,
                            password: bcrypt.hashSync(process.env.NODE_RED_PASSWORD, 8),
                            allowAnonymous: (process.env.NODE_RED_GUEST_ACCESS === 'true')
                        }
                    };
                    if (runtimeSettings.bluemixConfig &&
        runtimeSettings.bluemixConfig.hasOwnProperty('adminAuth')) {
                        delete runtimeSettings.bluemixConfig.adminAuth;
                        storage.saveSettings(runtimeSettings).then(() => {
                            startNodeRED(config);
                        });
                    } else {
                        startNodeRED(config);
                    }
                } else if (runtimeSettings.bluemixConfig) {
                    util.log("Using runtime settings for adminAuth");
                    startNodeRED(runtimeSettings.bluemixConfig);
                } else {
                    util.log("Starting first-use setup");
                    var server;
                    var express = require('express');
                    var bodyParser = require('body-parser');
                    var app = express();
                    app.use(bodyParser.json());
                    app.get("/", function (req, res) {
                        res.sendFile(path.join(__dirname, "public", "first-run.html"));
                    });
                    app.post("/setup", function (req, res) {
                        if (req.body.adminAuth && req.body.adminAuth.password) {
                            req.body.adminAuth.password =
        bcrypt.hashSync(req.body.adminAuth.password, 8);
                        }
                        runtimeSettings.bluemixConfig = req.body;
                        util.log("Received first-use setup configuration");
                        storage.saveSettings(runtimeSettings).then(() => {
                            res.status(200).end();
                            setTimeout(function () {
                                util.log("Stopping first-use setup application");
                                server.shutdown(() => {
                                    startNodeRED(req.body);
                                });
                            }, 1000);
                        }).catch(err => {
                            util.log("Failed to save configuration");
                            util.log(err);
                            res.status(200).end();
                        });
                    });
                    app.use("/", express.static(path.join(__dirname, "public")));
                    require('./routers/health')(app);
                    var http = require('http');
                    server = http.createServer(function (req, res) { app(req, res); });
                    server = require('http-shutdown')(server);
                    server.listen(settings.uiPort, settings.uiHost, function () { });
                    util.log("Waiting for first-use setup to complete");
                }
            }).catch(err => {
                console.log("Failed to initialize storage module");
                console.log(err);
            });
        } else {
            startNodeRED({});
        }
        function startNodeRED(config) {
            if (config.adminAuth && !settings.adminAuth) {
                util.log("Enabling adminAuth security - set NODE_RED_USERNAME and
        NODE_RED_PASSWORD to change credentials");
                settings.adminAuth = {
                    type: "credentials",
                    users: function (username) {
                        if (config.adminAuth.username == username) {
                            return Promise.resolve({ username: username, permissions: "*"
        });
                        } else {
                            return Promise.resolve(null);
                        }
                    },
                    authenticate: function (username, password) {
                        if (config.adminAuth.username === username &&
        bcrypt.compareSync(password, config.adminAuth.password)) {
                            return Promise.resolve({ username: username, permissions: "*"
        });
                        } else {
                            return Promise.resolve(null);
                        }
                    }
                };
                if ((process.env.NODE_RED_GUEST_ACCESS === 'true') ||
        (process.env.NODE_RED_GUEST_ACCESS === undefined &&
        config.adminAuth.allowAnonymous)) {
                    util.log("Enabling anonymous read-only access - set
        NODE_RED_GUEST_ACCESS to 'false' to disable");
                    settings.adminAuth.default = function () {
                        return Promise.resolve({ anonymous: true, permissions: "read" });
                    };
                } else {
                    util.log("Disabled anonymous read-only access - set
        NODE_RED_GUEST_ACCESS to 'true' to enable");
                }
            }
            require('./node_modules/node-red/red.js');
        }
```

Watson Assistant JSON Code

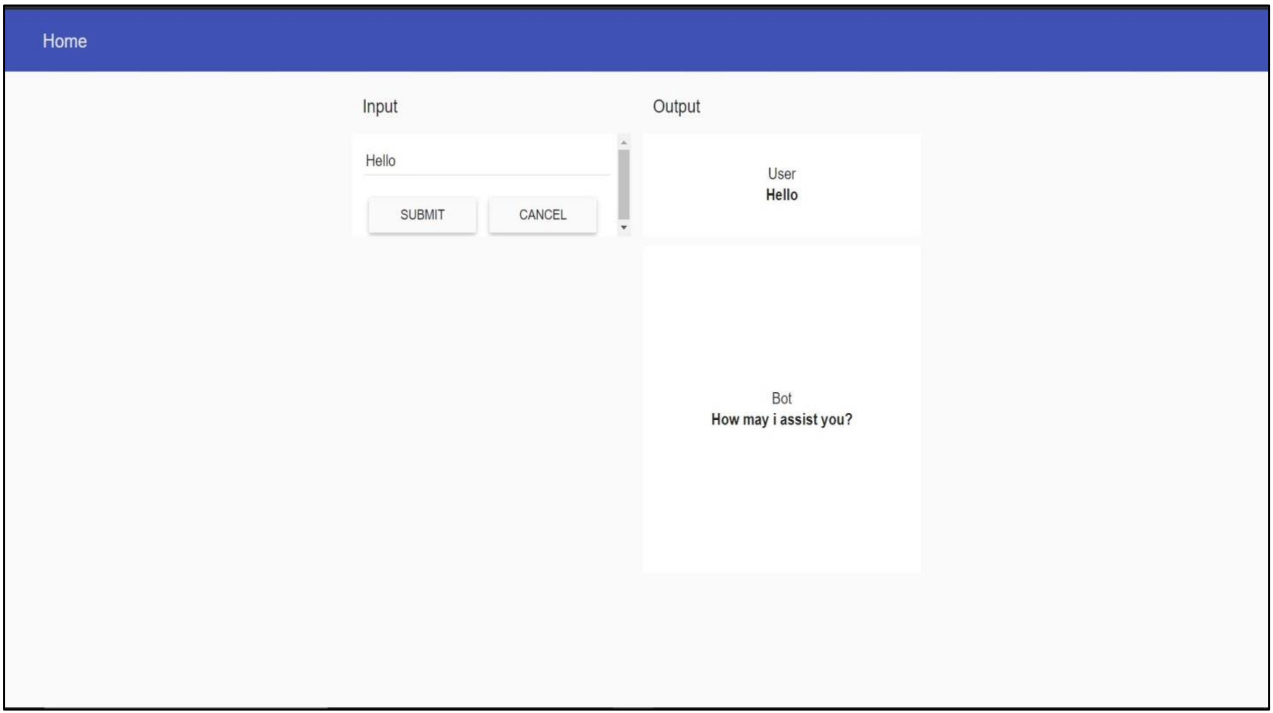
```
{ {
      "intent": "Help",
      "examples": [
        {
          "text": "Help me my thermostat is not working."
        },
        {
          "text": "My thermostat stopped working."
        },
        {
          "text": "Please help , my thermostat died."
        }
      ],
      "description": ""
    },
    {
      "intent": "Configuration",
      "examples": [
        {
          "text": "I want to configure my heat pump"
        },
        {
          "text": "How do i configure wires in the ecobee ?"
        },
        {
          "text": "Tell me how can i configure wifi settings ?"
        },
        {
          "text": "How can i name my thermostat ?"
        }
      ]
"description": ""
```

```
      },
    {
       "intent": "caring_thermostat",
       "examples": [
         {
          "text": "How do i reset my thermostat?"
         },
         {
          "text": "How should i reboot my thermostat?"
         },
         {
          "text": "How can i clean my thermostat?"
         }
       ],
       "description": ""
      },
    {
       "intent": "Greetings",
       "examples": [
         {
          "text": "Hey there"
         },
         {
          "text": "Hi"
         },
         {
          "text": "Good Morning"
         },
         {
          "text": "Hey man"
         },
         {
          "text": "Hello"
         }
       ],
       "description": ""
      },
    {
      "intent": "Basic_Functions",
      "examples": [
        {
          "text": "How can i see the main menu ?"
        },
        {
          "text": "I want to see the main menu"
        },
        {
          "text": "Does it show local weather forecast?"
        }
      ],
      "description": ""
    },
"webhooks": [
    {
      "url": "https://eu-gb.functions.cloud.ibm.com/api/v1/web/abhijeet.behera%40s.amity.edu_dev/default/Ecobee_user.json",
      "name": "main_webhook",
      "headers": []
    }
  ],
  "dialog_nodes": [
    {
      "type": "response_condition",
      "parent": "node_6_1590429972718",
      "conditions": "anything_else",
      "dialog_node": "response_2_1590430009337",
      "previous_sibling": "response_3_1590430006279"
    },
    {
      "type": "response_condition",
      "output": {
        "generic": [
          {
```

```
        "values": [
          {
            "text": "\"<?$webhook_result_2.results[0].text?>\""
          }
        ],
        "response_type": "text",
        "selection_policy": "sequential"
      }
    ]
  },
  "parent": "node_6_1590429972718",
  "conditions": "$webhook_result_2",
  "dialog_node": "response_3_1590430006279"
},
{
  "type": "response_condition",
  "output": {
    "text": {
      "values": [
        "Can you please try again ? Try to be clear and precise."
      ],
      "selection_policy": "sequential"
    }
  },
  "parent": "node_10_1590427102304",
  "conditions": "anything_else",
  "dialog_node": "response_7_1590428189442",
  "previous_sibling": "response_2_1590428185792"
},
{
  "type": "response_condition",
  "output": {
    "generic": [
      {
        "values": [
          {
            "text": "\"<?$webhook_result_1.results[0].text?>\""
```

```
              }
            ],
            "response_type": "text",
            "selection_policy": "sequential"
          }
        ]
      },
      "parent": "node_10_1590427102304",
      "conditions": "$webhook_result_1",
      "dialog_node": "response_2_1590428185792"
    },
    {
      "type": "response_condition",
      "parent": "node_9_1590430831153",
      "conditions": "anything_else",
      "dialog_node": "response_7_1590431128980",
      "previous_sibling": "response_1_1590431125280"
    },
    {
      "type": "response_condition",
      "output": {
        "generic": [
          {
            "values": [
              {
                "text": "\"<?$webhook_result_3.results[0].text?>\""
              }
            ],
            "response_type": "text",
            "selection_policy": "sequential"
          }
        ]
      }
```

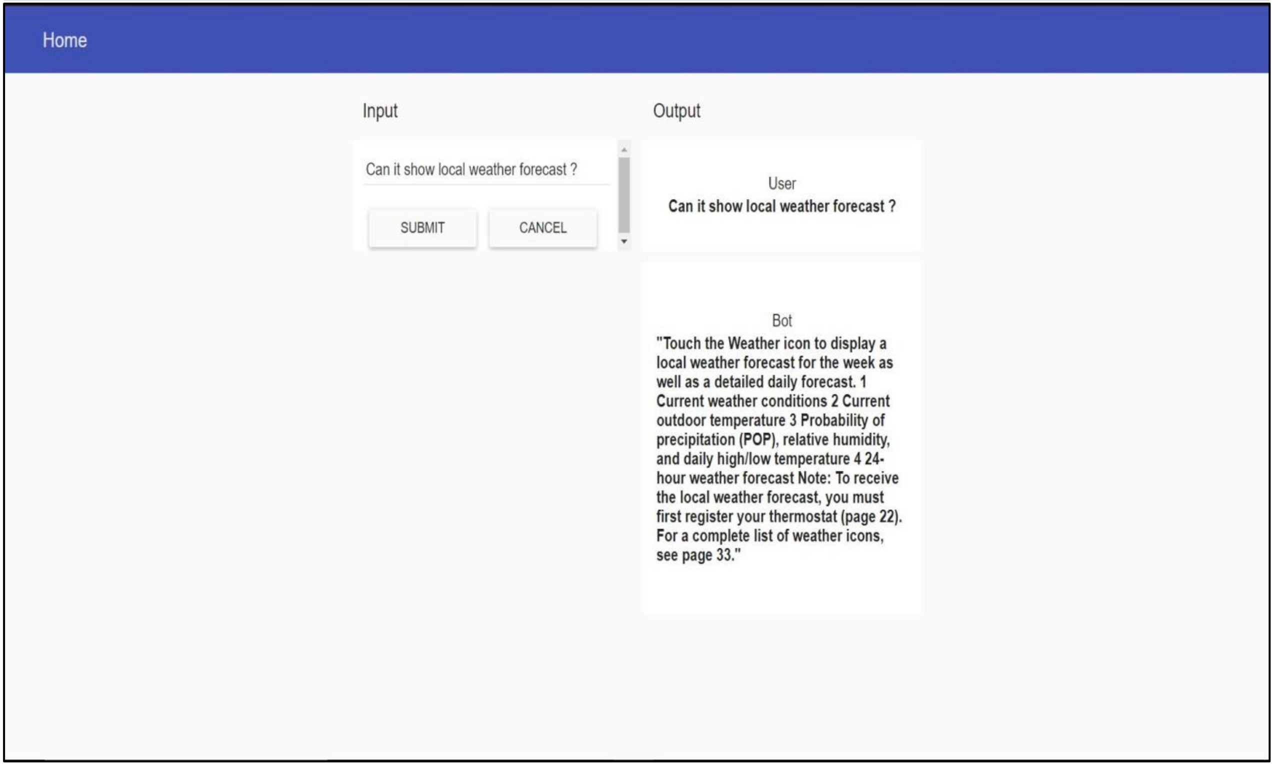
10.2 Sample Outputs



Hello

Hello

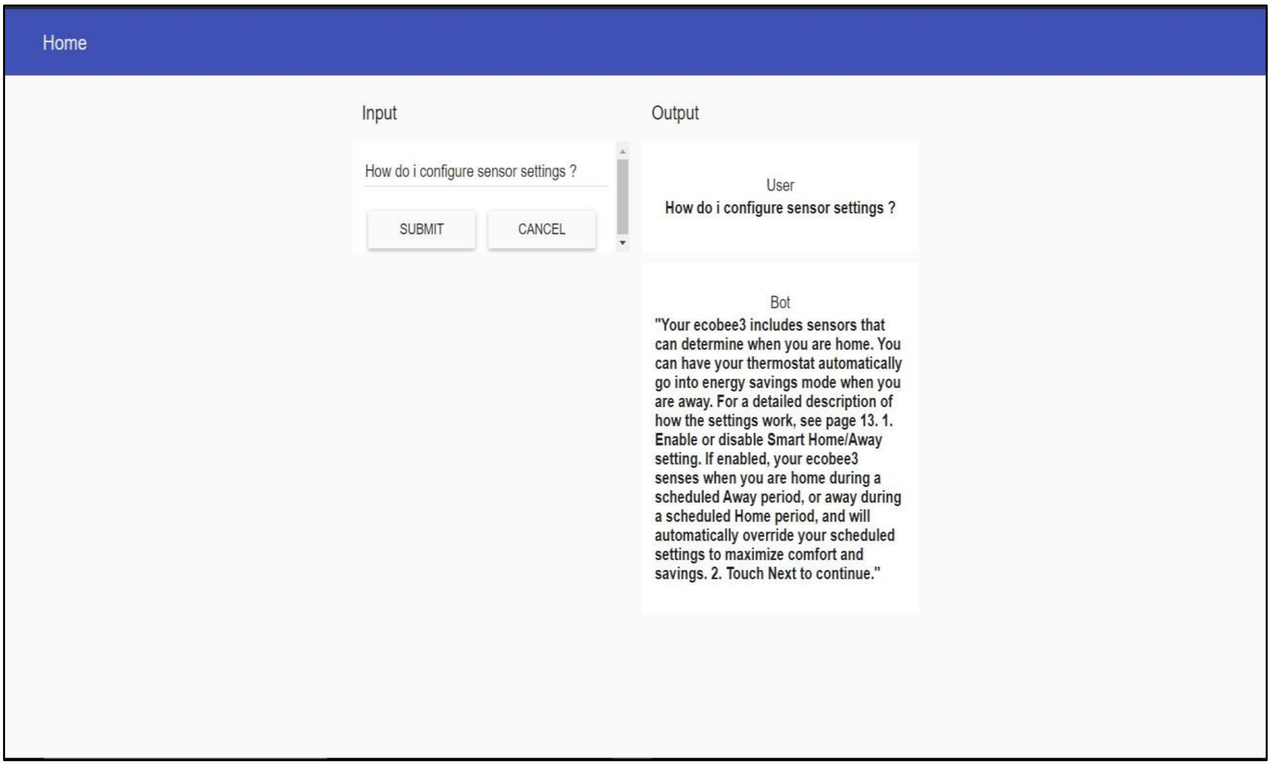
How may i assist you?



Can it show local weather forecast ?

Can it show local weather forecast ?

"Touch the Weather icon to display a local weather forecast for the week as well as a detailed daily forecast. 1 Current weather conditions 2 Current outdoor temperature 3 Probability of precipitation (POP), relative humidity, and daily high/low temperature 4 24-hour weather forecast Note: To receive the local weather forecast, you must first register your thermostat (page 22). For a complete list of weather icons, see page 33."



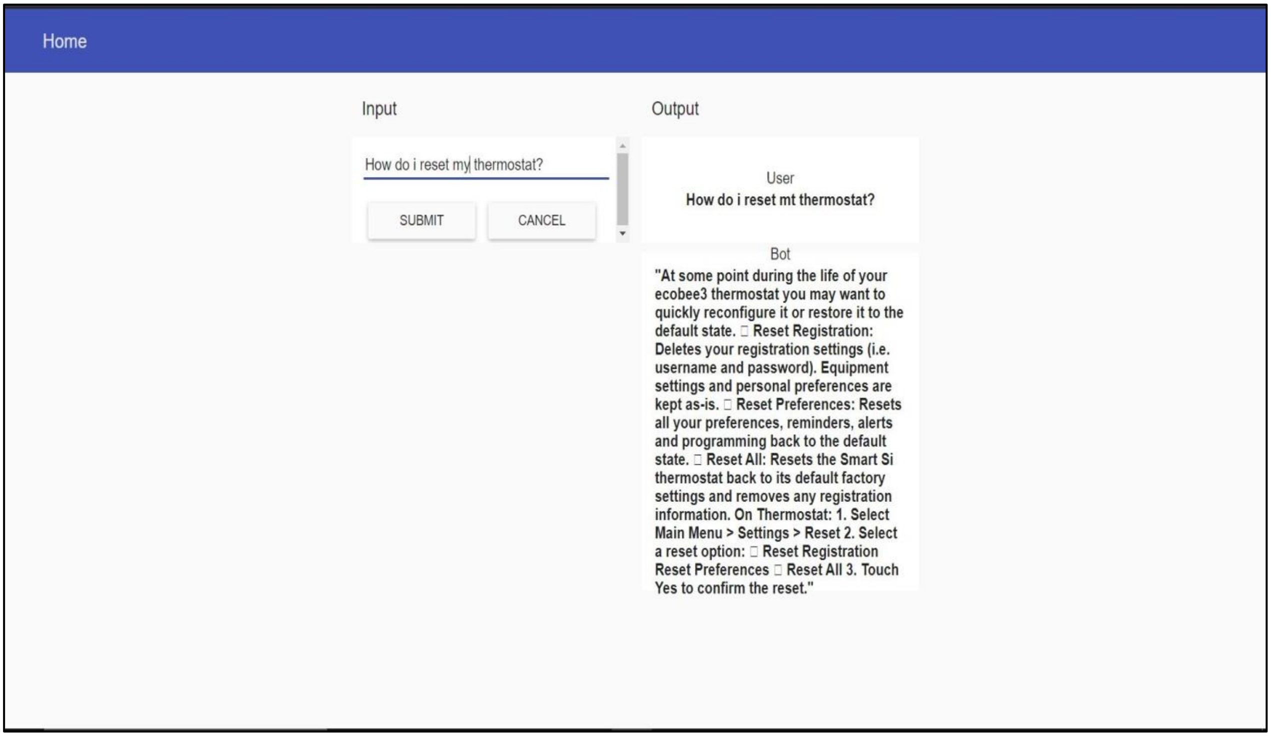
How do i configure sensor settings ?

User

How do i configure sensor settings ?

Bot

"Your ecobee3 includes sensors that can determine when you are home. You can have your thermostat automatically go into energy savings mode when you are away. For a detailed description of how the settings work, see page 13. 1. Enable or disable Smart Home/Away setting. If enabled, your ecobee3 senses when you are home during a scheduled Away period, or away during a scheduled Home period, and will automatically override your scheduled settings to maximize comfort and savings. 2. Touch Next to continue."



How do i reset my thermostat?

User

How do i reset mt thermostat?

Bot

"At some point during the life of your ecobee3 thermostat you may want to quickly reconfigure it or restore it to the default state. □ Reset Registration: Deletes your registration settings (i.e. username and password). Equipment settings and personal preferences are kept as-is. □ Reset Preferences: Resets all your preferences, reminders, alerts and programming back to the default state. □ Reset All: Resets the Smart Si thermostat back to its default factory settings and removes any registration information. On Thermostat: 1. Select Main Menu > Settings > Reset 2. Select a reset option: □ Reset Registration Reset Preferences □ Reset All 3. Touch Yes to confirm the reset."