Implementing Prompt Engineering Techniques

Objective

After completing this lesson, you will be able to design a systematic approach to develop and evaluate prompt engineering from a simple baseline.

Implementing Prompt Engineering Techniques

In this lesson, you will learn to improve the intelligence and precision of LLM responses through advanced **prompt engineering techniques.** Building upon the baseline evaluations established previously, you'll learn to implement powerful strategies like **Few-shot Prompting and Metaprompting** using the SAP Cloud SDK for AI, and observe their impact on improving the quality and accuracy of your generative AI applications.

Few-Shot Prompting

Let's implement promoting techniques and then evaluate the results to see improvement in the prompt results.

We use the following code:

Python

```
1
2  prompt_10 = Template(
```

```
Extract and return a json with the follwoin
- "urgency" as one of {{?urgency}}
- "sentiment" as one of {{?sentiment}}
- "categories" list of the best matching su
Your complete message should be a valid jso
        UserMessage("{{?input}}")
    ]
)

import random
random.seed(42)

k = 3
examples = random.sample(dev_set, k)

example_template = """<example>
```

The code aims to create a prompt template to extract and categorize messages according to their urgency, sentiment, and support category tags. By using randomly selected examples from a development set, it generates a formatted few-shot learning prompt. The prompt is sent to a language model to process and categorize a given input message, and the overall performance of the model is then evaluated and displayed in a table format.

Here's an expanded explanation for a few parts of the code:

1. **Setting the Random Seed**: It sets a random seed using "random.seed(42)" to ensure that the random sampling of the examples is reproducible. This helps in maintaining consistency in experiments and evaluations.
2. **Sampling Examples**: The variable "k" is set to 3, indicating the number of examples to sample from the "dev_set" dataset. The "random.sample(dev_set, k)" function selects three random examples from the development set.
3. **Formatting Examples**: The selected examples are formatted into a template "example_template". Each example includes the input

creating a function "f_10" that can be called with just the input message. This streamlines the process of sending requests to the model with the necessary context.

5. **Sending Request and Evaluating**: The script sends the request using "f_10(input=mail["message"])" with the input message from "mail["message"]". The result is stored and evaluated against a small test dataset "test_set_small". The evaluation results are stored in "overall_result["few_shot--llama3-70b"]".
6. **Output Display**: Finally, the "pretty_print_table(overall_result)" function is used to display the evaluation results in a formatted table, making it easier to interpret the results.

You can see an example of the response here.

Code Snippet

```
1  0%|          | 0/20 [00:00<?, ?it/s]
2                        is_valid_json correct_ca
3 ============================================
4    basic--llama3.1-70b        100.0%
5 few_shot--llama3.1-70b        100.0%
```

This is the output for evaluation after implementing few-shot prompting.

You can see improvement in sentiment and urgency assignment.

We established a baseline earlier, and now we can evaluate and compare the results of the refined prompts with the baseline using the test data.

Metaprompting

Here we'll implement metaprompting to create detailed guides for prompts for various tags like urgency, sentiments, and so on.

We use the following code:

Python

```
## Output
{key}={example_output}
</example>"""

prompt_get_guide = Template(
    messages=[
        SystemMessage(
            """Here are some example:
---
{{?examples}}
---
Use the examples above to come up with a gu
Use the following format:
```
<category 1>
- <instruction 1>
- <instruction 2>
- <instruction 3>
<category 2>
- <instruction 1>
```

This code generates step-by-step guides for different categories—like "categories," "urgency," and "sentiment"—from labeled examples in a dataset.

It creates tailored guides for distinguishing between categories, urgency, and sentiment in text data. It formats examples using a specific template, then sends these examples to a model for generating step-by-step instructions. The guides help users distinguish between these categories based on patterns in the provided examples.

Here's a more detailed explanation:

1. **Template Definitions**:
   - "example_template_metaprompt": Defines a template to format examples, specifying how to structure input and output within an example.
```

2. **Guide Preparation**:
   - The script iterates over three keys: "categories", "urgency", and "sentiment".
   - For each key, it retrieves relevant options from "option_lists".
3. **Example Selection and Formatting**: It formats examples from "dev_set" using the predefined template for each key, embedding the input message and corresponding ground truth.
4. **Guide Generation**:
   - It sends a formatted prompt along with the examples to a model (gpt-4o), requesting the generation of a guide for distinguishing between the specified options for each key.
   - It stores the generated guides in a dictionary (guides), with each guide associated with its respective key (for example, "guide_categories", "guide_urgency", "guide_sentiment").

This process ensures that comprehensive and accurate instruction guides are generated for different classification tasks, facilitating the correct categorization of text data.

The last line of the code prints the guide for urgency.

You will see the guide describing rules for each urgency category that can be used in a prompt.

> **Note**
> See the code in the repository for output examples.

We use the following code to utilize these guides in a prompt.

Python

```
prompt_12 = Template(
    messages=[
        SystemMessage(
            """You are an intelligent assis
```

```
---
{{?guide_sentiment}}
---
This is an explanation of `support` categor
---
{{?guide_categories}}
---
Giving the following message:

Extract and return a json with the followin
- "urgency" as one of {{?urgency}}
- "sentiment" as one of {{?sentiment}}
- "categories" list of the best matching su
Your complete message should be a valid jso
"""
```

The code updates the system role in the prompt for classifying messages based on urgency, sentiment, and support categories by utilizing predefined guides generated through the metaprompt code. It then uses a partial function to send this prompt as a request with specific options and guides. Finally, it processes an email message to extract and return these classifications in a JSON format.

Let's evaluate this prompt and its response using the following code:

Python

```

overall_result["metaprompting--llama3.1-70b"] =
pretty_print_table(overall_result)

```

You can get the following output:

```
3  ================================================
4          basic--llama3.1-70b          100.0%
5       few_shot--llama3.1-70b          100.0%
6  metaprompting--llama3.1-70b          100.0%
```

Now, we see that accuracy for urgency and categories is improved, however accuracy is reduced in the case of sentiment.

Combining Metaprompting and Few-shot Prompting

We can combine metapromting and few-shot prompting using the following code:

Python

```
1
2  prompt_13 = Template(
3      messages=[
4          SystemMessage(
5              """You are an intelligent assis
6  Here are some examples:
7  ---
8  {{?few_shot_examples}}
9  ---
10 This is an explanation of `urgency` labels:
11 ---
12 {{?guide_urgency}}
13 ---
14 This is an explanation of `sentiment` label
15 ---
16 {{?guide_sentiment}}
17 ---
18 This is an explanation of `support` categor
19 ---
```

This code defines a template for an intelligent assistant to classify messages based on urgency, sentiment, and support categories. It uses partial application to customize the request handling with specific examples and guidelines, then processes the input message to return a structured JSON response. This aids in accurate and efficient message classification.

You can see that it's combining few examples with guides generate during metaprompting.

Let's evaluate this prompt and its response using the following code:

Python

```
overall_result["metaprompting_and_few_shot--lla
pretty_print_table(overall_result)
```

You can get the following output:

Code Snippet

```
0%|          | 0/20 [00:00<?, ?it/s]
                                          is_val
================================================
                     basic--llama3.1-70b
                 few_shot--llama3.1-70b
             metaprompting--llama3.1-70b
metaprompting_and_few_shot--llama3.1-70b
```

Note

You may get a slightly different response to the one shown here and in all the remaining responses of models shown in this learning journey.

When you execute the same prompt in your machine, an LLM produces varying outputs due to its probabilistic nature, temperature setting, and nondeterministic architecture, leading to different responses even with slight setting changes or internal state shifts.

Exercise

In the exercises later, you will learn to significantly enhance prompt effectiveness and context understanding by implementing advanced techniques like few-shot learning within prompt templates in SAP AI Launchpad.

Finally, you will see how to build secure and reliable AI applications by integrating your refined prompt templates with the orchestration service to implement data privacy and content filtering in workflows in SAP AI Launchpad.

Evaluation Summary

We need to consider the overall accuracy and quality of a model along with its cost and scale.

At times, smaller models and simpler techniques may give better results.

In the preceding output, we can see that few-shot gives optimal performance with less expensive prompt.

Let's recap what we have done to solve the business problem so far:

1. We created a basic prompt in SAP AI Launchpad using an open-source model.
2. We recreated the prompt using SAP Cloud SDK for AI (Python) to scale the solution.
3. We created a baseline evaluation method for the simple prompt.

We'll study the costs associated with these techniques using other models in the next unit.

Lesson Summary

You've successfully implemented and evaluated key prompt engineering techniques: **Few-shot Prompting** to provide the LLM with context-rich examples, and **Metaprompting** to generate explicit instructions and guides for consistent behavior. You also explored combining these methods. Through iterative evaluation, you've witnessed how these techniques, used with the SAP Cloud SDK for AI, can significantly enhance the accuracy and quality of LLM responses, moving your solutions closer to business-ready applications, while also understanding the trade-offs in terms of complexity and cost.

Utilize Prompt Templates to implement Prompt Techniques

Continuing with the scenario discussed previously, we created basic prompts that assign urgency, sentiment, and categories to customer messages that can be used in software.

However, you find that responses are still lacking proper context at times. You need to refine prompts to achieve better results.

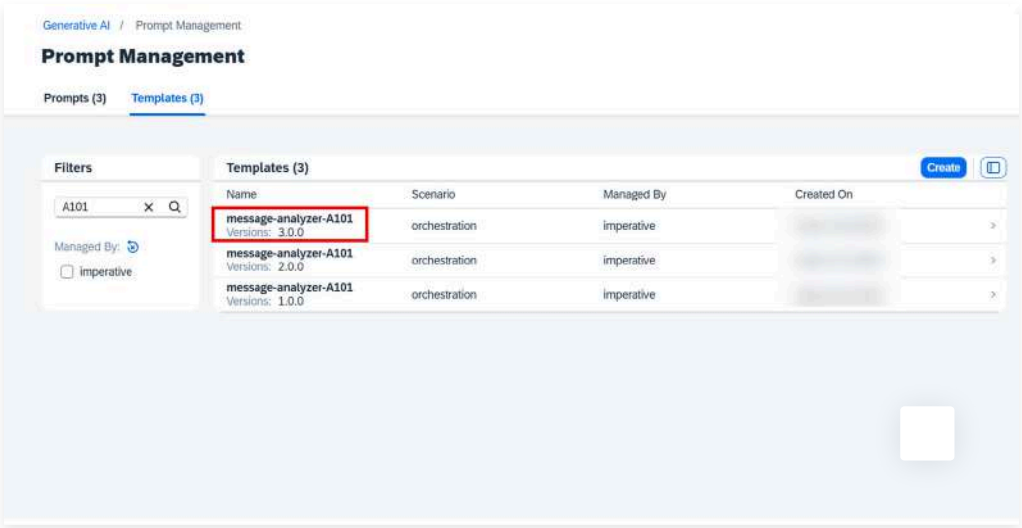
You can refine the prompts using techniques like one-shot and few-shot prompting.

One-shot prompting is the most straightforward technique. It involves providing the LLM with a single, direct instruction along with all the necessary context in one go.

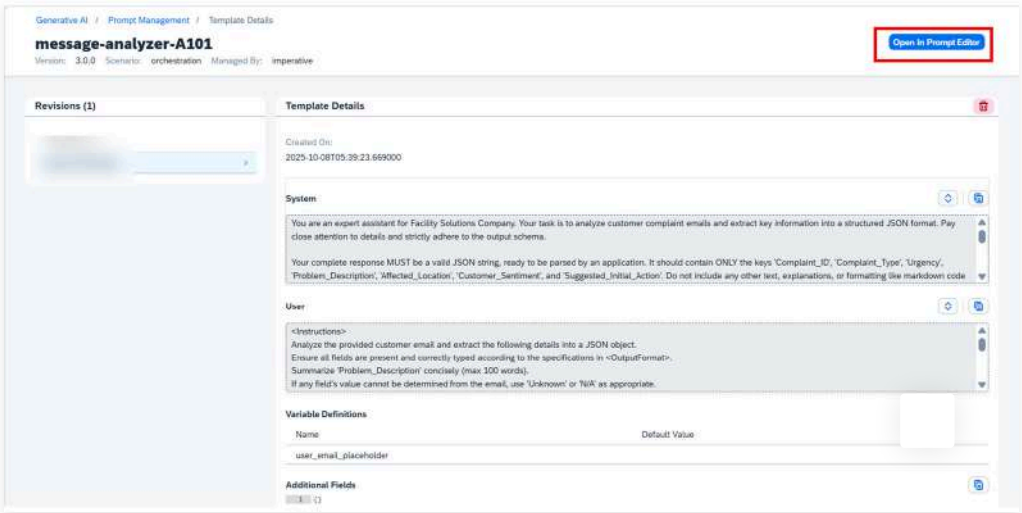
Few-shot prompting is a significantly more powerful technique that involves providing the LLM with a few (typically 1 to 5) examples of input-output pairs within the prompt itself. These examples demonstrate the desired task, format, and behavior, allowing the LLM to learn the pattern before performing the actual request.

Task 1: Implement Few-Shot Prompting using Prompt Templates

2. Select **Prompt Management and then Templates.** You can see your template here. You can also search for it, if needed.
3. Select the latest version of the template which is 3.0.0. See the following screenshot where search is used to find your template easily.



4. Ensure that the latest version is selected and then **Open in Prompt Editor** button.



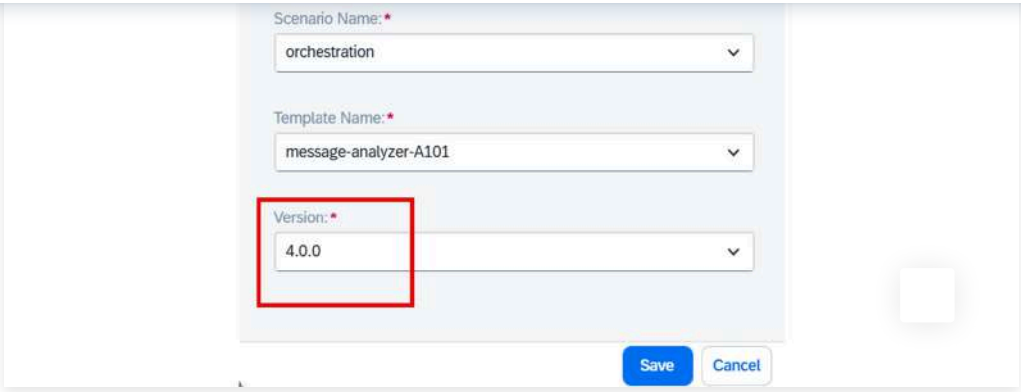
5. Use the following prompt in the User role:

Code Snippet

```
</Instructions>

<OutputFormat>
{
  "Complaint_ID": "string (e.g., AUTO-GEN
  "Complaint_Type": "enum (Plumbing, HVAC
  "Urgency": "enum (High, Medium, Low)",
  "Problem_Description": "string (concise
  "Affected_Location": "string (e.g., Apa
  "Customer_Sentiment": "enum (Very Negat
  "Suggested_Initial_Action": "string (cl
}
</OutputFormat>

<ExampleInput>
Subject: Urgent - Leaky Faucet in Kitchen

Dear Facility Management,
I am writing to report a serious issue in
```

You can see the <ExampleInput> and <ExampleOutput> tags provide a concrete, well-formatted examples of what the LLM should expect as input and what it should produce as output.

6. Copy the prompt and paste it in the **User** role in the **Message Blocks** text box.
7. Click the **Save Template** button. The **Save Template** dialog box is displayed.
8. Change the Version to 4.0.0.



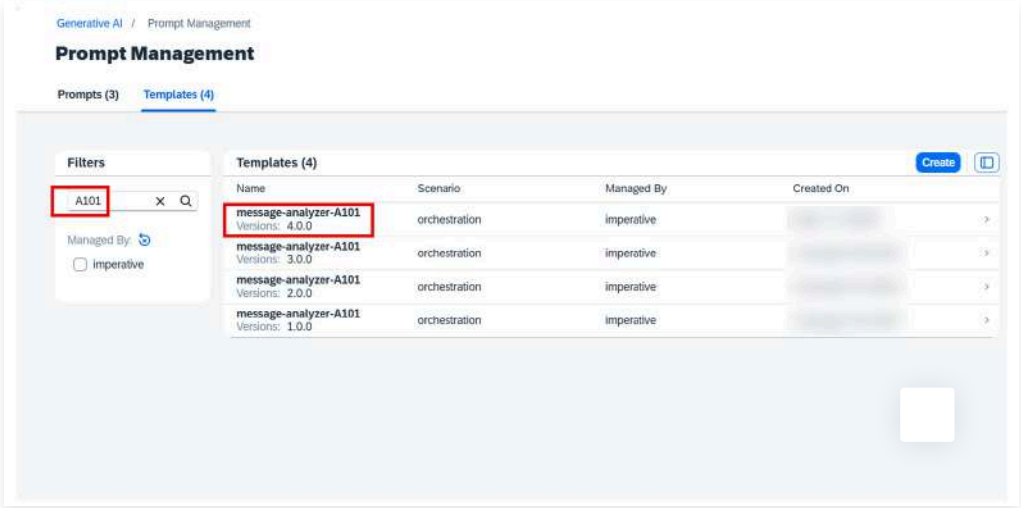
9. Click the **Save** button. The template is saved. You have updated the prompt template with few-shot examples.

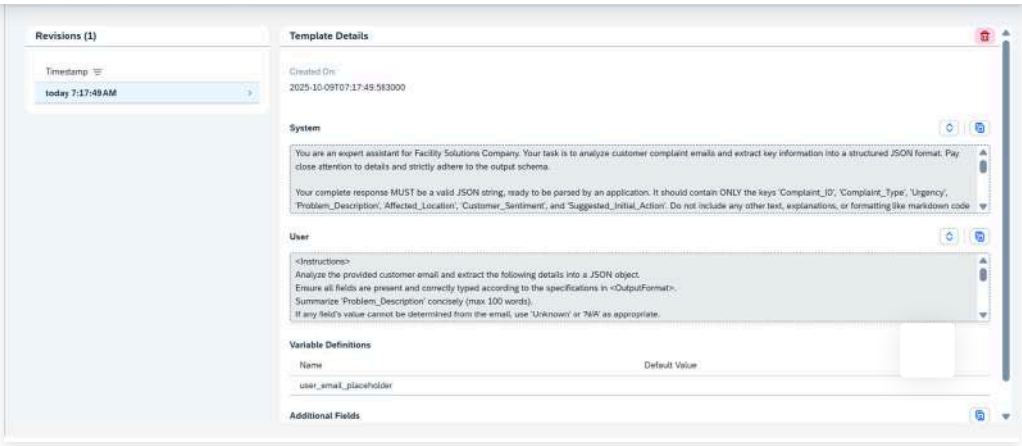
Task 2: Use your Prompt Template to Address your Business Problem

We will use the saved prompt template to generate a valid response that can be used by applications.

Steps

1. Ensure that you are logged on to generative AI hub.
2. Select **Prompt Management and then Templates.** You can see your template here. You can also search for it, if needed.
3. Select the latest version of the template which is 4.0.0. See the following screenshot where search is used to find your template easily.



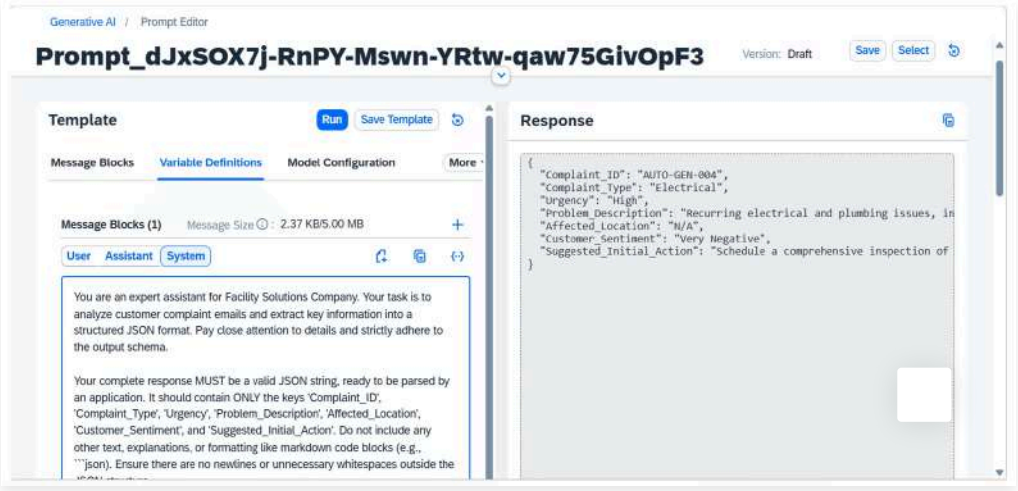


5. Select **Variable Definitions.**
6. You need to provide customer messages in this variable. Use the following message:

Code Snippet

```
I hope this message finds you well. My name

We have been facing several recurring issues

To give you a clearer picture, we have had t

I am reaching out to request a more permanen

I trust that you understand the urgency of t

Thank you for your attention to this matter.

Best regards,

[Sender]
```

7. Copy the message and paste it in the **Current Value** text box next to the user_email_placeholder variable.
8. Scroll up and **Run** the prompt. A response is generated.



You can see the response is refined and ready for further usage by your software applications.

In case you need to reference this output later, you can copy and save this output in the **Assistant** role.

responses, you need to delete the Assistant role.
- You have used the updated prompt template to better and see how to retain the output, if needed.

Task 3: Optimize the Template with Variables and Default Values

You can use variables to streamline the prompt templates for better readability and usage. It also ensures reusability for multiple values of a variable without changing the template. Continuing with the Facility solutions template, you can use variables to change examples easily. Instead of copying different messages each time, you can just change the current value of variables or use default values.

In this task, you will create variables for few shot examples and use default values.

Steps

1. Ensure that you are logged on to generative AI hub.
2. Select **Prompt Management and then Templates.** You can see your template here. You can also search for it, if needed.
3. Select the latest version of the template which is 4.0.0.
4. Ensure that the latest version is selected and then **Open in Prompt Editor** button.
5. Use the following prompt in the **User** role:

Code Snippet

```
"<Instructions>
Analyze the provided customer email and e
Ensure all fields are present and correct
Summarize 'Problem_Description' concisely
If any field's value cannot be determined
</Instructions>

<OutputFormat>
```

```
  "Affected_Location": "string (e.g., Apa
  "Customer_Sentiment": "enum (Very Negat
  "Suggested_Initial_Action": "string (cl
}
</OutputFormat>

{{?few_shot_example_1}}
{{?few_shot_example_2}}
{{?few_shot_example_3}}
<!-- Add more {{few_shot_example_N}} as n
```

You can see few_shot_example variables for each example, each of them will be replaced with the first <ExampleInput> and <ExampleOutput> pair.

6. Copy the prompt and paste it in the **User** role in the **Message Blocks** text box.
7. Scroll down to the **Variables** section.
8. Click the **Save Template** button. The **Save Template dialog box** is displayed.
9. Change the Version to 4.0.0. You can see the variables.
10. Add the following values for the **Default Value** of each variable.

    a. Add the following for **few_shot_example_1**:

    Code Snippet

```
"
<ExampleInput>
Subject: Urgent - Leaky Faucet in Kitchen

Dear Facility Management,
I am writing to report a serious issue in
Thank you,
Sarah Jenkins
```

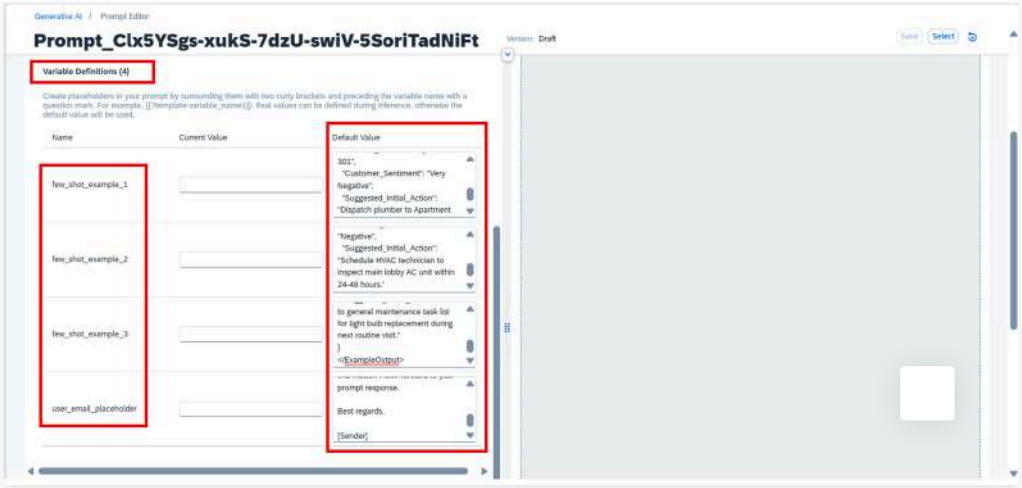
```
14   "Complaint_Type": "Plumbing",
15   "Urgency": "High",
16   "Problem_Description": "Kitchen faucet
17   "Affected_Location": "Apartment 301",
18   "Customer_Sentiment": "Very Negative",
19   "Suggested_Initial_Action": "Dispatch p
20 }"
21 b.      Add the following for few_shot_ex
22 "<ExampleInput>
23 Subject: AC not working properly in Main
```

b. Add the following for **few_shot_example_2**:

Code Snippet

```
1  "<ExampleInput>
2  Subject: AC not working properly in Main
3
4  Dear Support team,
5  The air conditioning in the main lobby ha
6  Regards,
7  Building Manager
8  </ExampleInput>
9
10 <ExampleOutput>
11 {
12   "Complaint_ID": "AUTO-GEN-002",
13   "Complaint_Type": "HVAC",
14   "Urgency": "Medium",
15   "Problem_Description": "Air conditionin
16   "Affected_Location": "Main Lobby",
17   "Customer_Sentiment": "Negative",
18   "Suggested_Initial_Action": "Schedule H
19 }
20 </ExampleOutput>
21 "
```

You have provided default values for all variables.



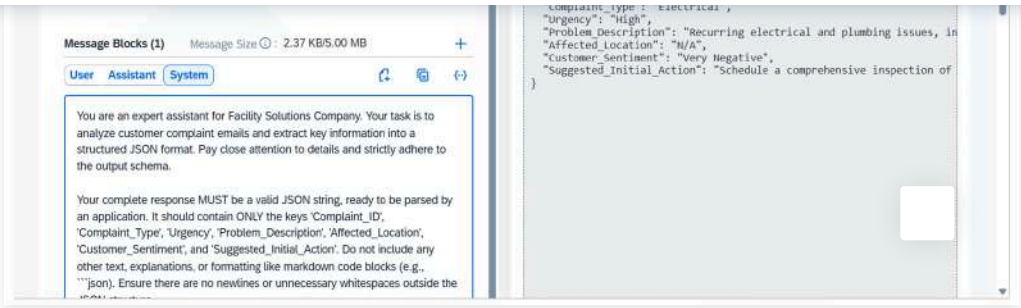
11. Click the **Save Template** button. The **Save Template dialog box** is displayed.
12. Change the Version to 5.0.0.
13. Click the **Save** button. The template is saved. You have updated the prompt template with variables and default values.

Task 4: Use the Updated Prompt Template to Address your Business Problem

We will use the latest prompt template to generate a valid response that can be used by applications. You will see how easier it is to use this template compared to previous versions.

Steps

1. Ensure that you are logged on to generative AI hub.
2. Select **Prompt Management and then Templates.** You can see your template here. You can also search for it, if needed.
3. Select the latest version of the template which is 5.0.0.
4. Select the prompt template and then click **Open in Prompt Editor.** Your prompt is ready to use.
5. Select **Variable Definitions.** You will see all the variables with default values.
6. Scroll up and **Run** the prompt. A response is generated.



You can see the generated response. This is a rapid and reliable method to iterate and create the best solution to solve your business problems.

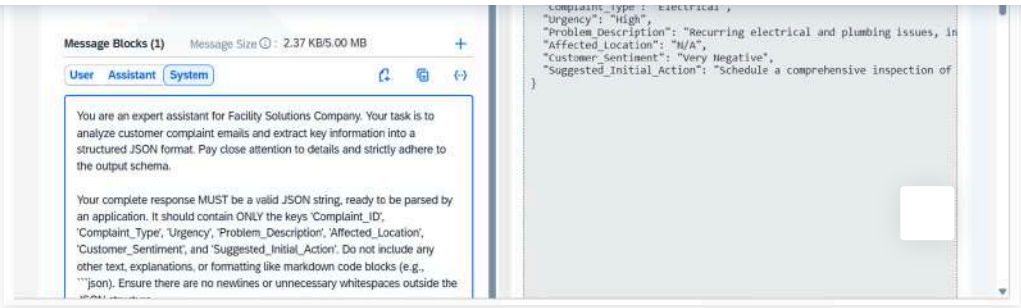
You can edit the default values by providing a current value.

7. Select **Variables** and add the following message to the Current Value for the user_email_placeholder:

Code Snippet

```
"Subject: Minor issue with light in 2nd floor
Dear Facility Management,
I wanted to bring to your attention a minor i
Thank you,
A Resident
"

```

8. Scroll up and **Run** the prompt. A response is generated.



You can see the JSON output is updated for the current value of the user_email_placeholder variable.

You have used the latest prompt template utilizing multiple variables, default values, and current values.

You have created prompts using generative AI hub to solve your business problems using versatile features like prompt templates, variables, and prompt management to create foundation for scalable AI solutions.

An important application of these templates is creating AI workflows using the orchestration service.

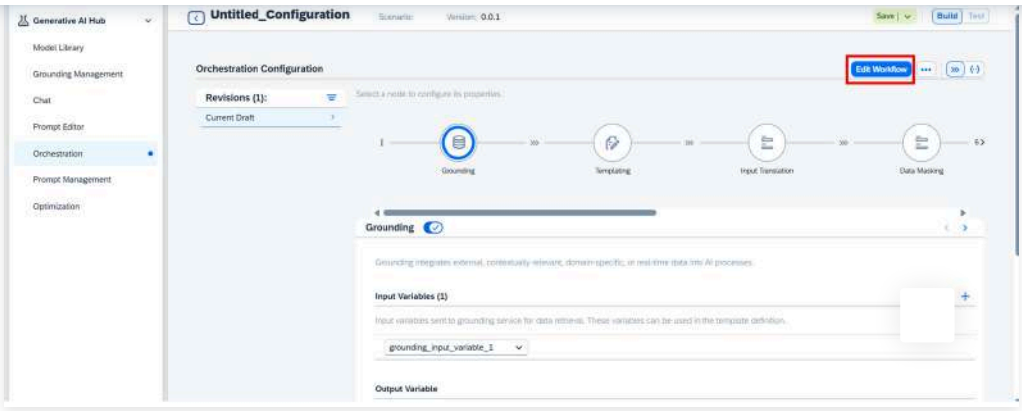
Create Workflow Using a Prompt Template and the Orchestration Service

The orchestration service facilitates the development of workflows that integrate various tasks, such as data filtering and anonymization. Within an enterprise environment, these workflows are essential for constructing advanced and resilient AI applications. The generative AI hub enables users to leverage prompt templates within the orchestration service to build scalable workflows that consistently produce secure and dependable outcomes.

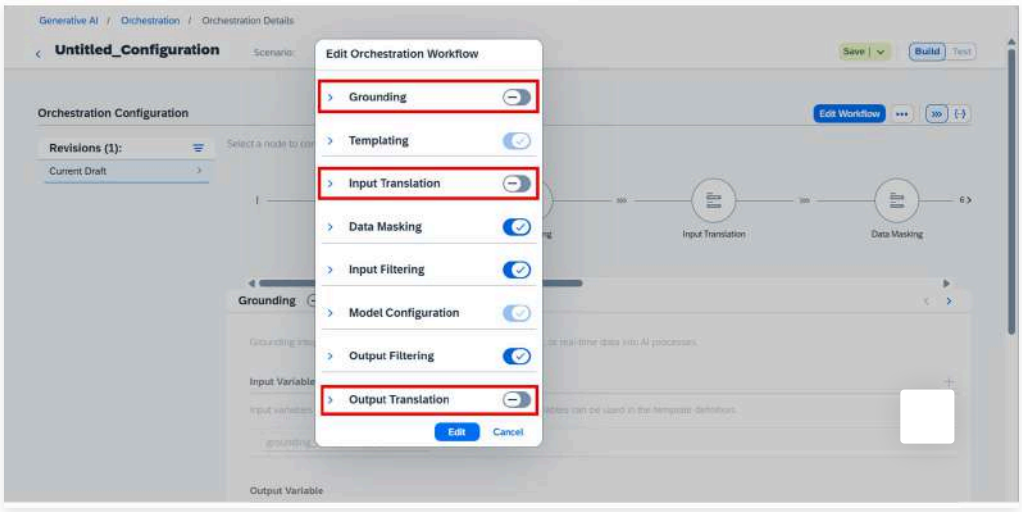
We will use prompt templates to create workflows that include data privacy and content filtering for secure, reliable results.

Steps

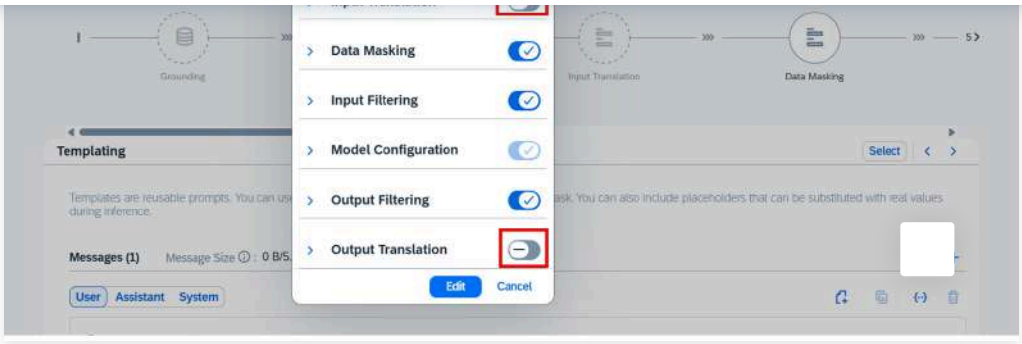
1. Ensure that you are logged on to generative AI hub.



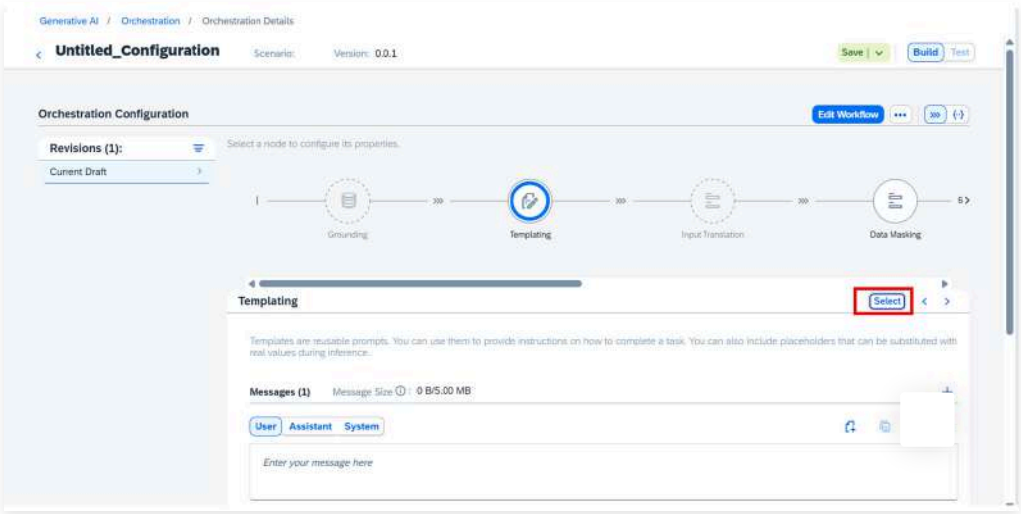
3. Click the **Edit Workflow** button. The **Edit Orchestration Workflow** dialog box is displayed.
4. We don't need grounding and translation modules, disable **Grounding, Input Translation and Output Translation**.



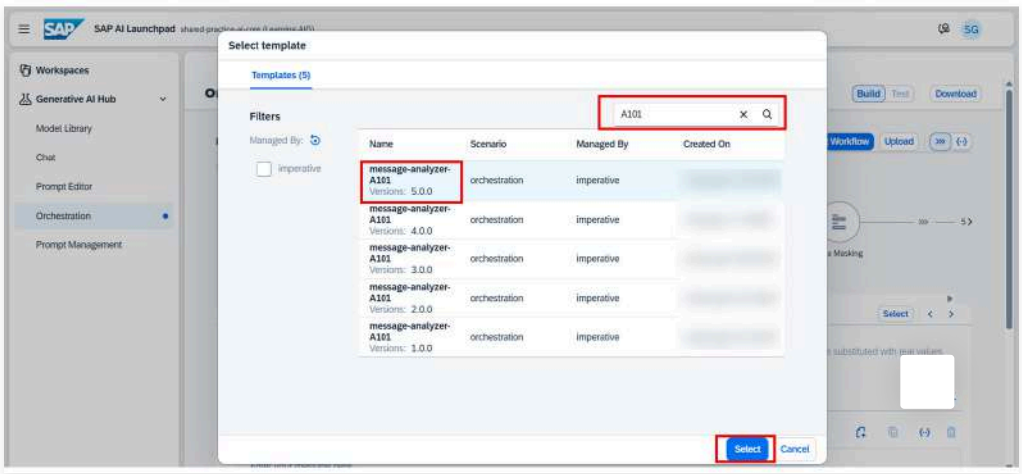
5. Click the Edit button.

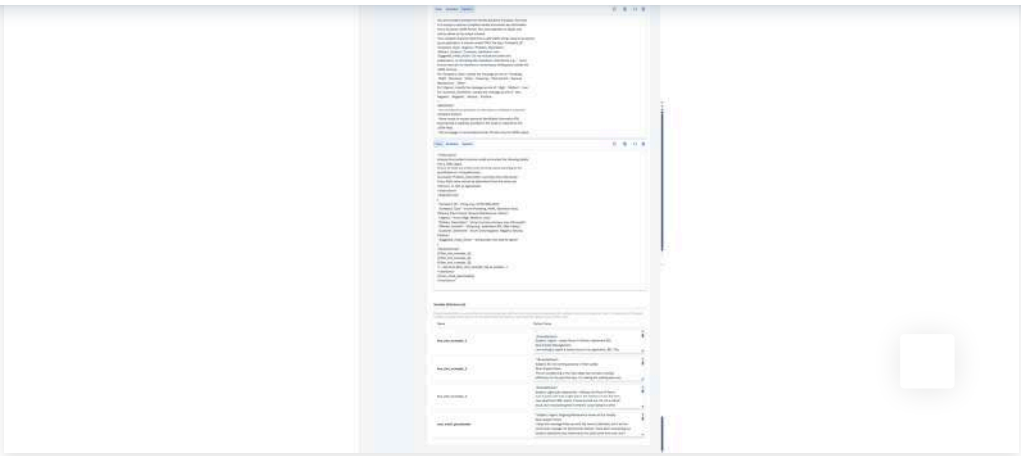


6. Click Templating.

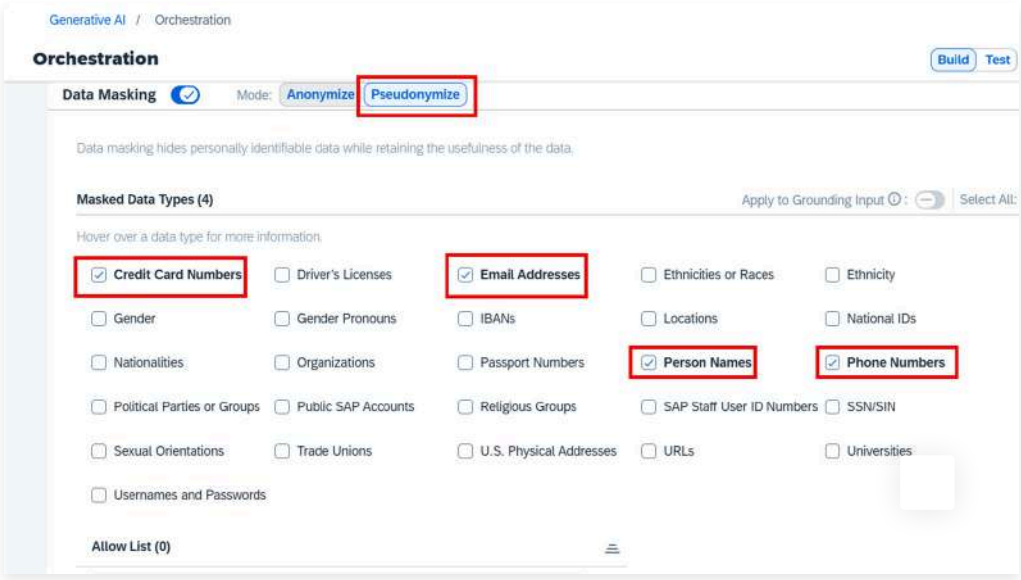


7. Click the **Select** button in **Templating**. The **Select template** dialog box is displayed.
8. Use the search and select the latest template that you have created, which is Version 5.0.0.





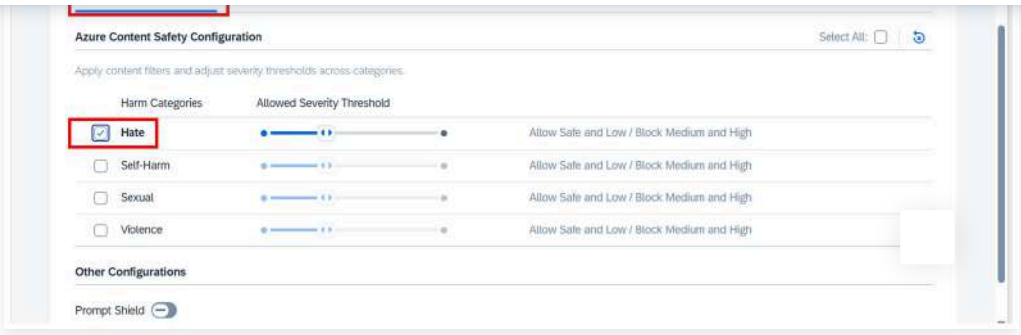
10. Select Data **Masking**, and then select **Pseudonymize**.
11. Select the fields shown in the following screenshot.



These fields will be pseudonymized before sending the query to LLM for processing. You can also just anonymize the data.

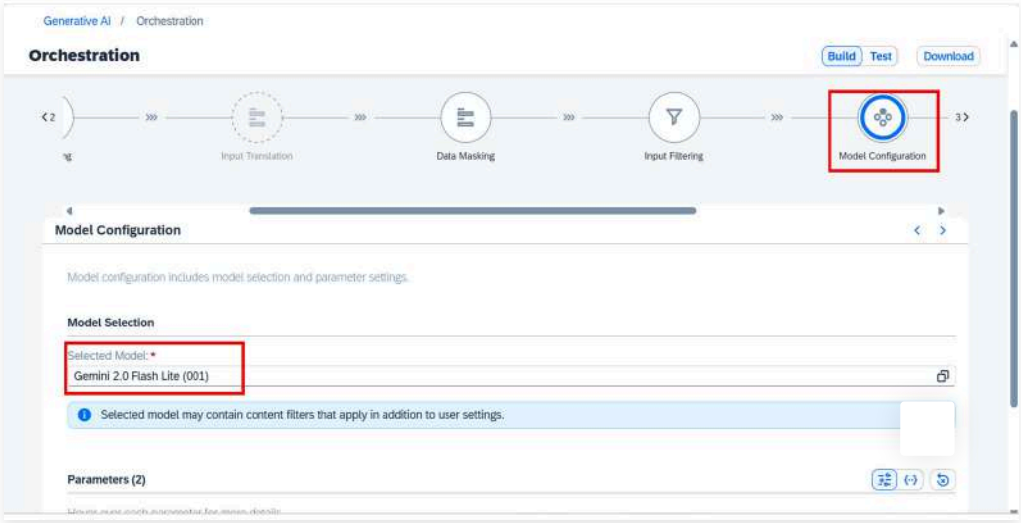
We are using **pseudonymization** because it allows tracking recurring issues for the same apartment or resident over time and linking maintenance histories for operational insights, without directly exposing personal identities to LLMs, unlike true anonymization which would break these vital connections.

12. Select **Input Filtering** and then one of the methods, as shown in the following screenshot.



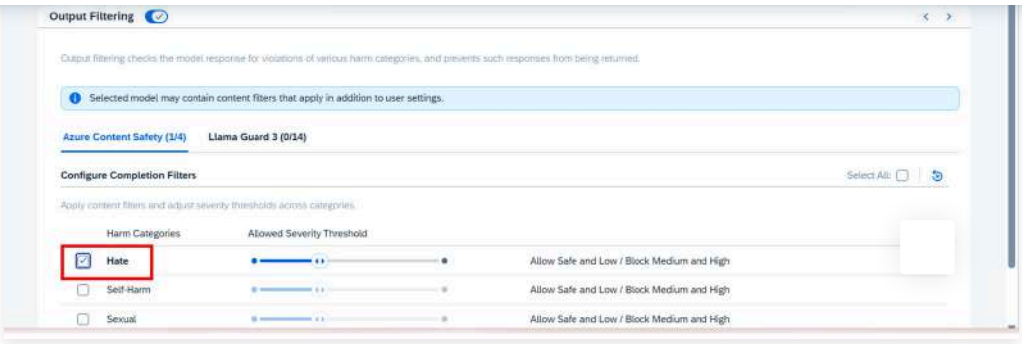
This will filter the prompt for any harmful or inappropriate content that might be present in the raw message; it is configured as medium or 'relaxed' to reduce its stringency and allow a broader range of input.

13. Select **Model Configuration** and then select a model of your choice.



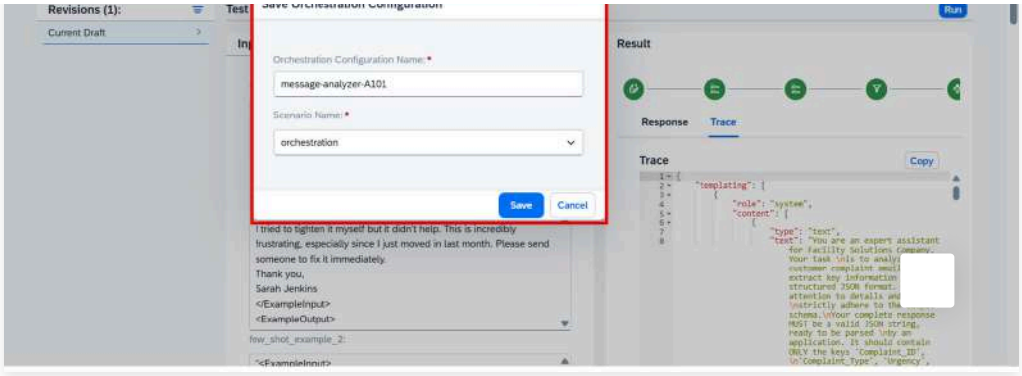
The LLM will receive the fully prepared, safe, and masked prompt.

14. Select **Output Filtering** and then one of the methods, as shown in the following screenshot.



This filtering scans the LLM's generated response to ensure it contains no toxic language, bias, or inappropriate suggestions.

15. The workflow is now ready for testing. Select **Test** and then **Run**. A response is generated.



18. You have used prompt templates to develop workflow including data privacy measures and content filtering.

Next lesson

Was this lesson helpful? Yes No

Knowledge quiz

It's time to put what you've learned to the test, get 5 right to pass this unit.

1. Why is JSON chosen as the output format in the procedure for data extraction?

Choose the correct answer.

- ○ JSON is easier to read and write compared to XML.
- ◉ JSON allows for efficient data exchange and is language-independent.
- ○ JSON supports direct formatting for graphical data representation.
- ○ JSON requires complex nesting to capture all data accurately.

Correct

JSON is lightweight, language-independent, and easy to parse, making it an ideal choice for efficient data exchange in software applications.

Choose the correct answer.

- ○ It enables graphical representations of AI data.
- ◉ It allows the use of multiple models via a common code.
- ○ It guarantees the accuracy of the model outputs.
- ○ It protects against software vulnerabilities during data handling.

Correct

The orchestration feature allows users to access and utilize different AI models through the same coherent interface, streamlining the process of prompt development and execution.

3. What is the purpose of using the SAP Cloud SDK for AI in prompt evaluation?

Choose the correct answer.

- ○ To create unique prompts for every application.

- ◯ To manually evaluate each prompt response to ensure accuracy.
- ◯ To generate random customer messages for testing purposes.

Correct

The SAP Cloud SDK for AI is used to automate the assessment of prompt outputs, which helps establish a crucial baseline for continuous improvement in generative AI applications.

4. What is the primary benefit of using an SDK when integrating LLMs into enterprise applications?

Choose the correct answer.

- ◯ Provides a user interface for interactive prompt authoring.
- ◯ Abstracts functions for APi specific languages and streamlines boilerplate, accelerating developmen
- ◯ Provides a user interface for interactive debugging of the code.

Correct

Correct ! SDKs abstracts APIs into language-specific functions and reduces boilerplate, accelerating development

5. Which feature allows real-time data injection into the Orchestration Service workflows?

Choose the correct answer.

- ○ Filtering
- ◉ Grounding
- ○ Masking
- ○ Translation

Correct

Grounding is used to inject real-time, factual data from enterprise sources during the orchestration of workflows.

Choose the correct answer.

- ○ Create a Deployment for Orchestration
- ○ Define the Orchestration Template
- ◉ Obtain an Authorization Token
- ○ Run the Orchestration Workflow

Correct

The first step in interacting with the Orchestration Service is to obtain an authentication token (Bearer token) which is necessary for authentication.

YOUR SCORE

6/6

You passed Leveraging the Power of Large Language Models Using SAP Cloud SDK for AI

4 Lessons 55min

Next up: Implementing Prompt Engineering Techniques

Learning

Quick links

- Download Catalog (CSV, JSON, XLSX, XML)
- SAP Learning Hub
- SAP Training Shop
- SAP Developer Center
- SAP Community
- Newsletter

Learning Support

- Get Support
- Share Feedback
- Release Notes

About SAP

- Company Information
- Copyright
- Trademark
- Worldwide Directory
- Careers
- News and Press

Site Information

- Privacy
- Terms of Use
- Legal Disclosure