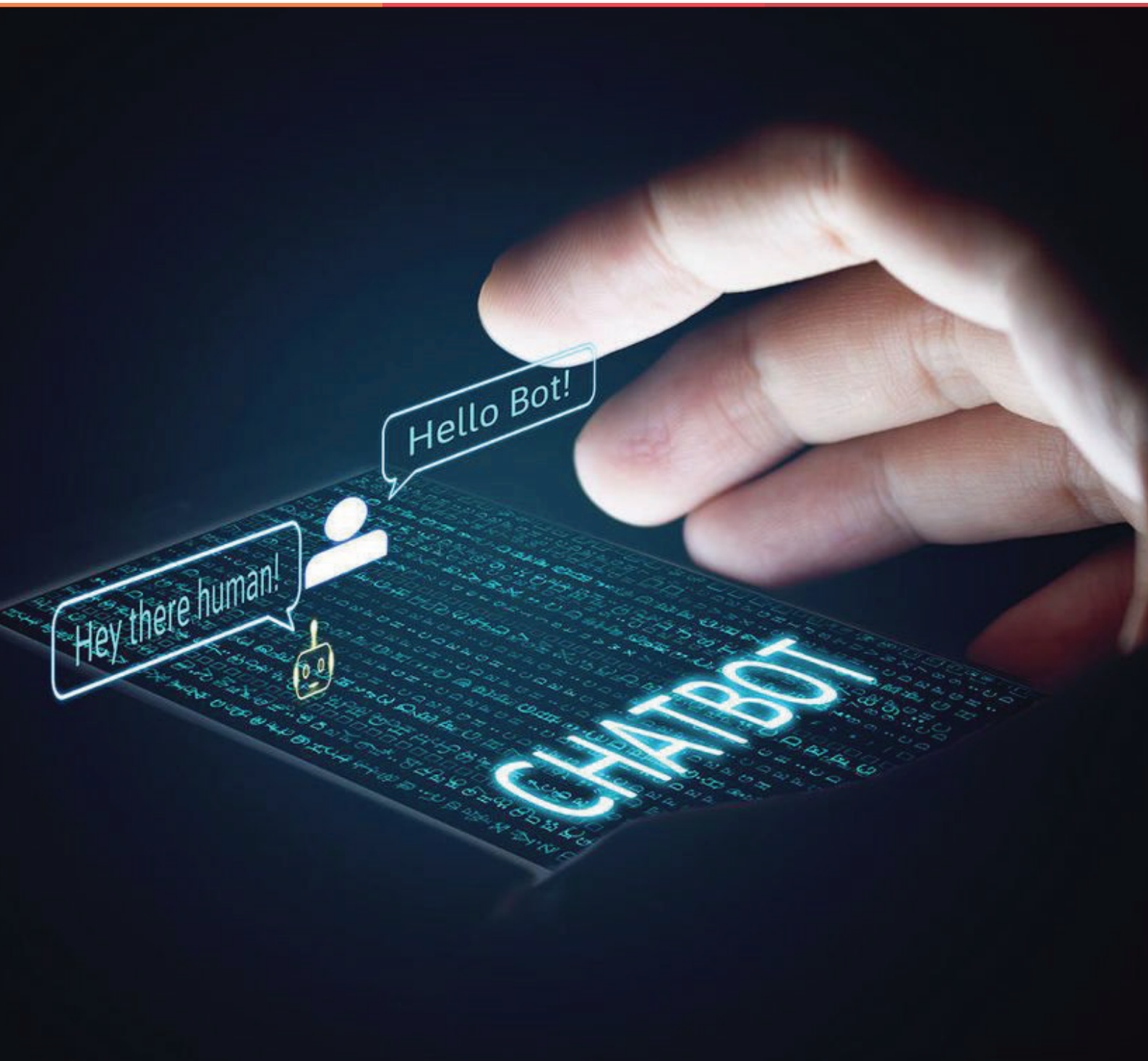
mobolutions
Enterprise Sustainability Simplified.

Redefine customer communication with Chatbot - SAP Plant Maintenance On-premise integration



Introduction

Chatbots are integrated into almost every application that we see today. But what exactly is a chatbot and how does it work? Simply stated, a chatbot is artificial intelligence (AI) software that can simulate a talk with a user in a language that the user understands via messaging, mobile or website applications or even through a telephone. Chatbots leverage natural language processing (NLP) to compose responses to questions in a natural language.

Several enterprises' end-user applications are applying chatbot software to take the user experience to the next level. Among them, ERP applications, notably SAP is taking a lead in integrating chatbot software to generate a seamless interactive communication system.
Though Cloud-based SAP applications are still ruling the world, however, there's still a small chunk of organizations, which are reluctant to move their applications to the cloud, considering the fear of data breaches and misuse. It is in this context that some of the organizations are still relying on on-premise SAP applications to ensure the highest levels of data security. While Chatbots can easily be integrated with cloud-based SAP applications, the challenge, of course, is to integrate them with SAP on-premise applications to cater to such users, who maintain in-house servers, ensuring data confidentiality and security.

This blog decodes this challenge and throws light on how Mobolutions, an SAP partner has accomplished the chatbot integration with SAP on-premise server for the Plant Maintenance (PM) module.

Overview of the Plant Maintenance module

This PM module primarily addresses the requirements of the maintenance personnel working with heavy equipment within a plant. It empowers maintenance technicians and quality inspectors to work safely and productively relating to installing new equipment or maintaining, inspecting and repairing existing assets. Apart from that, this module also mitigates the risk of injury by helping workers complete safety checks and follow safe work practices.

Using this module, a quality inspector can notify their field personnel and technicians about an asset that needs maintenance/repair or even update an existing work order to let their field personnel accomplish a specific maintenance task of an equipment/machinery in their factory premises. On the other hand, field technicians can constantly keep track and update the status of their notifications/work orders or even fill up timesheets to keep track of the time spent on their respective activities.

Apart from creating a notification for specific equipment that needs repair, notifications can also be created for a specific functional location that involves the maintenance of multiple types of equipment.

Why such integration is required?

In a conventional mode, tracking each notification on a specific date or on a functional location is quite a cumbersome process that requires you to manually access the PM module, enter the required parameter and retrieve the notification details. Instead, a chatbot can make this process simple by allowing you to enter simple commands and receive responses instantly in the natural language you are most comfortable with. The results are reliable and accurate, unlike manual intervention that is prone to errors and data mismatches.

There are three SAP notification use cases, which have been integrated into the Chatbot application:

1. Notification created on a specific date
2. Notification created for a specific functional location
3. Notification, based on priority status.

How is it accomplished?

The following is the method to integrate the on-premise Odata with Chatbot that can take the user experience to the next level. Let's illustrate this integration with a simple example:
Let's query the data from an on-premise server and expose the data into a URL using Ngrok. For this, Flask-Ngrok is used to expose the data from Odata queries in a Python environment.

But before we start the process, let's take note of some of the key pointers:

1. The on-premise's Odata is local and hence cannot be accessed outside the VPN environment.

2. To make the on-premise data accessible to Chatbot webhooks, you need to expose this Odata as Ngrok URL. Since Ngrok is an encrypted channel, it can be used for secure communication.

3. This Odata cannot be accessed directly in Chatbot's webhooks since the JSON type required in Webhook is different from the normal Odata's output's JSON response. For this, you need to convert the JSON format of the Webhook URL's response in the Python backend.

Steps for querying the data

Fetch Odata

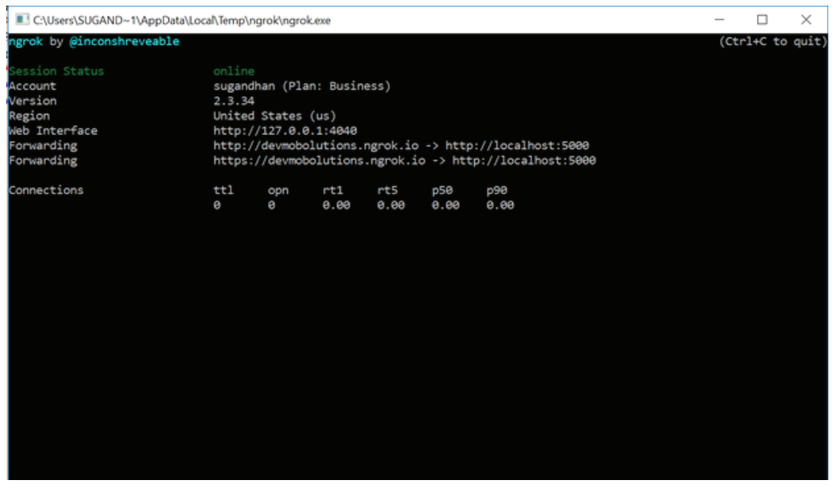
Note: You can access Odata inside the Python environment using the Odata client libraries. The link https://www.odata.org/libraries/ shows the list of all Odata libraries available in Python. Here, we used OdataPy to get the data inside the Python environment.

Integrate and expose the Odata as Ngrok URL.

To do so, follow the below sub-steps:

a. To create a URL, run Flask-Ngrok on your local computer.
b. To get custom domain names, buy a subscription from Ngrok.
c. Expose your data using Flask-Ngrok to get a static URL in the domain name, you have configured.
d. Make changes in the Flask-Ngrok file in the site-packages directory of python.

Note: Ngrok is a tool for integrating any data into webhooks.



Train Chatbot on Plant Maintenance tasks:

We have created the Chatbot training for plant maintenance tasks like

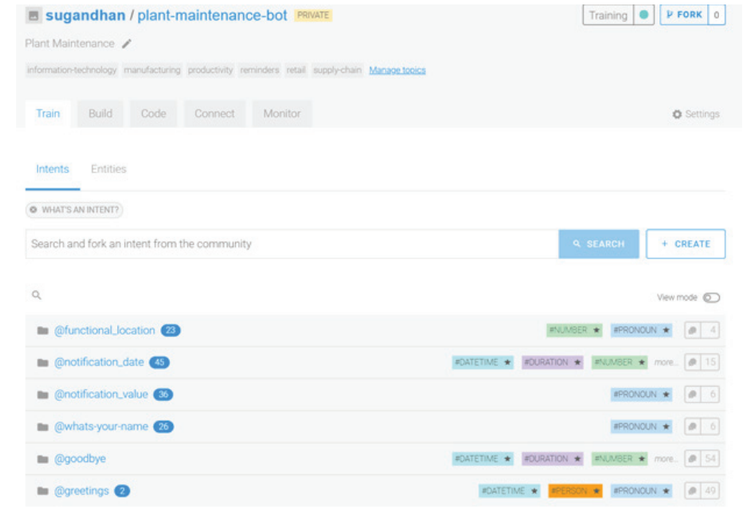
1. Querying high priority notifications
2. Querying notifications created on a specific date
3. Querying notifications created on a specific functional location.

How notifications can be retrieved on a specific date (a typical example)

1. Create Plant Maintenance bot with intents.

Note: An intent refers to a box of sentences that connote similar meanings, though they can be quite divergent from one another. In this case, intents refer to notification_date, functional location, and priority.

When a user types a text to your bot, our algorithm relates it to various phrases, available in the intents. Then it verifies if it's matching closely with one of them and decodes the purpose of the message

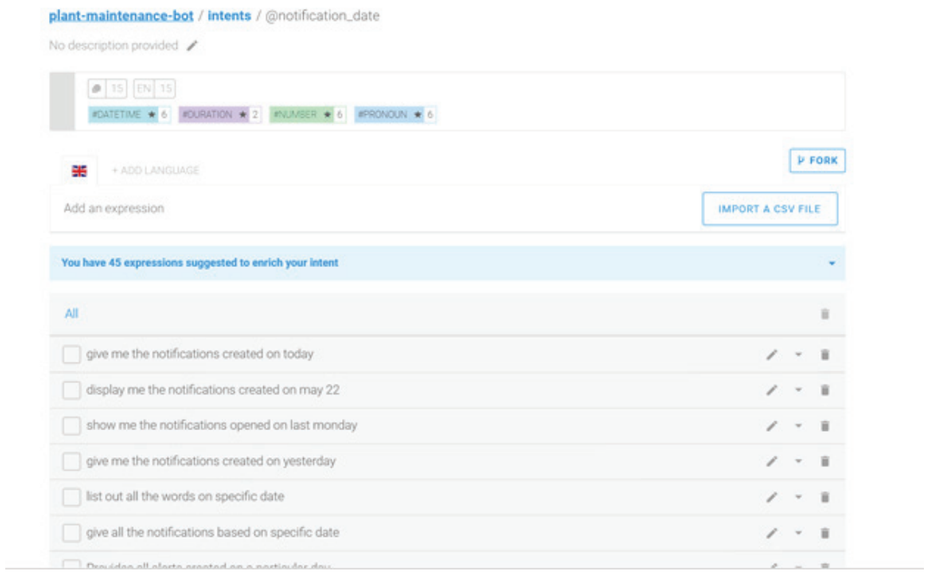


2. Add expressions to the intent: notification_date and train the bot.

Once you have created the intents, you need to populate them with the right expressions. An expression, in this case, refers to the name of a sentence that is appended to an intent. To get the best possible results, it is always preferable to include a minimum of 30 expressions or a maximum of 50 expressions to an intent.

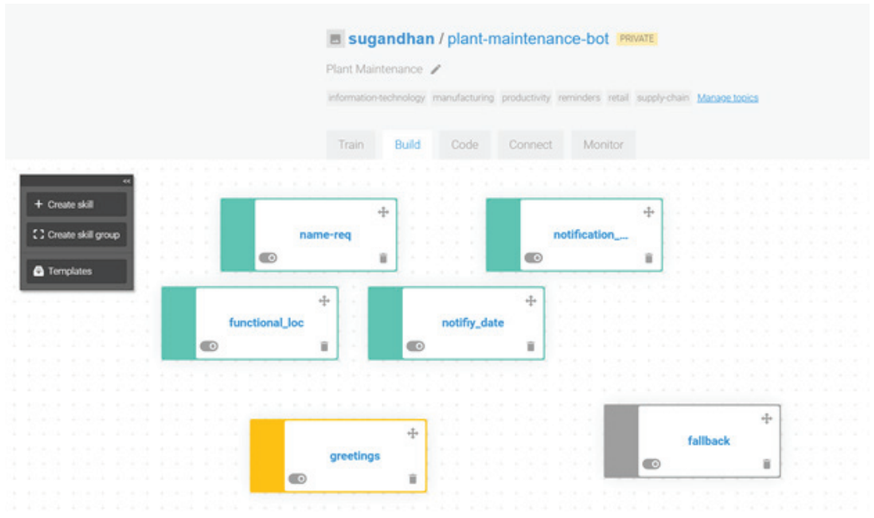
Additionally, add skills to help construct the conversation flow of your bot. A skill typically consists of 4 components: Readme, Triggers, Requirements, and Actions.

In this context, you need to focus only on Triggers and Requirements (Refer steps 3 and 4).



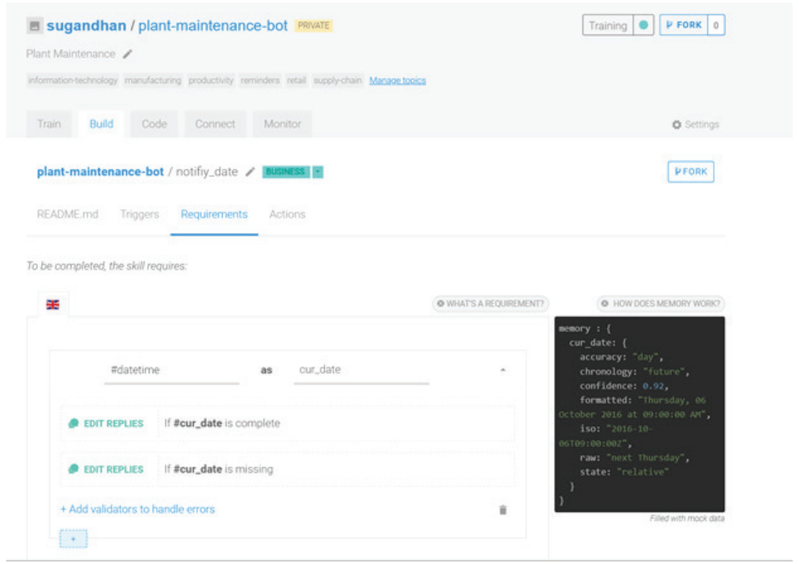
3. Create requirements and add triggers.

When a user types a text to your bot, our algorithm relates it to various phrases, available in the intents. Then it verifies if it's matching closely with one of them and decodes the purpose of the message



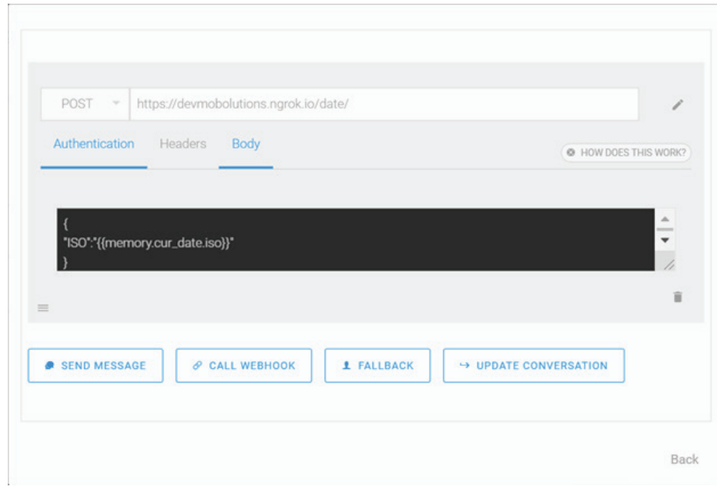
4. Add the NLP information: date_time and set the memory variable as cur_date.

The NLP analyses the information about the date and time from the users' textual information. Once, you have this information, you can trigger a webhook.



5. Add the webhook URL created from Ngrok and add body content to post.

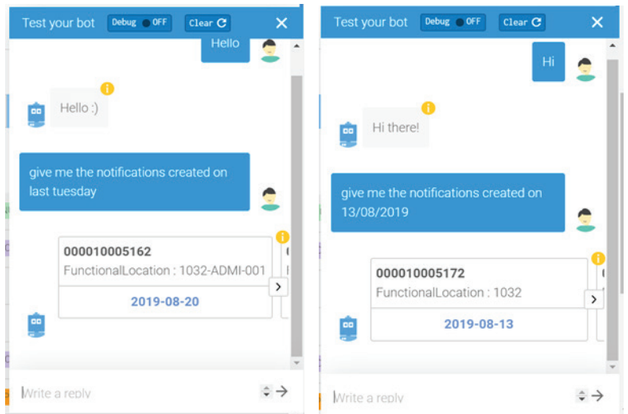
In the webhook of the Requirements page, enter the ngrok URL and the memory variable of the date-time as the header to post the data.



Practical scenarios of this Chatbot application

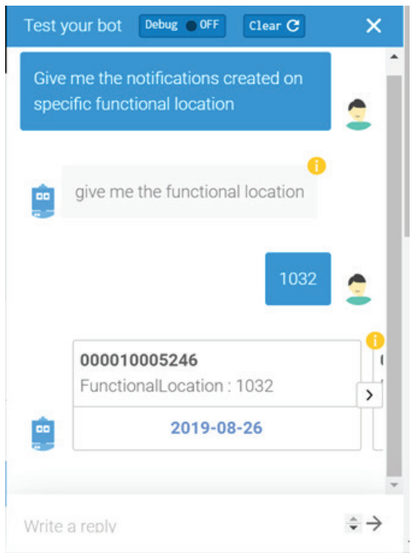
1. Query notifications created on a specific date

Type a question asking the chatbot to display notifications that have been created on a specific day, say last Tuesday (refer below screenshot).
The chatbot then queries and generates all the matching results – in this case, all the notification records that have been created on Tuesday will be displayed.



2. Query notifications created on a specific functional location

Type a question asking the chatbot to display notifications that have been created on a specific functional location (say, 1032). The chatbot processes the request and displays a specific notification matching the intent i.e., functional location – 1032.



3. Query high priority notifications

Type a question asking the chatbot to display high priority (rank 1) notifications. The chatbot processes the request and displays high priority notifications that have been marked with the highest ranking

Key benefits

- Eliminate the necessity of logging into the system for tracking notifications
- Query notifications by simple commands in plain English
- Get quick and accurate responses within seconds
- Simple setup with easy to use interface

From the above discussion, by exposing the Odata to Ngrok URL, you can make the on-premise data accessible to Chatbot webhooks, which is a great step forward for a chatbot – SAP On-premise integration, saving enormous time, resources and above all taking SAP user experience to the next level.