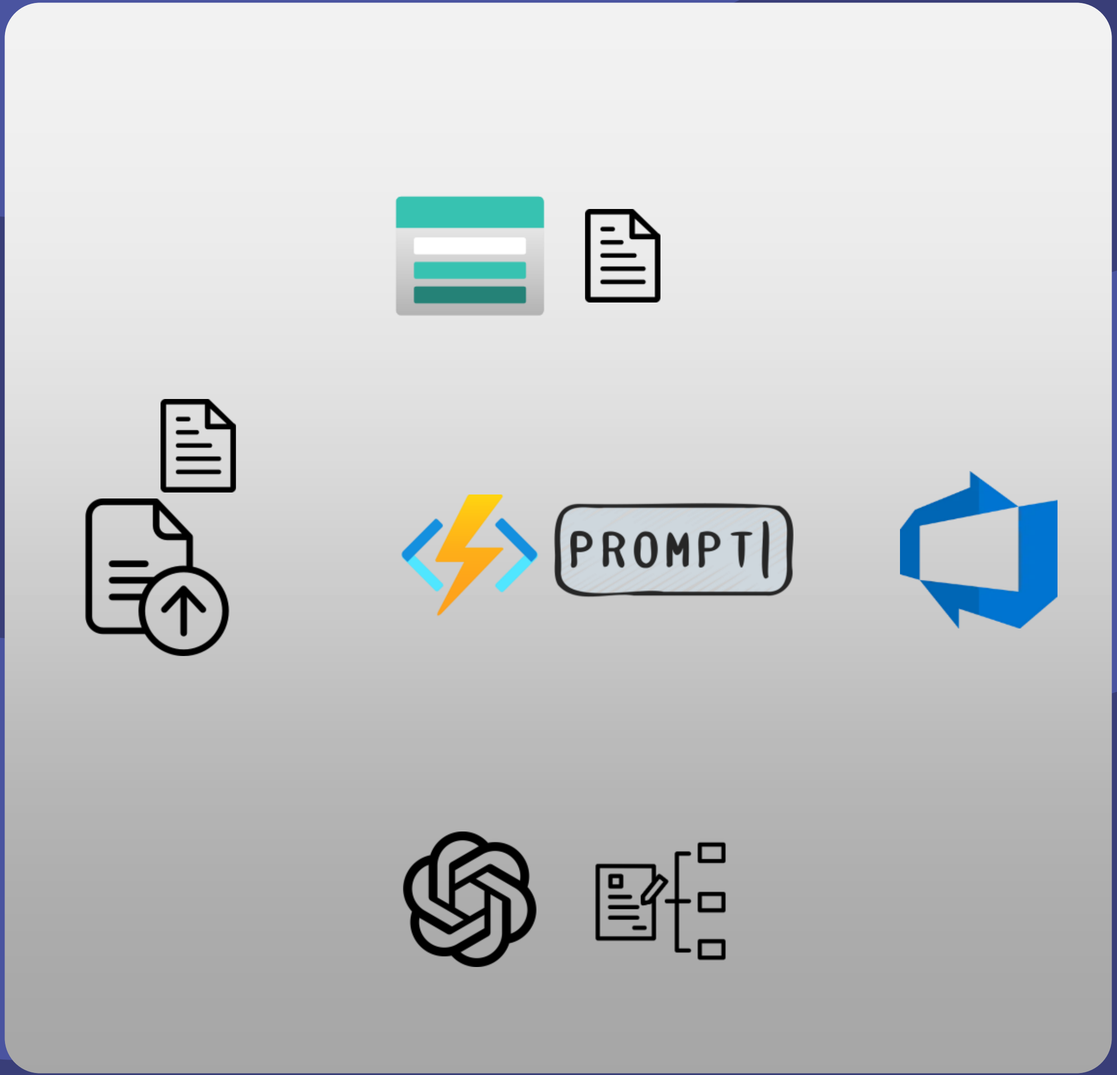
directions vienna
emea 2024

How to be a Prompt Engineer



directions vienna
emea 2024

How to be a Prompt Engineer

How to be a Prompt Engineer

You are a tool that extracts project backlog items from given texts.

If there are any quotes change them to single quotes.

You should read whole text and extract array of backlog items in JSON format from it.

directions vienna
emea 2024

Tine Starič

Software Architect @ Companial

MVP Microsoft® Most Valuable Professional

directions vienna
emea 2024

Agenda

- Build prompts with structure and examples
- Use model capabilities to your advantage
- Test prompts and test them again

Build prompts with structure and examples

You are a tool that extracts project backlog items from given texts.

If there are any quotes change them to single quotes.

You should read whole text and extract array of backlog items in JSON format from it.

Build prompts with structure and examples

Build prompts with structure and examples

All about the words

Instructions/Task Definition

Steps

Format Requirements

Examples

Notes/Remarks

Few-shot learning

Don’t only tell the model what you want, show the model how you want it

directions vienna emea 2024

👤 Add 50 units of Item A and 100 units of Item B to the purchase order.

Please extract the items mentioned in the following text that the user wants to add to the purchase order and return them in JSON format.

Few-shot learning

Don’t only tell the model what you want, show the model how you want it

Add 50 units of Item A and 100 units of Item B to the purchase order.

Extract the items a user specifies they want to add to a purchase order in the ERP and return them in JSON format. Here are some examples:

Example 1:
Text: 'Please add 30 units of Item X, 10 units of Item Y, and 5 units of Item Z to the purchase order.'
Output:

{ "items": [{ "name": "Item X", "quantity": 30 }, { "name": "Item Y", "quantity": 10 }, { "name": "Item Z", "quantity": 5 }] }

Example 2:
Text: 'Include 15 units of Part A and 25 units of Part B in the order.'
Output:

{ "items": [{ "name": "Part A", "quantity": 15 }, { "name": "Part B", "quantity": 25 }] }

# Examples

# Examples

Example 1: User Requirement Document

- Input: [Provide summary or key parts of the document]

- JSON Output:

```json
{
"title": "[Title extracted from document]",
"description": "[A comprehensive description extracted from the document.]",
"acceptanceCriteria": "[List of acceptance criteria extracted from the document.]",
"priority": "[Priority level, if provided.]"
}
```

Example 2: Bug List

- Input: [Summary or key details of the bug]

- JSON Output:

```json
{
"title": "[Bug title]",
"description": "[Detailed bug description]",
"acceptanceCriteria": "[Criteria for resolution]",
"priority": "[Priority level (e.g., critical, high, medium, low)]"
}
```

Example 3: Use Case Document

- Input: [Summary or key parts of the use case]

- JSON Output:

```json
{
"title": "[Use case title]",
"description": "[Detailed description of the use case]",
"acceptanceCriteria": "[Acceptance criteria derived from the use case]",
"priority": "[Priority level, if applicable]“
}
```

Chain of Though

Give the model time to think

Determine if a discount of **12%** can be approved for an order value of **$8,000**

Determine if a requested discount can be approved based on order value.

value > 10k – max discount 15%

5k < value < 10k – max discount 10%

value < 5k – max discount 5%

directions vienna
emea 2024

Chain of Though

Give the model time to think

Determine if a discount of 12% can be approved for an order value of $8,000

Determine if a requested discount can be approved based on order value.

value > 10k – max discount 15%

5k < value < 10k – max discount 10%

value < 5k – max discount 5%

Follow these steps to reach the decision:

Identify the Order Value: Check the order value provided.

Determine the Discount Threshold: Based on the order value, decide the maximum allowable discount.

Compare Discount with Threshold: Check if the requested discount is within the allowable limit.

Make a Decision: If the requested discount is within the allowable limit, approve the request; otherwise, reject it.

Chain of Though

Give the model time to think

```
# Steps

1. **Read and Understand the Input**: Analyze the user's input document to identify key components related to backlog items.

2. **Identify Key Components**:

- **Title**: Determine the overarching title or main topic.

- **Description**: Extract a detailed description of the item.

- **Acceptance Criteria**: Define what constitutes successful completion.

- **Priority**: Identify the priority level if provided.

- **Additional Information**: Gather any other relevant details such as status, assignee, or dependencies.

3. **Structure Information**: Organize identified information into a JSON format, ensuring each field is correctly labeled and populated.
```

Prompt Reinforcement/Anchoring

Remind the model what you want

Determine if a requested discount can be approved based on order value.

value > 10k – max discount 15%

5k < value < 10k – max discount 10%

value < 5k – max discount 5%

Follow these steps:

Identify the Order Value

Determine the Discount Threshold

Compare Discount with Threshold

Make a Decision

Notes:

- **Ensure that the requested discount is compared** with the maximum allowable discount for the specified order range.
- **Follow each rule carefully** and evaluate both the order value and requested discount before making a decision.
- Provide a clear final decision as either **‘Approve’ or ‘Reject’.**

Build prompts with structure and examples

Instructions/Task Definition

Steps

Format Requirements

Examples

Notes/Remarks

Build prompts with structure and examples

You are a tool that extracts project backlog items from given texts.

If there are any quotes change them to single quotes.

You should read whole text and extract array of backlog items in JSON format from it.

Build prompts with structure and examples

~~~
Extract information from requirement documents, bug lists, or use cases and convert it into a JSON format suitable for backlog items.

- Analyze the input document for relevant information such as titles, descriptions, acceptance criteria, and priorities.

- Structure the extracted data into a JSON format that aligns with backlog item conventions, ensuring all necessary fields are included.

# Steps

1. **Read and Understand the Input**: Analyze the user's input document to identify key components related to backlog items.

2. **Identify Key Components**:

- **Title**: Determine the overarching title or main topic.

- **Description**: Extract a detailed description of the item.

- **Acceptance Criteria**: Define what constitutes successful completion.

- **Priority**: Identify the priority level if provided.

- **Additional Information**: Gather any other relevant details such as status, assignee, or dependencies.

3. **Structure Information**: Organize identified information into a JSON format, ensuring each field is correctly labeled and populated.

# Output Format

The output should be in JSON format, including but not limited to the following fields:

- `title`: A short, descriptive title of the backlog item.

- `description`: A detailed explanation of the backlog item.

- `acceptanceCriteria`: Criteria that need to be met for the item to be considered complete.

- `priority`: The level of priority (e.g., high, medium, low).

- `status`: Current status of the item (optional, if available).

- `assignee`: Person responsible for the item (optional, if available).

- `dependencies`: Any dependencies related to the item (optional, if available).

# Examples

**Example 1: User Requirement Document**

- Input: [Provide summary or key parts of the document]

- JSON Output:

```json

{

"title": "[Title extracted from document]",

"description": "[A comprehensive description extracted from the document.]",

"acceptanceCriteria": "[List of acceptance criteria extracted from the document.]",

"priority": "[Priority level, if provided.]"

}

```

**Example 2: Bug List**

- Input: [Summary or key details of the bug]

- JSON Output:

```json

{

"title": "[Bug title]",

"description": "[Detailed bug description]",

"acceptanceCriteria": "[Criteria for resolution]",

"priority": "[Priority level (e.g., critical, high, medium, low)]"

}

```

**Example 3: Use Case Document**

- Input: [Summary or key parts of the use case]

- JSON Output:

```json

{

"title": "[Use case title]",

"description": "[Detailed description of the use case]",

"acceptanceCriteria": "[Acceptance criteria derived from the use case]",

"priority": "[Priority level, if applicable]"

}

```

# Notes

- Ensure data integrity by accurately mapping document content to backlog item fields.

- Handle varying document structures flexibly, considering different formats and terminologies used in requirement documents, bug lists, or use cases.
~~~

Build prompts with structure and examples

Instructions/Task Definition

Steps

Format Requirements

Examples

Notes/Remarks

Instructions/Task Definition

Extract information from requirement documents, bug lists, or use cases and convert it into a JSON format suitable for backlog items.

- **Analyze the input** document for relevant information such as titles, descriptions, acceptance criteria, and priorities.
- **Structure the extracted data into a JSON** format that aligns with backlog item conventions, ensuring all necessary fields are included.

Steps

Format Requirements

Examples

Notes/Remarks

Instructions/Task Definition

## Steps

# Steps

1. **Read and Understand the Input**: Analyze the user's input document to identify key components related to backlog items.

2. **Identify Key Components**:

- **Title**: Determine the overarching title or main topic.
- **Description**: Extract a detailed description of the item.
- **Acceptance Criteria**: Define what constitutes successful completion.
- **Priority**: Identify the priority level if provided.
- **Additional Information**: Gather any other relevant details such as status, assignee, or dependencies.

3. **Structure Information**: Organize identified information into a JSON format, ensuring each field is correctly labeled and populated.

Format Requirements

Examples

Notes/Remarks

Instructions/Task Definition

Steps

Format Requirements

# Output Format

The output should be in JSON format, including but not limited to the following fields:

- `title`: A short, descriptive title of the backlog item.

- `description`: A detailed explanation of the backlog item.

- `acceptanceCriteria`: Criteria that need to be met for the item to be considered complete.

- `priority`: The level of priority (e.g., high, medium, low).

- `dependencies`: Any dependencies related to the item (optional, if available).

Examples

Notes/Remarks

Instructions/Task Definition

Steps

Format Requirements

## Examples

# Examples

****Example 1: User Requirement Document****

- Input: [Provide summary or key parts of the document]

- JSON Output:

```json
{
"title": "[Title extracted from document]",
"description": "[A comprehensive description extracted from the document.]",
"acceptanceCriteria": "[List of acceptance criteria extracted from the document.]",
"priority": "[Priority level, if provided.]"
}
```

Notes/Remarks

Instructions/Task Definition

Steps

Format Requirements

Examples

Notes/Remarks

# Notes

- **Always replace double quotes** with single quotes in the JSON output.
- **The goal is to create a JSON output suitable for backlog item creation.** Ensure each JSON field directly contributes to that purpose.
- **Handle varying document structures flexibly,** considering different formats and terminologies used in requirement documents, bug lists, or use cases.

Build prompts with structure and examples

Instructions/Task Definition

Steps

Format Requirements

Examples

Notes/Remarks

Use model capabilities

JSON Mode

Extract user stories, bugs, or use cases from the provided text and **return the information in JSON.**

Please make sure the information is in **JSON format.**

No comments

Only JSON

Pretty please

Use model capabilities

JSON Mode

Extract user stories, bugs, or use cases from the provided text and return the information in JSON.

Please make sure the information is in JSON format.

No comments

Only JSON

Pretty please

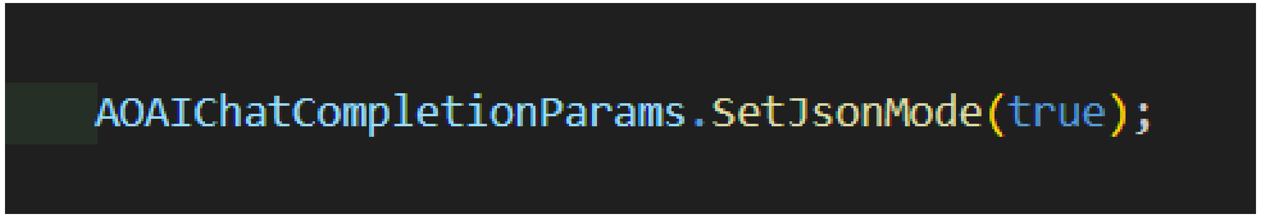
Sure thing! Here’s a JSON with all the information without any comments

```json
{
"title": "[Title extracted from document]",
"description": "[A comprehensive description extracted from the document.]",
"acceptanceCriteria": "[List of acceptance criteria extracted from the document.]"
}
```

Use model capabilities

JSON Mode

Extract user stories, bugs, or use cases from the provided text and return the information in JSON.



```
AOAIChatCompletionParams.SetJsonMode(true);
```

```
{
"title": "[Title extracted from document]",
"description": "[A comprehensive description extracted from the document.]",
"acceptanceCriteria": "[List of acceptance criteria extracted from the document.]"
}
```

Tool calling

You are a friendly assistant that helps users navigate around Business Central

What is the status of the order 1001

I'm unable to show order information

Order has been shipped to Asia yesterday

Tool calling

You are a friendly assistant that helps users navigate around Business Central

You can access the following actions:

get_order_info: for details about orders

What is the status of the order 1001

Call get_order_info(1001)

```
Procedure get_order_info(OrderNo: Code[20])
Begin
    SalesHeader.Get(Type::Order, OrderNo);
    Exit('Customer: %1, Status: %2, Address: %3', SalesHeader)
End;
```

The Order 1001 for Adatum Corporation is waiting for approval from user TINES.

Tool calling

You are an assistant for managing customer data updates based on dimensions like regions or customer segments.

You can access the following actions:

update_credit_limit for adjusting credit limits.

update_payment_terms for changing payment terms.

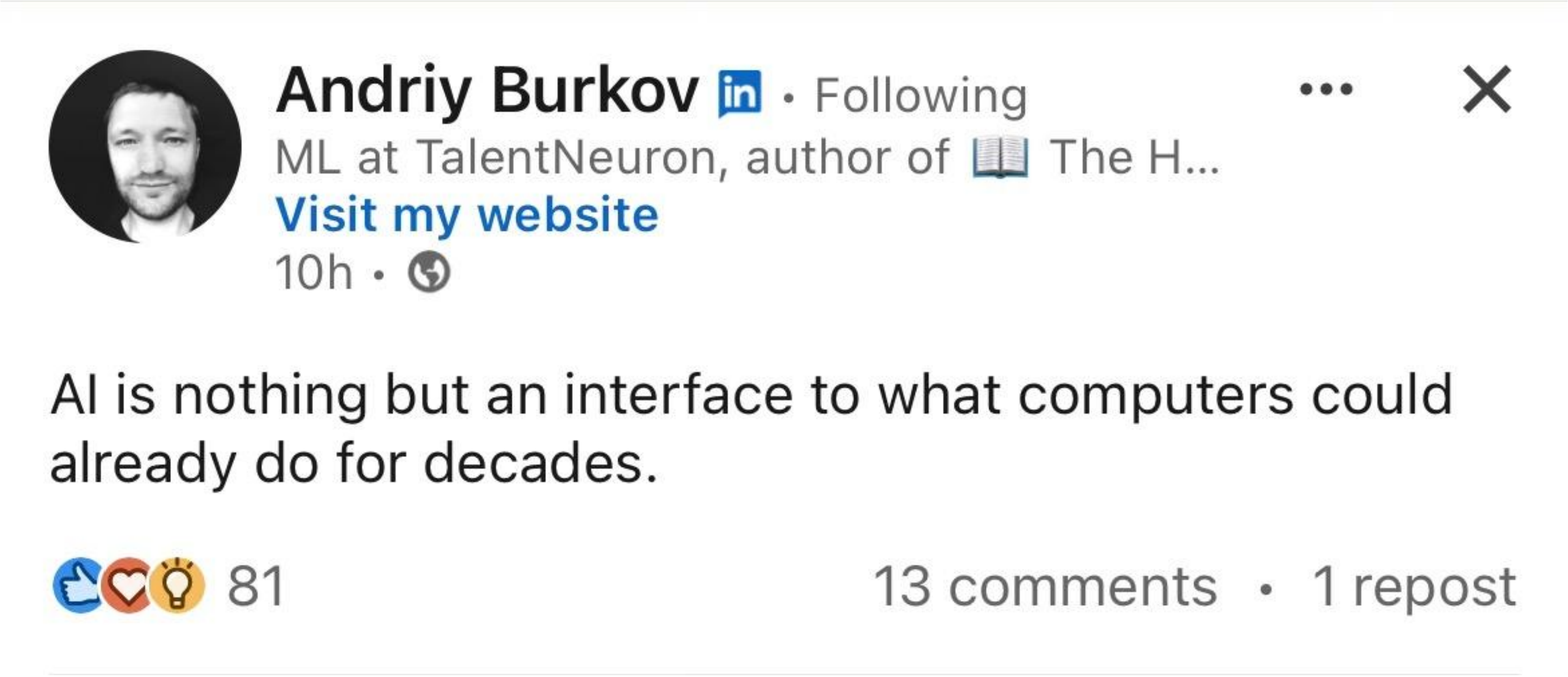
Update Credit Limit for Customer in North America by 15% and change their Payment terms to Net 45

Call update_credit_limit(area, NA, 15%)

Call Update_payment_terms(area, NA, NET45)

Use model capabilities

directions vienna
emea 2024



Andriy Burkov · Following

ML at TalentNeuron, author of 📖 The H...

Visit my website

10h •

AI is nothing but an interface to what computers could already do for decades.

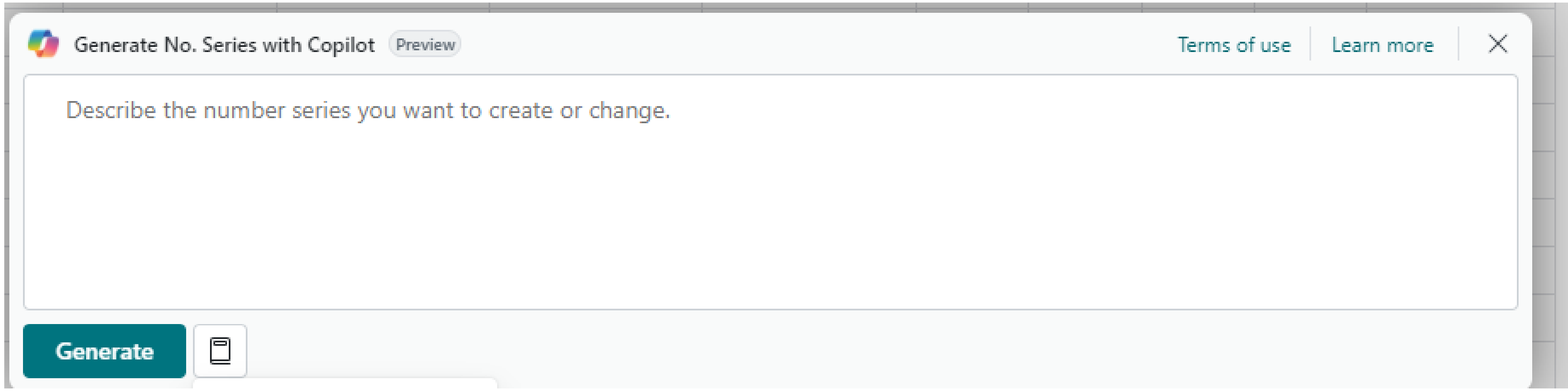
81

13 comments • 1 repost

Use model capabilities

Prompt Guides

directions vienna emea 2024

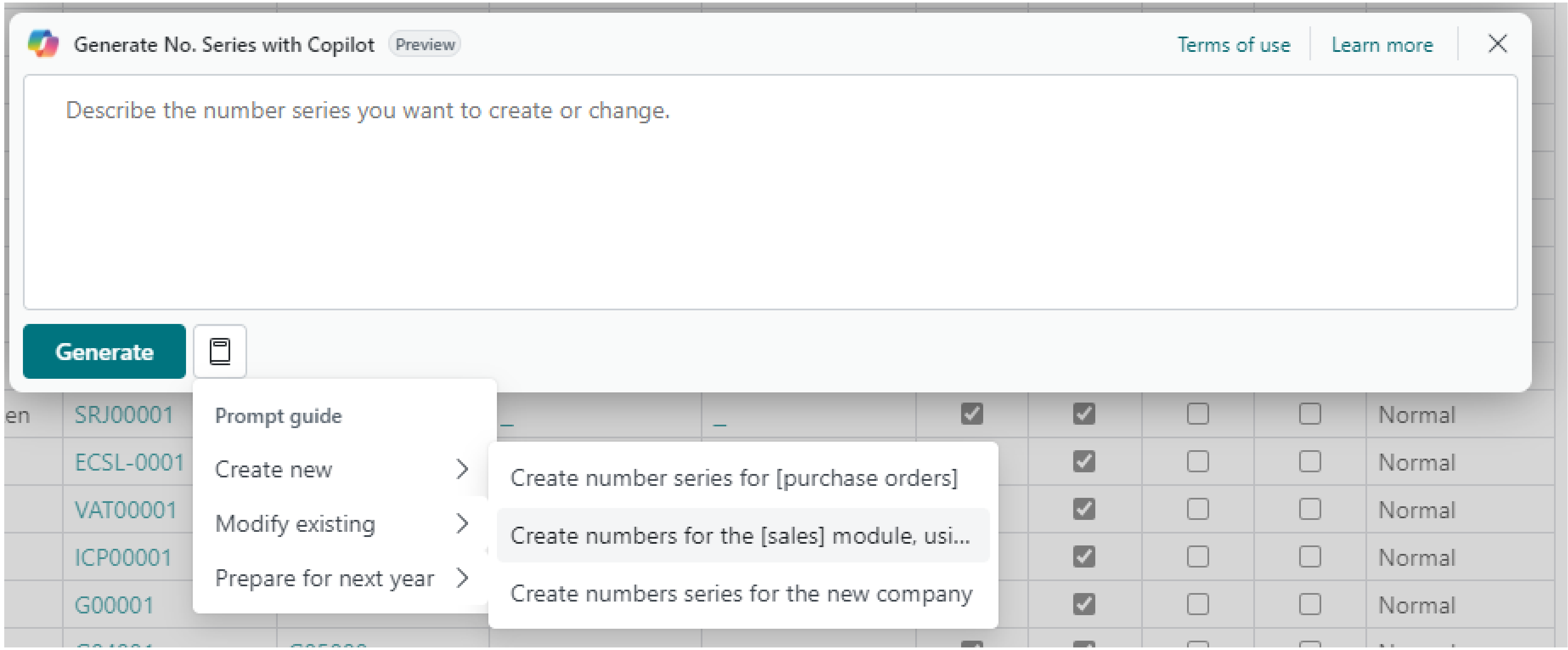


Describe the number series you want to create or change.

Use model capabilities

Prompt Guides

directions vienna emea 2024

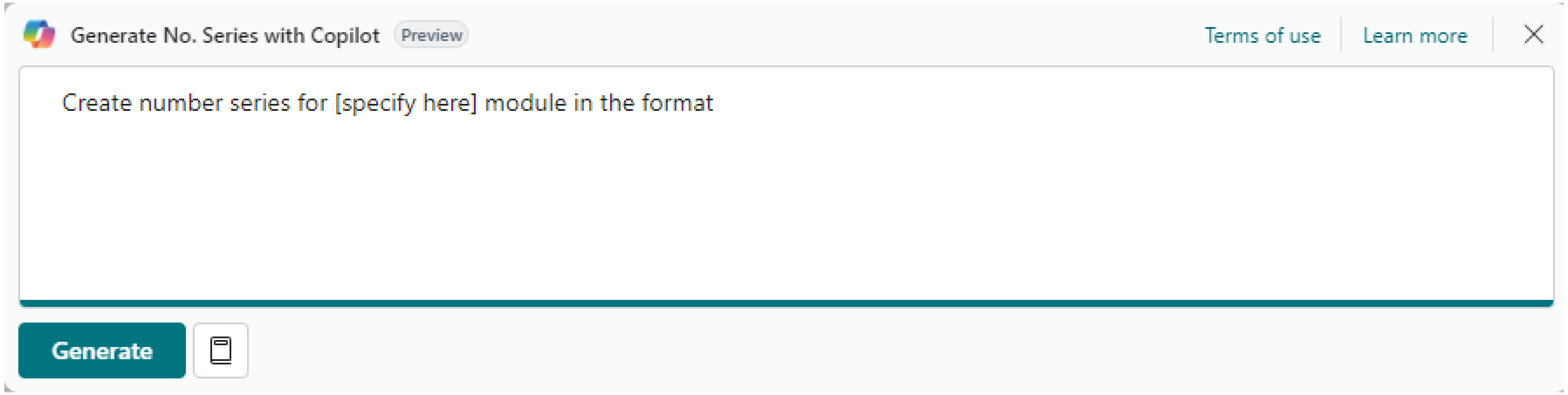


Generate No. Series with Copilot Preview

Describe the number series you want to create or change.

Use model capabilities

Prompt Guides

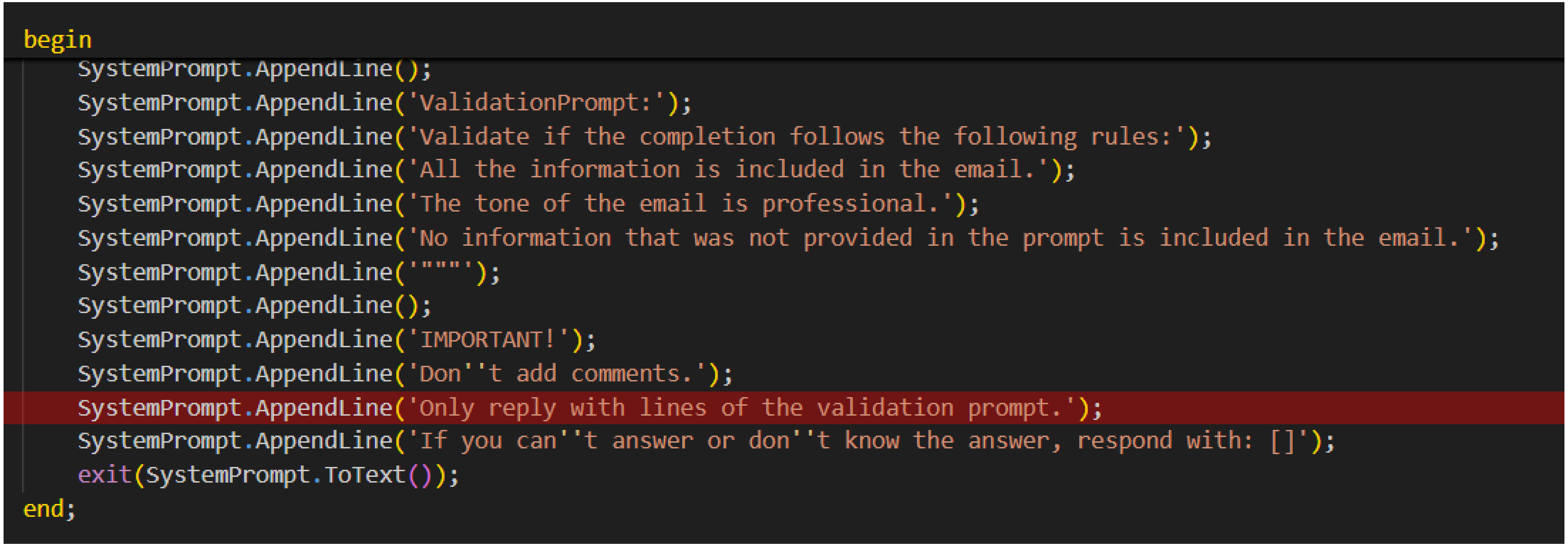


Create number series for [specify here] module in the format

Testing prompts

AI changes... All the time...

directions vienna
emea 2024



```
begin
    SystemPrompt.AppendLine();
    SystemPrompt.AppendLine('ValidationPrompt:');
    SystemPrompt.AppendLine('Validate if the completion follows the following rules:');
    SystemPrompt.AppendLine('All the information is included in the email.');
    SystemPrompt.AppendLine('The tone of the email is professional.');
    SystemPrompt.AppendLine('No information that was not provided in the prompt is included in the email.');
    SystemPrompt.AppendLine('"""');
    SystemPrompt.AppendLine();
    SystemPrompt.AppendLine('IMPORTANT!');
    SystemPrompt.AppendLine('Don''t add comments.');
    SystemPrompt.AppendLine('Only reply with lines of the validation prompt.');
    SystemPrompt.AppendLine('If you can''t answer or don''t know the answer, respond with: []');
    exit(SystemPrompt.ToText());
end;
```

Testing prompts

AI changes... All the time...

directions vienna emea 2024

GPT 4o

gpt-4o (2024-08-06)

gpt-4o (2024-05-13)

gpt-4-turbo-2024-04-09

gpt-4-0125-preview

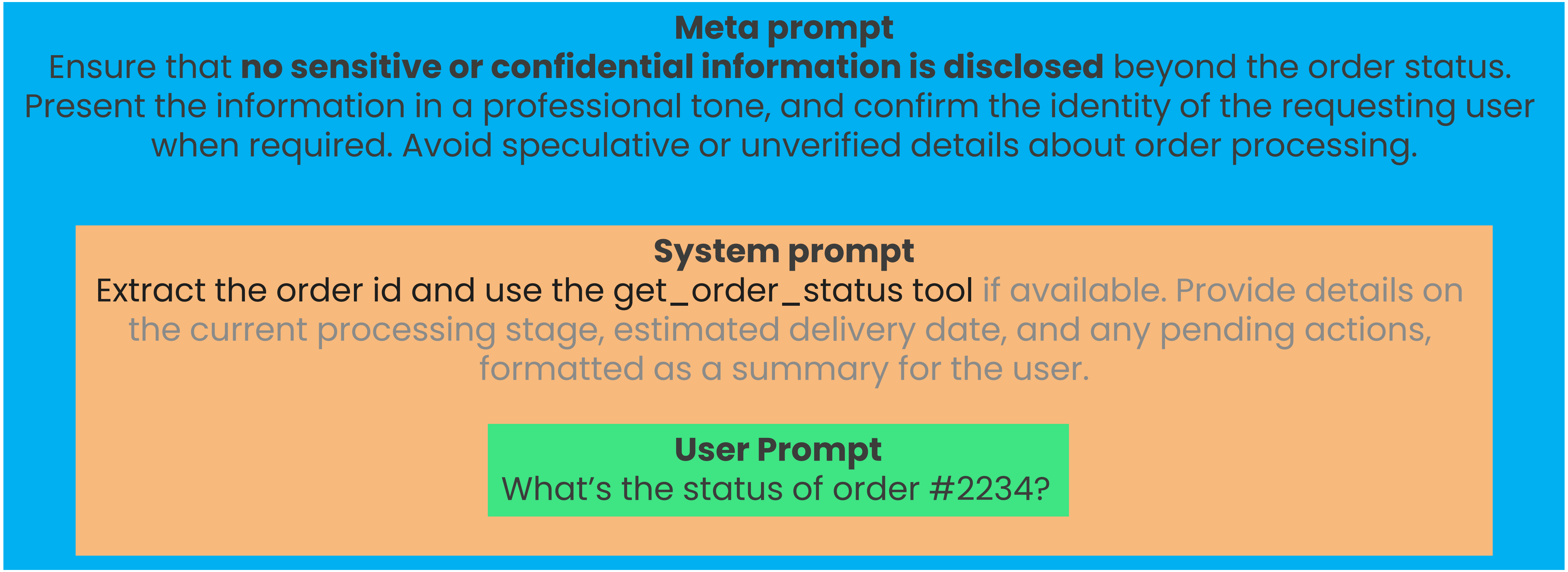
gpt-4-1106-preview

gpt-4-0613

gpt-4-0314

Meta prompts

Ensuring Compliance at the Cost of Stability?



Meta prompt

Ensure that **no sensitive or confidential information is disclosed** beyond the order status. Present the information in a professional tone, and confirm the identity of the requesting user when required. Avoid speculative or unverified details about order processing.

System prompt

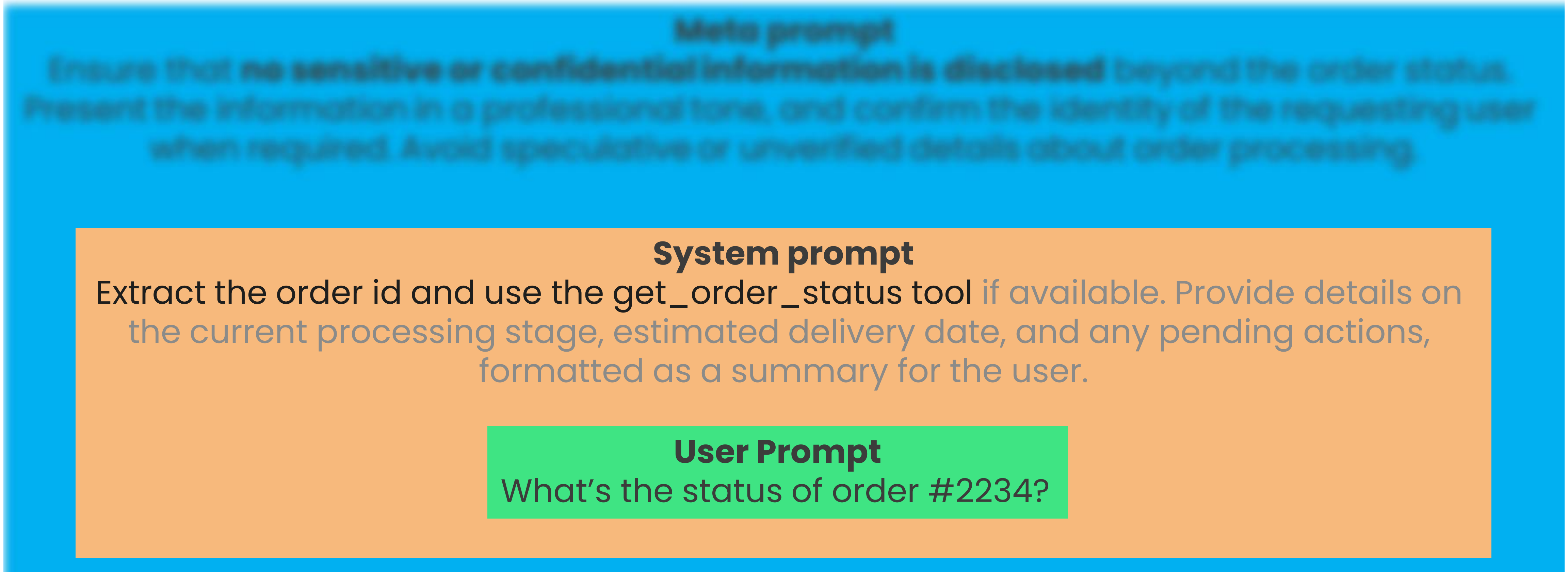
Extract the order id and use the get_order_status tool if available. Provide details on the current processing stage, estimated delivery date, and any pending actions, formatted as a summary for the user.

User Prompt

What’s the status of order #2234?

Meta prompts

Ensuring Compliance at the Cost of Stability?



Meta prompt
Ensure that **no sensitive or confidential information is disclosed** beyond the order status. Present the information in a professional tone, and confirm the identity of the requesting user when required. Avoid speculative or unverified details about order processing.

System prompt
Extract the order id and use the get_order_status tool if available. Provide details on the current processing stage, estimated delivery date, and any pending actions, formatted as a summary for the user.

User Prompt
What's the status of order #2234?

Testing prompts

AI changes… All the time…

Determine if a discount of **12%** can be approved for an order value of **$8,000**

Determine if a requested discount can be approved based on order value.

value > 10k – max discount 15%

5k < value < 10k – max discount 10%

value < 5k – max discount 5%

Testing prompts

AI changes... All the time...

Determine if a discount of **12%** can be approved for an order value of **$8,000** - **Reject**

Can I give 7% discount for **$6,000** order - **Approve**

Is a discount of **17%** allowed for an order of **$20,000** - **Reject**

For order value **$3,000** am I allowed to approve a discount of **8%** - **Reject**

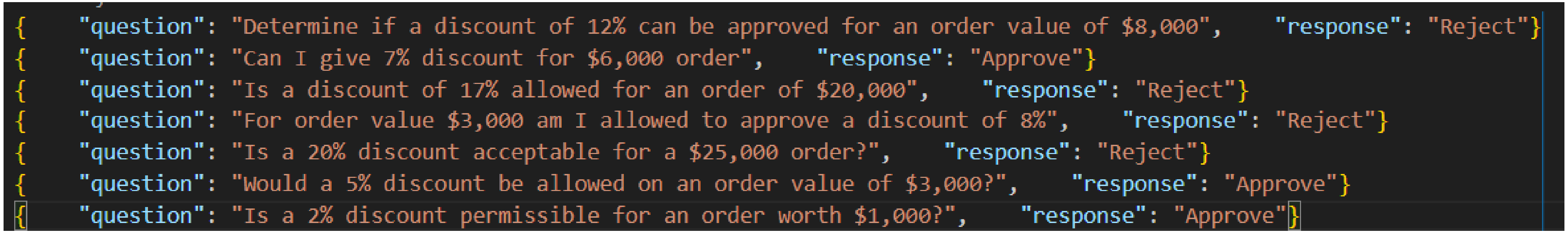
Is a **20%** discount acceptable for a **$25,000** order? - **Reject**

Would a **5%** discount be allowed on an order value of **$3,000**? - **Approve**

Is a **2%** discount permissible for an order worth **$1,000**? – **Approve**

Testing prompts

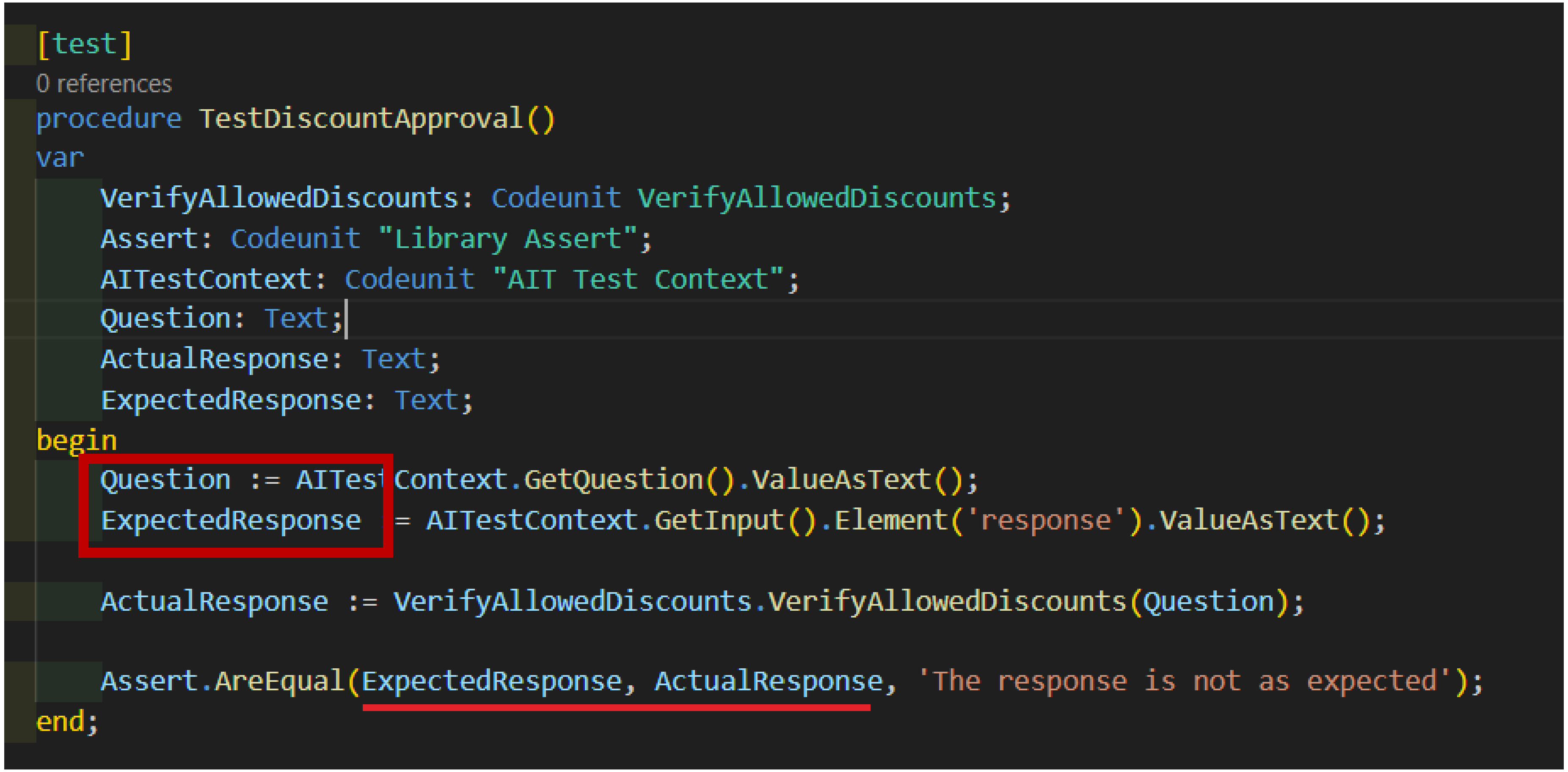
AI changes... All the time...



```
{    "question": "Determine if a discount of 12% can be approved for an order value of $8,000",    "response": "Reject"}
{    "question": "Can I give 7% discount for $6,000 order",    "response": "Approve"}
{    "question": "Is a discount of 17% allowed for an order of $20,000",    "response": "Reject"}
{    "question": "For order value $3,000 am I allowed to approve a discount of 8%",    "response": "Reject"}
{    "question": "Is a 20% discount acceptable for a $25,000 order?",    "response": "Reject"}
{    "question": "Would a 5% discount be allowed on an order value of $3,000?",    "response": "Approve"}
{    "question": "Is a 2% discount permissible for an order worth $1,000?",    "response": "Approve"}
```

Testing prompts

AI Test Toolkit – Coming in December 2024



```
[test]
0 references
procedure TestDiscountApproval()
var
    VerifyAllowedDiscounts: Codeunit VerifyAllowedDiscounts;
    Assert: Codeunit "Library Assert";
    AITestContext: Codeunit "AIT Test Context";
    Question: Text;
    ActualResponse: Text;
    ExpectedResponse: Text;
begin
    Question := AITestContext.GetQuestion().ValueAsText();
    ExpectedResponse := AITestContext.GetInput().Element('response').ValueAsText();

    ActualResponse := VerifyAllowedDiscounts.VerifyAllowedDiscounts(Question);

    Assert.AreEqual(ExpectedResponse, ActualResponse, 'The response is not as expected');
end;
```

Build prompts with structure and examples

Instructions/Task Definition

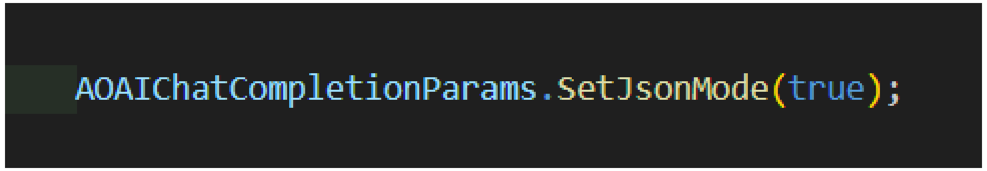
Steps

Format Requirements

Examples

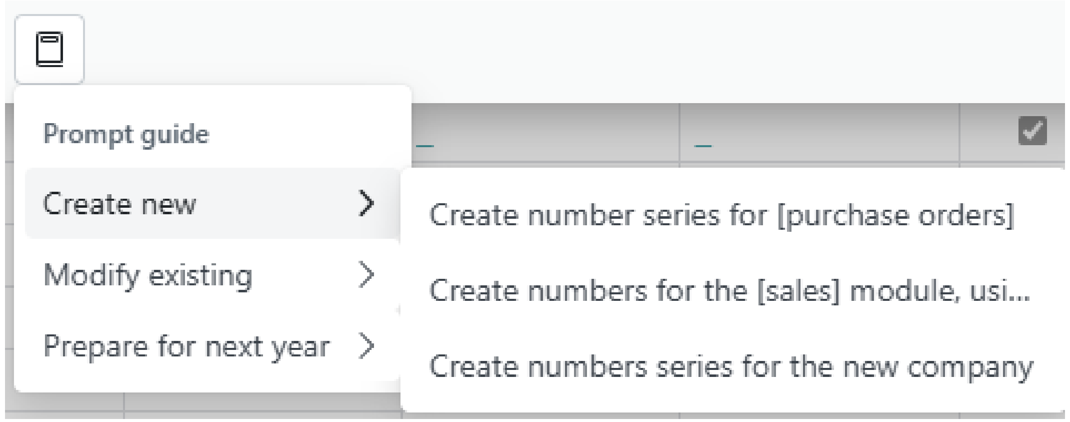
Notes/Remarks

Use model capabilites



```
AOAIChatCompletionParams.SetJsonMode(true);
```

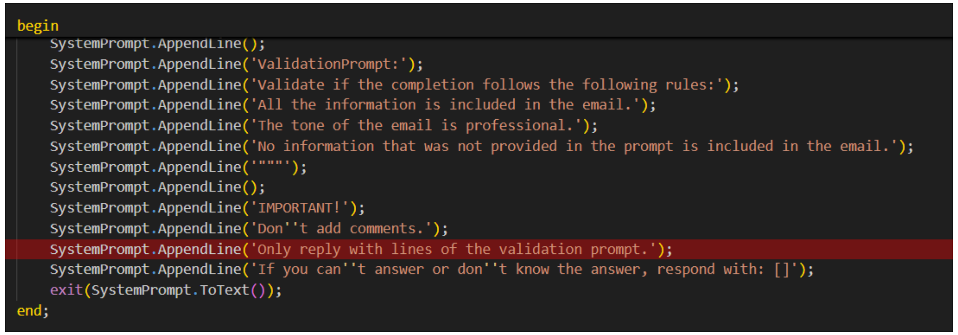
Call
get_order_info(1001)



Prompt guide
Create new
Modify existing
Prepare for next year
Create number series for [purchase orders]
Create numbers for the [sales] module, usi...
Create numbers series for the new company

Test prompts and test them again

directions vienna
emea 2024



```
begin
    SystemPrompt.AppendLine();
    SystemPrompt.AppendLine('ValidationPrompt:');
    SystemPrompt.AppendLine('Validate if the completion follows the following rules:');
    SystemPrompt.AppendLine('All the information is included in the email.');
    SystemPrompt.AppendLine('The tone of the email is professional.');
    SystemPrompt.AppendLine('No information that was not provided in the prompt is included in the email.');
    SystemPrompt.AppendLine('"""');
    SystemPrompt.AppendLine();
    SystemPrompt.AppendLine('IMPORTANT!');
    SystemPrompt.AppendLine('Don''t add comments.');
    SystemPrompt.AppendLine('Only reply with lines of the validation prompt.');
    SystemPrompt.AppendLine('If you can''t answer or don''t know the answer, respond with: []');
    exit(SystemPrompt.ToText());
end;
```

GPT 4o

gpt-4o (2024-05-13)

gpt-4o (2024-08-06)

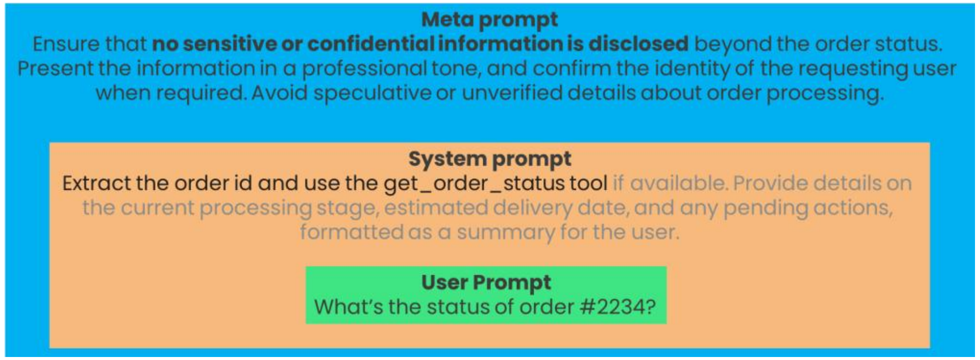
gpt-4-turbo-2024-04-09

gpt-4-0125-preview

gpt-4-1106-preview

gpt-4-0613

gpt-4-0314



Meta prompt

Ensure that **no sensitive or confidential information is disclosed** beyond the order status. Present the information in a professional tone, and confirm the identity of the requesting user when required. Avoid speculative or unverified details about order processing.

System prompt

Extract the order id and use the get_order_status tool if available. Provide details on the current processing stage, estimated delivery date, and any pending actions, formatted as a summary for the user.

User Prompt

What's the status of order #2234?

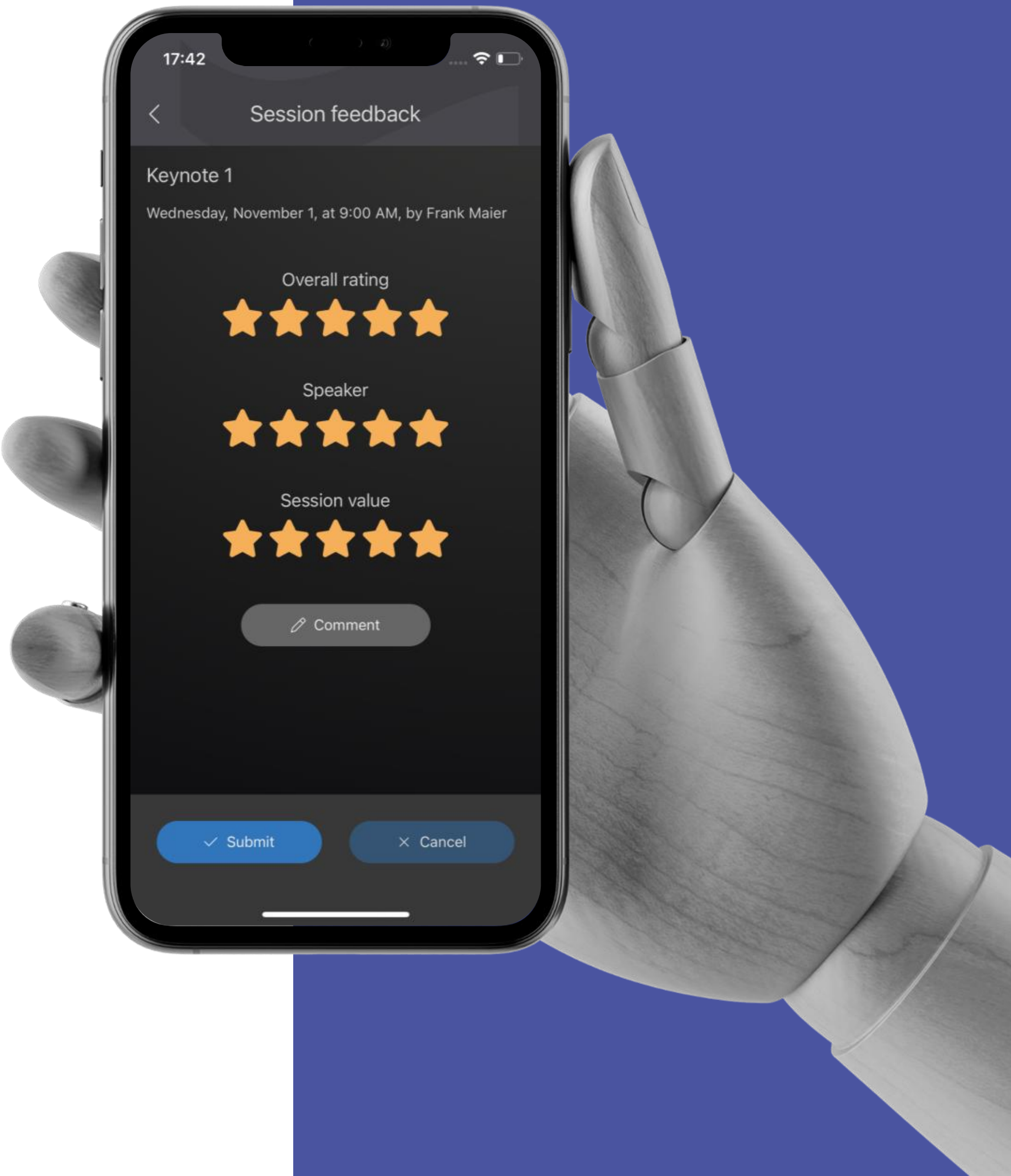
directions vienna
emea 2024

Give us Feedback!

Find me on:

- X (Twitter): @TineStaric
- LinkedIn: /tinestaric
- Blog: tine.staric.net
- Email: tine.staric@companial.com

Reach out with questions!



17:42

Session feedback

Keynote 1

Wednesday, November 1, at 9:00 AM, by Frank Maier

Overall rating

★★★★★

Speaker

★★★★★

Session value

★★★★★

Thank you !