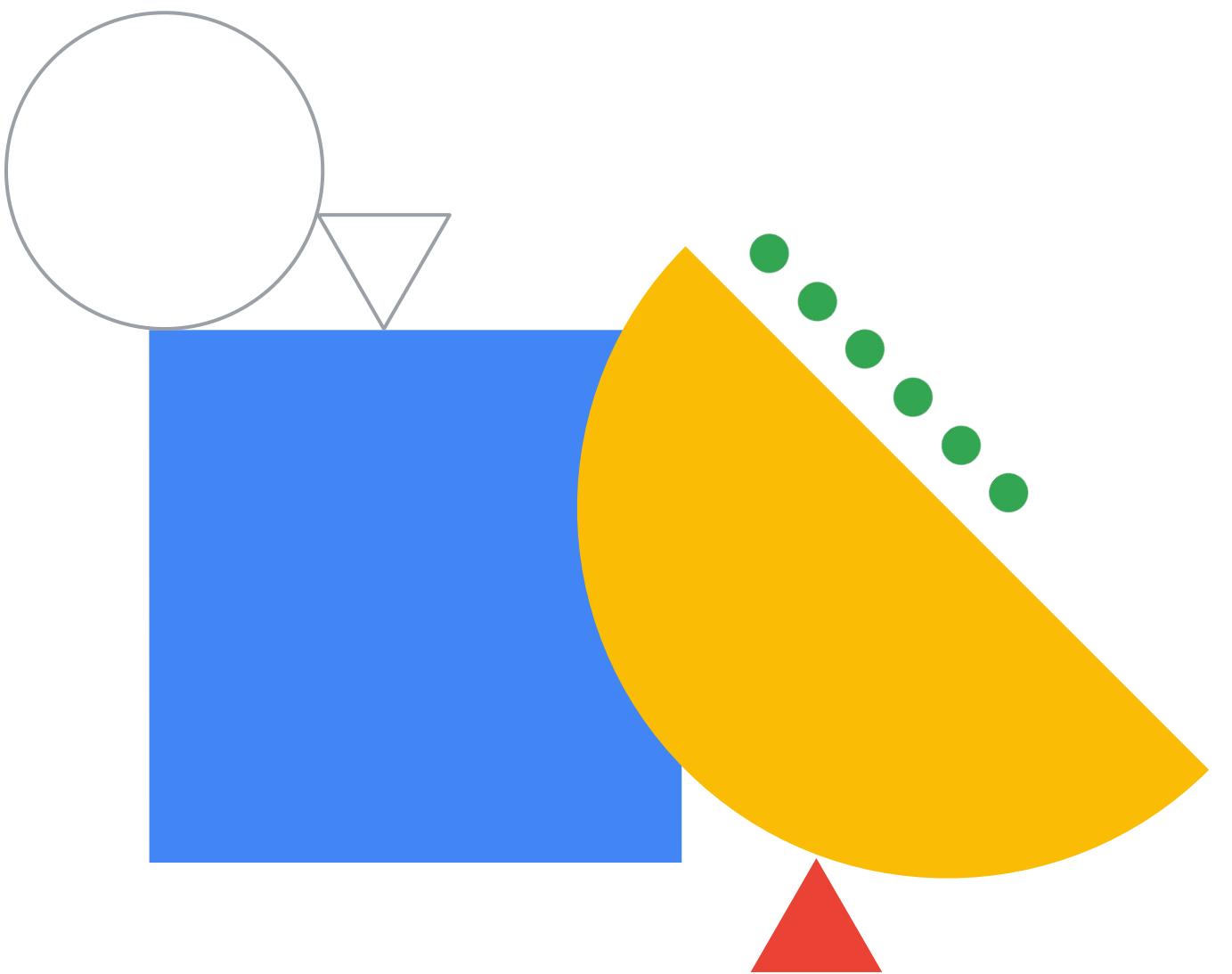
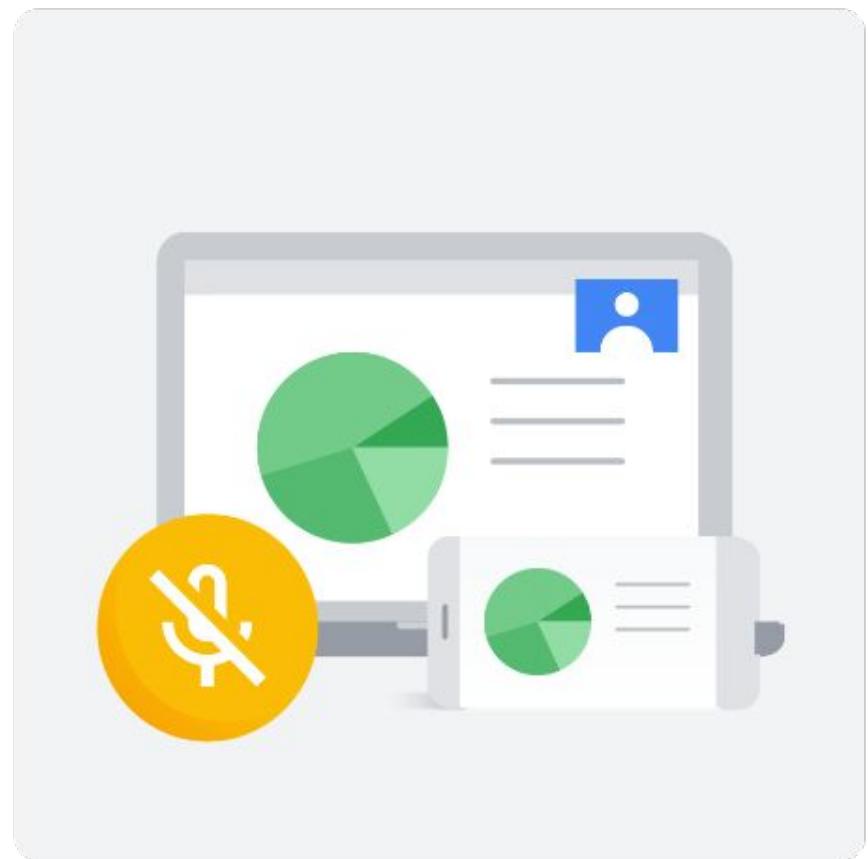


# Industry Use Cases for Generative AI

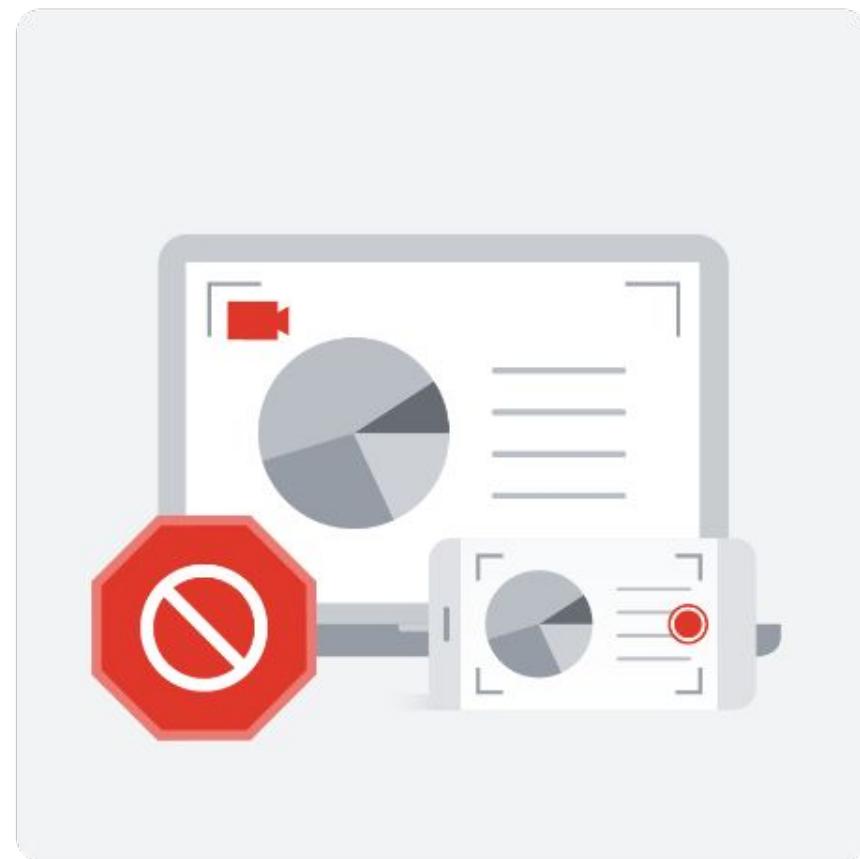


Please mark your attendance at  
[goo.gle/genai-checkin](https://goo.gl/genai-checkin)

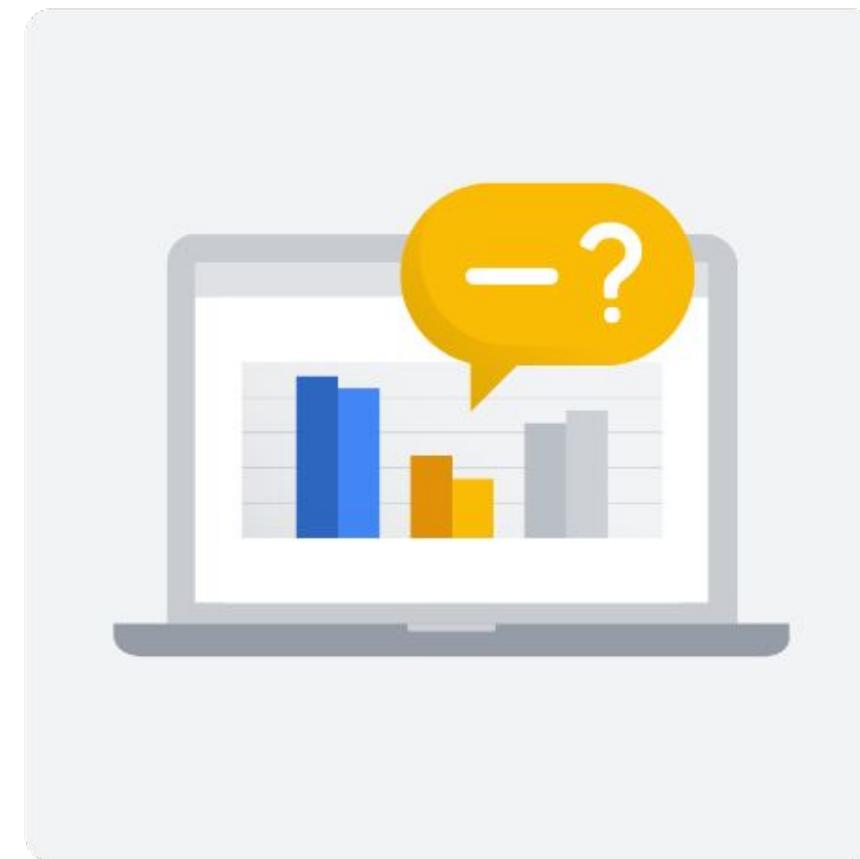
# Etiquette



Mute microphone

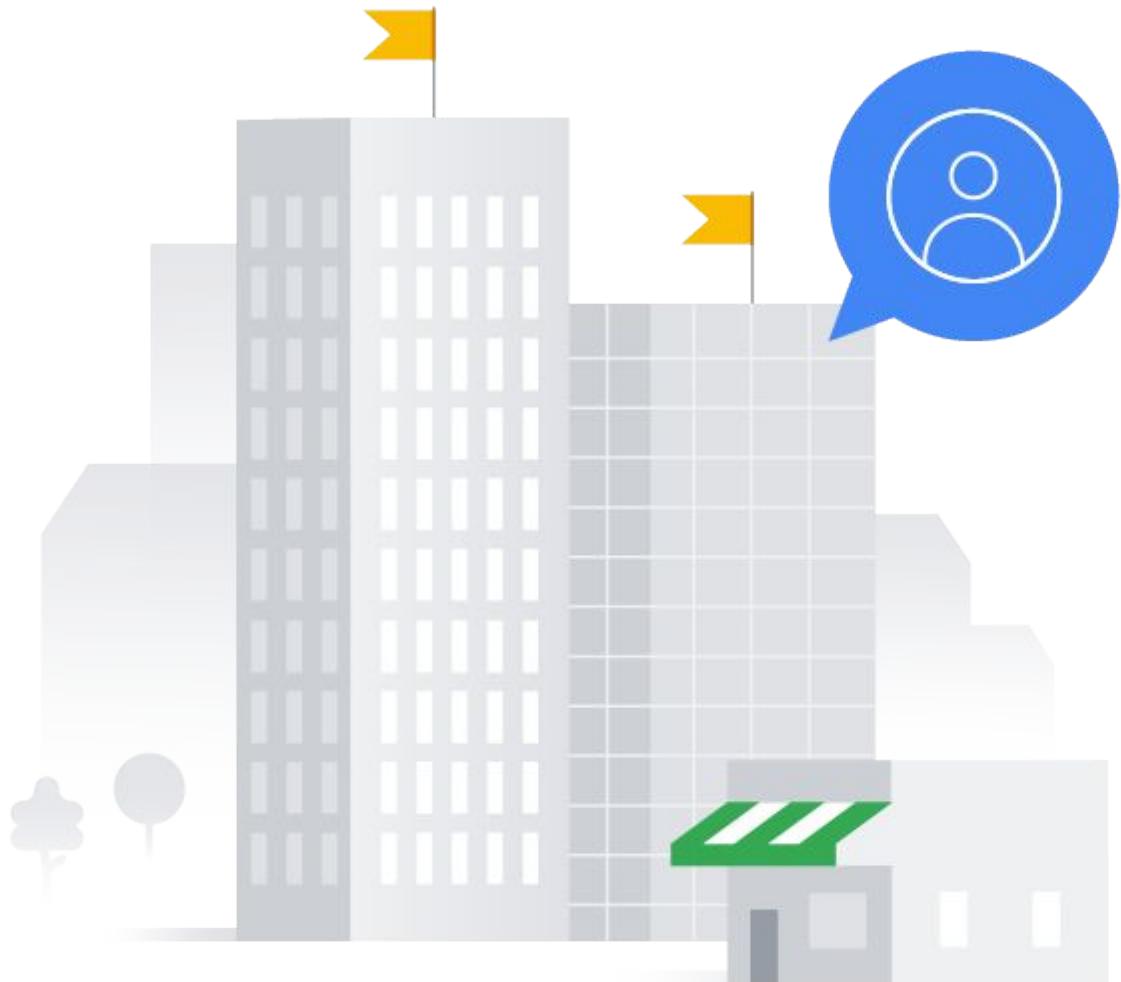


No recording



Ask questions

# Target audience



- Enterprise Business Leaders
- Business End Users
- Developers

Google Cloud

# Helpful knowledge



No Prereqs



Helpful to be familiar with



Cloud computing



Google Cloud Platform

# In this course, you learn ...

- 01 What Generative AI can do, and how it works
- 02 How Generative AI solutions can be built for real-world use cases in various industries
- 03 Which Google Cloud products offer Generative AI functionality, and when to use them
- 04 How to get started with Generative AI in your organization





# Generative AI in Action

# In this module, you learn...

01

What Generative AI (GenAI) can do

02

How Generative AI functionality can enable critical workflows across industries

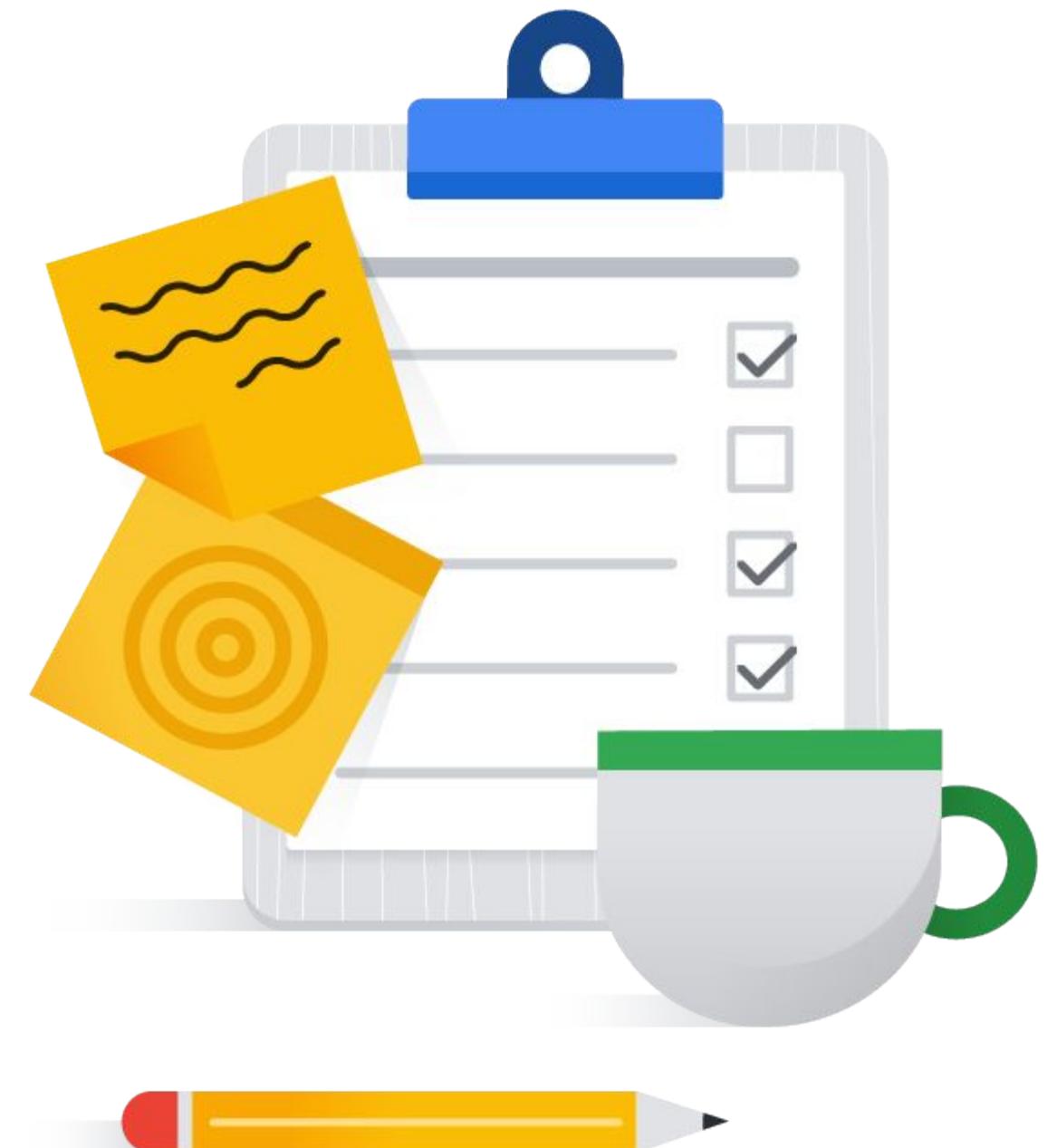
03

How production-grade solutions are built by integrating multiple GenAI products



# Topics

- 01 What is Generative AI?
- 02 Real-world examples
- 03 Example architectures



# GenAI can solve different classes of problems



## CREATE

Bring your thoughts and visions to life



## SUMMARIZE

Condense and summarize your knowledge base into a simple format



## DISCOVER

Help your customers and employees find what they need at the right time



## AUTOMATE

Automate your customer service across multiple channels

### Use cases

- Generate product descriptions from images
- Generate images from text

### Use cases

- Content/video summarization
- Intra-knowledge Q&A

### Use cases

- Search for a document
- Machine-generated event monitoring

### Use cases

- Contract information extraction
- Feedback classification and ticket creation

# Topics

- 01 What is Generative AI?
- 02 Real-world examples
- 03 Example architectures



**AI can improve the product discovery experience, increase basket size, and reduce cart abandonment.**

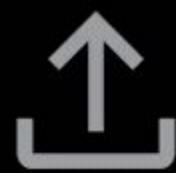
**72% of consumers'**  
favorite retail websites are ones where they  
can quickly find what they are looking for.

**72% of consumers**  
buy the product they're looking for  
when they can find it quickly and easily.

Reference: [GOOGLE CLOUD: New Research on Search Abandonment in Retail \(March '23\)](#)



- Add New Product



Upload



- Add New Product





Add New Product · Categorize · Assign Attributes · Sort

Your product has been successfully categorized and sorted.

Would you like to create content for the product description page now?

Yes

Back

## Product Description Page

✍ Tell me what you need to build

Write a detailed product description of this product for inclusion in the Spring collection, based on the product attribute list

*This Spring, you can't beat a classic. This shirt has timeless styling - mid-weight pinpoint cotton, button cuffs and collar, and a tailored but not-too-tight fit - combined with modern features like wrinkle resistance and fade-free dye. 100% cotton, traditional blue.*

## Metadata:

```
content="width=device-width, initial-scale=1.0">
<meta name="keywords" content="Men's Oxford Shirt
[Traditional]">
```

I need to make edits

Looks good 

## Live Preview

Cymbal Superstore  

ALL PRODUCTS / MENSWEAR / SHIRTS / LONG SLEEVE / BUTTON DOWNS & OXFORDS



**Men's Wrinkle-Resistant Oxford Button-Down Shirt**

\$29.00

This Spring, you can't beat a classic. This shirt has timeless styling - mid-weight pinpoint cotton, button cuffs and collar, and a tailored but not-too-tight fit - combined with modern features like wrinkle resistance and fade-free dye. 100% cotton, traditional blue.

MATERIAL  
SIZE & FIT  
CARE  
DELIVERY  
RETURNS



## Product Description Page

Tell me what you need to build

Write a detailed product description of this product for inclusion in the Spring collection, based on the product attribute list

*This Spring, you can't beat a classic. This shirt has timeless styling - mid-weight pinpoint cotton, button cuffs and collar, and a tailored but not-too-tight fit - combined with modern features like wrinkle resistance and fade-free dye. 100% cotton, traditional blue.*

## Metadata:

```
<meta name="Localized" content="Men's Oxford Shirt">  
<meta name="translate">
```

I need to make edits

Looks good

Localize

Create images

## Localization

## ENGLISH

*This Spring, you can't beat a classic. This shirt has timeless styling - mid-weight pinpoint cotton, button cuffs and collar, and a tailored but not-too-tight fit - combined with modern features like wrinkle resistance and fade-free dye. 100% cotton, classic blue.*

## FILIPINO

*Ngayong Spring, hindi mo matatalo ang isang klasiko. Ang kamiseta na ito ay walang tiyak na pag-istilo - mid-weight pinpoint cotton, button cuffs at collar, at isang pinasadya ngunit hindi masyadong masikip na fit - na sinamahan ng mga modernong tampok tulad ng wrinkle resistance ...*

## FRENCH

*Ce printemps, vous ne pouvez pas battre un classique. Cette chemise a un style intemporel - coton pointu d'épaisseur moyenne, poignets et col boutonnés, et une coupe ajustée mais pas trop serrée - combinée à des caractéristiques modernes comme la résistance aux plis et la teinture ...*

## JAPANESE

*この春、クラシックに勝るものはありません。このシャツは、中厚手のピンポイントコットン、ボタン付きの袖口と襟、テーラードでありながらタイトすぎないフィット感など、時代を超えたスタイルと、しわになりにくく色あせしない染料などの現代的な機能を兼ね備えています。綿100%、伝統的なブルー。*

## SPANISH

*Esta primavera, no puedes vencer a un clásico. Esta camisa tiene un estilo atemporal: algodón fino de peso medio, puños y cuello abotonados y un ajuste entallado pero no demasiado ajustado, combinado con características modernas como resistencia a las arrugas y tinte que ...*

## AFRIKAANS

*I vär kan du inte slå en klassiker. Den här skjortan har tidlös styling - mellanvikts fin bomull, manschetter och krage med knappar, och en*

## Live Preview

search product

EAR / SHIRTS / LONG SLEEVE / BUTTON DOWNS & OXFORDS



**Men's Wrinkl Oxford Butt**

\$29.00

This Spring, you can't beat a classic. This shirt has timeless styling - mid-weight pinpoint cotton, button cuffs and collar, and a tailored but not-too-tight fit - combined with modern features like wrinkle resistance and fade-free dye. 100% cotton, traditional blue.

- MATERIAL  
- SIZE & FIT  
- CARE  
- DELIVERY  
- RETURNS

# Let's Discuss: Retail

- In that video, what stood out to you?
  - Generating marketing copy, generating images, translation, categorization
  - Blending of human insight and computer predictions, custom application, leveraging proprietary data
- For those who work in the industry...
  - To what extent have you seen conversations or projects aimed at addressing these problems
  - How valuable would a system like this be?
- For those working in other industries...
  - How could you apply this pattern in your domain?



# AI can help deliver the premium, accessible customer service experience that modern buyers expect.

**of consumers**  
would be willing to pay more for  
a product with premium support

## Immediate results

**82% of consumers** expect an instant response when they have a sales question.

## Speed and ease

**65% of consumers** prefer chatting with customer service bots vs. waiting for a human agent.

Source: [MCKINSEY & COMPANY: Addressing Automotive Customer Experience \(2019\)](#)



Google Cloud



Cymbal

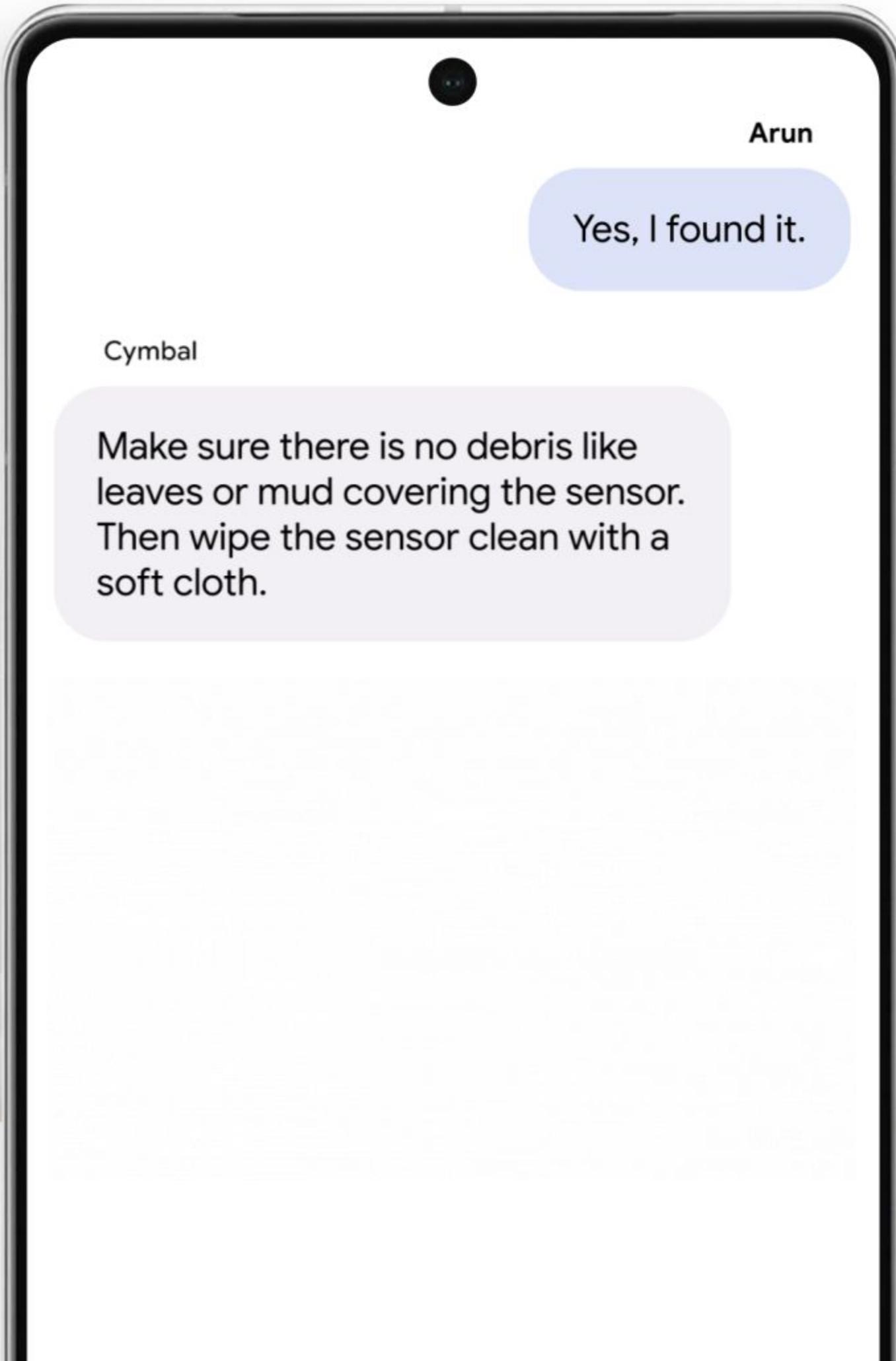
Are you parked safely and  
ready to attempt a potential  
solution?

Arun

Yes thanks

Cymbal

OK, great. The first thing to  
do is check if something is  
blocking the sensor. Please  
make sure the car is in park  
and the parking brake is on.  
Turn the car off and exit the  
vehicle.







Arun

Yes. Can you set this up  
near my office?

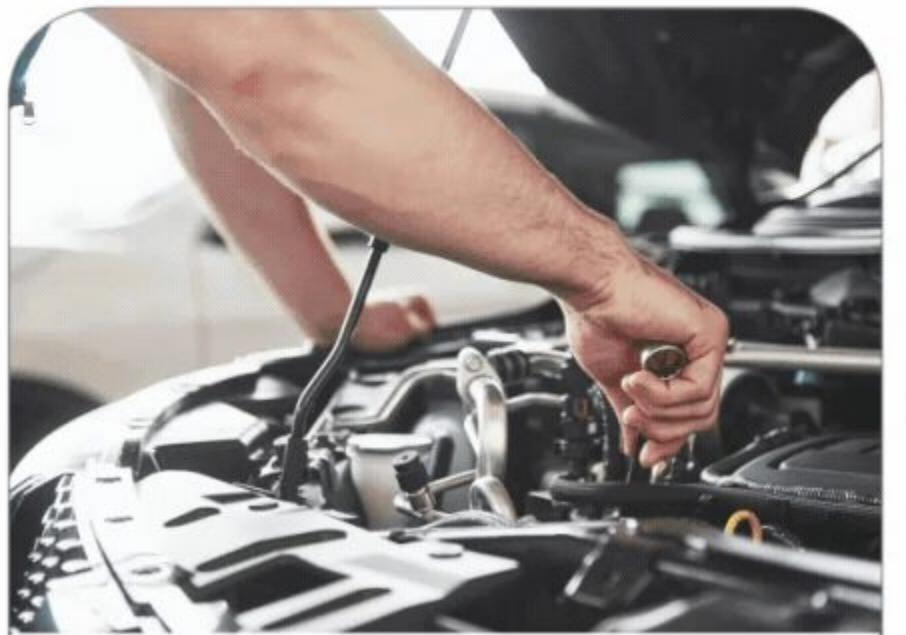
Cymbal

Sure. You previously shared  
your office zip code as 94809.  
Is that still correct?

Arun

Yes

# Search results



Everything Auto

4.7 ★★★★☆ (82)

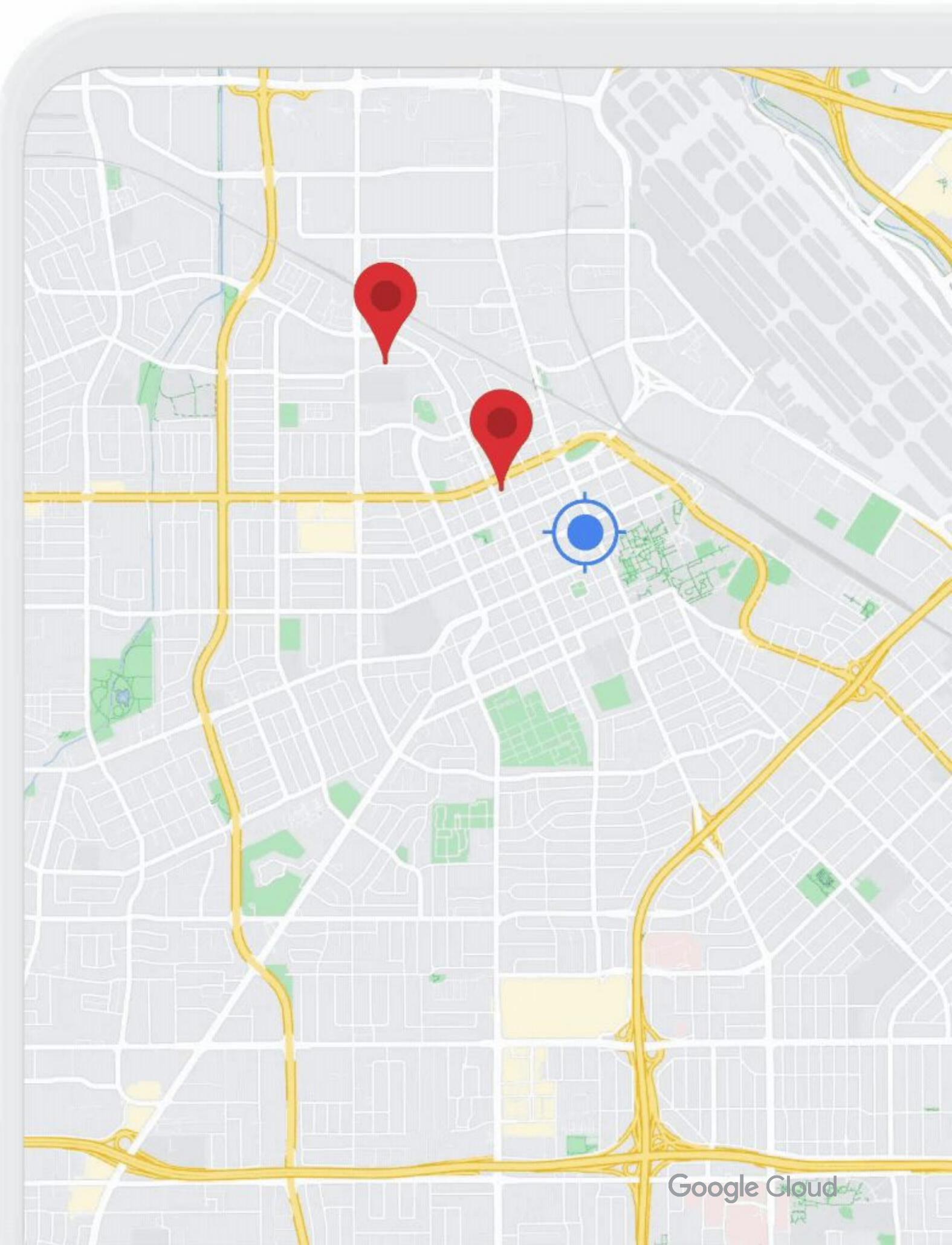
Vehicle repairs



SJ Mechanic Santa Clara

4.2 ★★★★☆ (612)

Vehicle repairs



Google Cloud

Cymbal

I found two shops nearby.  
Everything Auto is closest to  
your office, but SJ Mechanic  
has an appointment open  
tomorrow at 9am. Which  
would you like me to book?

Arun

Let's do SJ Mechanic

Cymbal

OK, great. I can book that for  
you if you'd like and inform  
them of the error and what  
steps we've already tried.  
Anything else we should take  
care of before I request your  
appointment?

Cymbal

No problem! I'll call the repair  
shop. I'll explain the situation,  
what other services you need,  
and book you a ride to your  
office.

# Let's Discuss: Automotive

- In that video, what stood out to you?
  - Proactive outreach, chat interaction, access to personalized data, voice-to-text, integration with other apps
- For those who work in the industry...
  - To what extent have you seen conversations or projects aimed at addressing these problems
  - How valuable would a system like this be?
- For those working in other industries...
  - How could you apply this pattern in your domain?











X



ons Help



D	E	F
nt Allocation	*Form 10-K Search* What are the company's commitments to reduce greenhouse gas emissions?	
3%		
8%		
5%		
6%		
7%		
3%		
7%		
3%		
5%		
2%		
3%		



1 1 2 3 4 5 6 7



Summary



Outline

*Headings you add to the document will appear here.*

 Help me write

Create

# Let's Discuss: Financial Services

- In that video, what stood out to you?
  - Natural language search, answers instead of links, summaries, citations
- For those who work in the industry...
  - To what extent have you seen conversations or projects aimed at addressing these problems
  - How valuable would a system like this be?
- For those working in other industries...
  - How could you apply this pattern in your domain?



# How can AI help solve critical productivity blockers for developers?

**30%**

of developers spend 30 minutes/day searching for solutions.

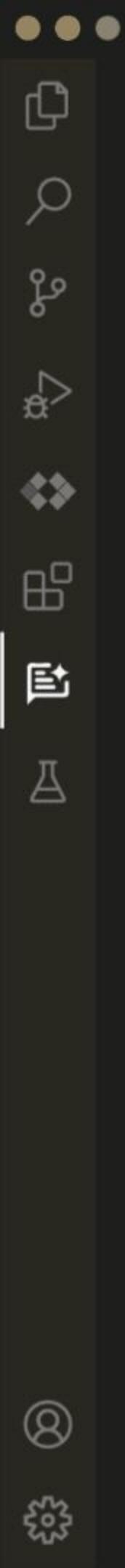
**10%**

spend more than an hour each day.

**30%**

encounter a knowledge silo at least once a week.

Source: [Stack Overflow Developer Survey 2022](#)



main.py

app.py

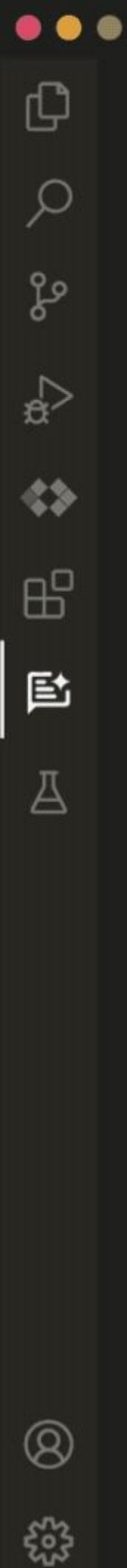
src > frontend > app.py

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38

Sure, here is a full schema DDL for your system using Cloud SQL for PostgreSQL, with performance in mind:

```
product_name varchar(200) NOT NULL,  
product_description varchar(255) NOT NULL,  
product_price decimal(10,2) NOT NULL  
);  
  
CREATE INDEX products_product_name_idx ON products (product_name);  
  
CREATE TABLE order_items (  
order_item_id serial PRIMARY KEY,  
order_id integer NOT NULL REFERENCES orders (order_id),  
product_id integer NOT NULL REFERENCES products (product_id),  
quantity integer NOT NULL,  
unit_price decimal(10,2) NOT NULL  
);  
  
CREATE INDEX order_items_order_id_idx ON order_items (order_id);  
CREATE INDEX order_items_product_id_idx ON order_items (product_id);
```

```
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38
```



main.py

app.py

src > frontend > app.py

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
```

DUET AI: CHAT

Sure, here is an example of how to create a FastAPI service that has an endpoint to return the orders of a customer, based on customer ID:

```
from fastapi import HTTPException, FastAPI
import uvicorn

app = FastAPI()

# Define the orders endpoint
@app.get("/orders/{customer_id}")
async def get_orders(customer_id: int):
    # Check if the customer ID is valid
    if int(customer_id) < 1:
        raise HTTPException(status_code=400, detail="Invalid customer ID")

    # Get the orders for the customer
    orders = get_orders_for_customer(customer_id)

    # Return the orders
    return orders

def get_orders_for_customer(customer_id):
    return {}

if __name__ == "__main__":
    uvicorn.run(app, host="0.0.0.0", port=5000)
```

app.py

src > frontend > app.py

```
1  from fastapi import HTTPException, FastAPI
2  import uvicorn
3
4
5  app = FastAPI()
6
7
8  # Define the orders endpoint
9  @app.get("/orders/{customer_id}")
10 async def get_orders(customer_id: int):
11     # Check if the customer ID is valid
12     if int(customer_id) < 1:
13         raise HTTPException(status_code=400, detail="Invalid customer ID")
14
15     # Get the orders for the customer
16     orders = get_orders_for_customer(customer_id)
17
18     # Return the orders
19     return orders
20
21
22
23
24 def get_orders_for_customer(customer_id):
25     return {}
26
27
28 if __name__ == "__main__":
29     uvicorn.run(app, host="0.0.0.0", port=5000)
30
31
32
33
34
35
36
37
38
```

I hope this helps! Let me know if you have any other questions.

The `from fastapi import HTTPException, FastAPI` statement imports the `HTTPException` and `FastAPI` classes from the `fastapi` module. The `HTTPException` class is used to raise errors in your API, and the `FastAPI` class is used to create a `FastAPI` application. The `app = FastAPI()` statement creates a `FastAPI` object called `app`.

The `@app.get("/orders/{customer_id}")` decorator tells `FastAPI` that the `get_orders()` function is an HTTP GET endpoint that is located at the `/orders` path and takes a customer ID as a parameter.

The `def get_orders(customer_id: int)` statement defines the `get_orders()` function. The `customer_id` parameter is an integer that represents the customer ID. The `if int(customer_id) < 1:` statement checks if the customer ID is valid. If it is not, the `raise` statement raises an `HTTPException` with a status code of 400 (Bad Request).

The `orders = get_orders_for_customer(customer_id)` statement gets the orders for the given customer ID.

The `return orders` statement returns the orders to the caller.

The `def get_orders_for_customer(customer_id):` statement defines the `get_orders_for_customer()` function. The `customer_id` parameter is an integer that represents the customer ID.

The `return {}` statement returns an empty dictionary.

The `if __name__ == "__main__":` statement checks if the code is being run as the main script. If it is, the `uvicorn.run()` function is called to start the server. The `app` argument specifies the `FastAPI` instance that was created earlier. The `host="0.0.0.0"` argument specifies that the development server should listen on all network interfaces. The `port=5000` argument specifies that the development server should listen on port 5000.

src > frontend > app.py

```
1  from fastapi import HTTPException, FastAPI
2  import uvicorn
3
4
5  app = FastAPI()
6
7
8  # Define the orders endpoint
9  @app.get("/orders/{customer_id}")
10 def get_orders(customer_id: int):
11     # Check if the customer ID is valid
12     if int(customer_id) < 1:
13         raise HTTPException(status_code=400, detail="Invalid customer ID")
14
15
16     # Get the orders for the customer
17     orders = get_orders_for_customer(customer_id)
18
19
20     # Return the orders
21     return orders
22
23
24 def get_orders_for_customer(customer_id):
25     return {}
26
27
28 if __name__ == "__main__":
29     uvicorn.run(app, host="0.0.0.0", port=5000)
30
31
32
33
34
35
36
37
38
```

## app.py

```
src > frontend > app.py

21 # Define the orders endpoint
22 @app.get("/orders/{customer_id}")
23 async def get_orders(customer_id: int):
24     # Check if the customer ID is valid
25     if int(customer_id) < 1:
26         raise HTTPException(status_code=400, detail="Invalid customer ID")
27
28     # Get the orders for the customer
29     orders = get_orders_for_customer(customer_id)
30
31     #check order status from provider
32     for order in orders:
33         update_order_status(order)
34
35     # Return the orders
36     return orders
37
38
39
40     def get_orders_for_customer(customer_id):
41         connection = engine.connect()
42         my_query = text(f"SELECT * FROM orders WHERE customer_id = {customer_id};")
43         results_dict = [dict(row) for row in connection.execute(my_query).mappings().all()]
44         connection.close()
45         # Return the orders.
46         return results_dict
47
48     def update_order_status(order):
49         order_status = get_order_status_api(order)
50         order['status'] = order_status
51
52     if __name__ == "__main__":
53         # Establish a connection to the database.
54         uvicorn.run(app, host="0.0.0.0", port=5000)
55
56
57
58
```

I hope this helps! Let me know if you have any other questions.

The `from fastapi import HTTPException, FastAPI` statement imports the `HTTPException` and `FastAPI` classes from the `fastapi` module. The `HTTPException` class is used to raise errors in your API, and the `FastAPI` class is used to create a `FastAPI` application. The `app = FastAPI()` statement creates a `FastAPI` object called `app`.

The `@app.get("/orders/{customer_id}")` decorator tells `FastAPI` that the `get_orders()` function is an HTTP GET endpoint that

app.py • test\_app.py •

```
src > frontend > app.py
27     Args:
28         customer_id (int): The ID of the customer to get orders for.
29
30     Returns:
31         A list of orders for the specified customer.
32
33     Raises:
34         HTTPException: If the customer ID is invalid.
35         """
36
37         # Check if the customer ID is valid
38         if int(customer_id) < 1:
39             raise HTTPException(status_code=400, detail="Invalid customer ID")
40
41         # Get the orders for the customer
42         orders = get_orders_for_customer(customer_id)
43
44         #check order status from provider
45         for order in orders:
46             update_order_status(order)
47
48         # Return the orders
49         return orders
50
51     def get_orders_for_customer(customer_id):
52         """
53         Get orders for the specified customer from a database.
54
55         Args:
56             customer_id (int): The ID of the customer to get orders for.
57
58         Returns:
59             A list of orders for the specified customer.
60         """
61
62         connection = engine.connect()
63         my_query = text(f"SELECT * FROM orders WHERE customer_id = {customer_id};")
64         results_dict = [dict(row) for row in connection.execute(my_query).mappings().all()]
65         connection.close()
```

I found an issue here:

```
def get_orders_for_customer(customer_id):
    connection = engine.connect()
    my_query = text(f"SELECT * FROM orders WHERE
customer_id = {customer_id};")
    results_dict = [dict(row) for row in
connection.execute(my_query).mappings().all()]

    connection.close()
    # Return the orders.
    return results_dict
```

Here is a security fix that you can make to the code:

```
import sqlalchemy

def get_orders_for_customer(customer_id):
    # Establish a connection to the database.
    connection = engine.connect()

    # Create a prepared statement.
    my_query = sqlalchemy.sql.text("SELECT * FROM orders WHERE
customer_id = :customer_id")

    # Bind the customer ID to the prepared statement.
    my_query = my_query.bindparams(customer_id=customer_id)

    # Execute the prepared statement.
    results = connection.execute(my_query)

    # Get the results as a list of dictionaries.
    results_dict = [dict(row) for row in results]

    # Close the connection.
    connection.close()

    # Return the orders.
    return results_dict
```

app.py

test\_app.py

```
# Close the connection.
connection.close()

# Return the orders.
return results_dict
```

Please provide a terraform script from Cymbal's code base for cloudrun deployments

Here is an example of an internal terraform module for deploying on cloudrun:

```
module "cloud_run" {
  source  = "Cymbal/google-cloud-run"
  version = "~> 0.2.0"

  # Required variables
  service_name      = "<SERVICE NAME>"
  project_id        = "<PROJECT ID>"
  location          = "<LOCATION>"
  image              =
  "gcr.io/cloudrun/orders_management_service:v1"
}
```

```
src > frontend > app.py

27     Args:
28         customer_id (int): The ID of the customer to get orders for.
29
30     Returns:
31         A list of orders for the specified customer.
32
33     Raises:
34         HTTPException: If the customer ID is invalid.
35         """
36
37     # Check if the customer ID is valid
38     if int(customer_id) < 1:
39         raise HTTPException(status_code=400, detail="Invalid customer ID")
40
41     # Get the orders for the customer
42     orders = get_orders_for_customer(customer_id)
43
44     #check order status from provider
45     for order in orders:
46         update_order_status(order)
47
48     # Return the orders
49     return orders
50
51 def get_orders_for_customer(customer_id):
52     """
53     Get orders for the specified customer from a database.
54
55     Args:
56         customer_id (int): The ID of the customer to get orders for.
57
58     Returns:
59         A list of orders for the specified customer.
60         """
61
62     connection = engine.connect()
63     my_query = text(f"SELECT * FROM orders WHERE customer_id = {customer_id};")
64     results_dict = [dict(row) for row in connection.execute(my_query).mappings().all()]
65     connection.close()
```

app.py

test\_app.py

```
orders = await asyncio.gather(*(update_order_status(order) for
order in orders))

async def update_order_status(order):
    order_status = await get_order_status_api(order)
    order['status'] = order_status
```

Amazing! Can you add docstrings to all functions in my file?

Sure, here are your functions with docstrings:

Can you write a unit tests for `get_orders()`?  
please patch the call to  
`app.get_orders_for_customer` and mock the  
response. You can use the `@patch` decorator.  
Also make sure that you consider that my  
method has `async` call and awaits for the  
response.

```
src>frontend>test_app.py.py
1 import unittest
2 from unittest.mock import patch
3 from app import get_orders
4
5 class TestGetOrders(unittest.IsolatedAsyncioTestCase):
6
7     @patch('app.get_orders_for_customer', autospec=True)
8     async def test_get_orders_with_valid_customer_id(self, get_orders_for_customer_mock):
9         # Create a mock customer ID
10        customer_id = 12
11        response = [{'order_id': 1, 'value': 320, 'status': 'completed'}]
12        get_orders_for_customer_mock.return_value = response
13
14        # Call the `get_orders` function
15        orders = await get_orders(customer_id)
16
17
18        # Assert that the `get_orders_for_customer_mock` is called once with the give customer_id
19        get_orders_for_customer_mock.assert_called_once_with(customer_id)
20
21        # Assert that the `orders` list contains the expected values
22        self.assertEqual(orders, response)
23
24    async def test_get_orders_with_invalid_customer_id(self):
25        # Create an invalid customer ID
26        customer_id = "abc"
27
28        # Call the `get_orders` function
29        with self.assertRaises(ValueError):
30            await get_orders(customer_id)
31
32
33
34
35
36
37
38
```

app.py • test\_app.py •

```
src > frontend > app.py
27     Args:
28         customer_id (int): The ID of the customer to get orders for.
29
30     Returns:
31         A list of orders for the specified customer.
32
33     Raises:
34         HTTPException: If the customer ID is invalid.
35         """
36         # Check if the customer ID is valid
37         if int(customer_id) < 1:
38             raise HTTPException(status_code=400, detail="Invalid customer ID")
39
40         # Get the orders for the customer
41         orders = get_orders_for_customer(customer_id)
42
43         #check order status from provider
44         for order in orders:
45             update_order_status(order)
46
47         # Return the orders
48         return orders
49
50
51     def get_orders_for_customer(customer_id):
52         """
53             Get orders for the specified customer from a database.
54
55         Args:
56             customer_id (int): The ID of the customer to get orders for.
57
58         Returns:
59             A list of orders for the specified customer.
60         """
61
62         connection = engine.connect()
63         my_query = text(f"SELECT * FROM orders WHERE customer_id = {customer_id};")
64         results_dict = [dict(row) for row in connection.execute(my_query).mappings().all()]
65         connection.close()
```

I found an issue here:

```
def get_orders_for_customer(customer_id):
    connection = engine.connect()
    my_query = text(f"SELECT * FROM orders WHERE
customer_id = {customer_id};")
    results_dict = [dict(row) for row in
connection.execute(my_query).mappings().all()]

    connection.close()
    # Return the orders.
    return results_dict
```

Here is a security fix that you can make to the code:

```
import sqlalchemy

def get_orders_for_customer(customer_id):
    # Establish a connection to the database.
    connection = engine.connect()

    # Create a prepared statement.
    my_query = sqlalchemy.sql.text("SELECT * FROM orders WHERE
customer_id = :customer_id")

    # Bind the customer ID to the prepared statement.
    my_query = my_query.bindparams(customer_id=customer_id)

    # Execute the prepared statement.
    results = connection.execute(my_query)

    # Get the results as a list of dictionaries.
    results_dict = [dict(row) for row in results]

    # Close the connection.
    connection.close()

    # Return the orders.
    return results_dict
```

app.py • test\_app.py •

```

src > frontend > app.py

27     Args:
28         customer_id (int): The ID of the customer to get orders for.
29
30     Returns:
31         A list of orders for the specified customer.
32
33     Raises:
34         HTTPException: If the customer ID is invalid.
35         """
36
37     # Check if the customer ID is valid
38     if int(customer_id) < 1:
39         raise HTTPException(status_code=400, detail="Invalid customer ID")
40
41     # Get the orders for the customer
42     orders = get_orders_for_customer(customer_id)
43
44     #check order status from provider
45     for order in orders:
46         update_order_status(order)
47
48     # Return the orders
49     return orders
50
51 def get_orders_for_customer(customer_id):
52     """
53     Get orders for the specified customer from a database.
54
55     Args:
56         customer_id (int): The ID of the customer to get orders for.
57
58     Returns:
59         A list of orders for the specified customer.
60         """
61
62     connection = engine.connect()
63     my_query = text(f"SELECT * FROM orders WHERE customer_id = {customer_id};")
64     results_dict = [dict(row) for row in connection.execute(my_query).mappings().all()]
65     connection.close()

```

DUET AI: CHAT

```

# Close the connection.
connection.close()

# Return the orders.
return results_dict

```

Please provide a terraform script from Cymbal's code base for cloudrun deployments

Here is an example of an internal terraform module for deploying on cloudrun:

```

module "cloud_run" {
  source  = "Cymbal/google-cloud-run"
  version = "~> 0.2.0"

  # Required variables
  service_name      = "<SERVICE NAME>"
  project_id        = "<PROJECT ID>"
  location          = "<LOCATION>"
  image              =
  "gcr.io/cloudrun/orders_management_service:v1"
}

```

# Let's Discuss: Software Development

- In that video, what stood out to you?
- For those who work in the industry...
  - To what extent have you seen conversations or projects aimed at addressing these problems
  - How valuable would a system like this be?
- For those working in other industries...
  - How could you apply this pattern in your domain?

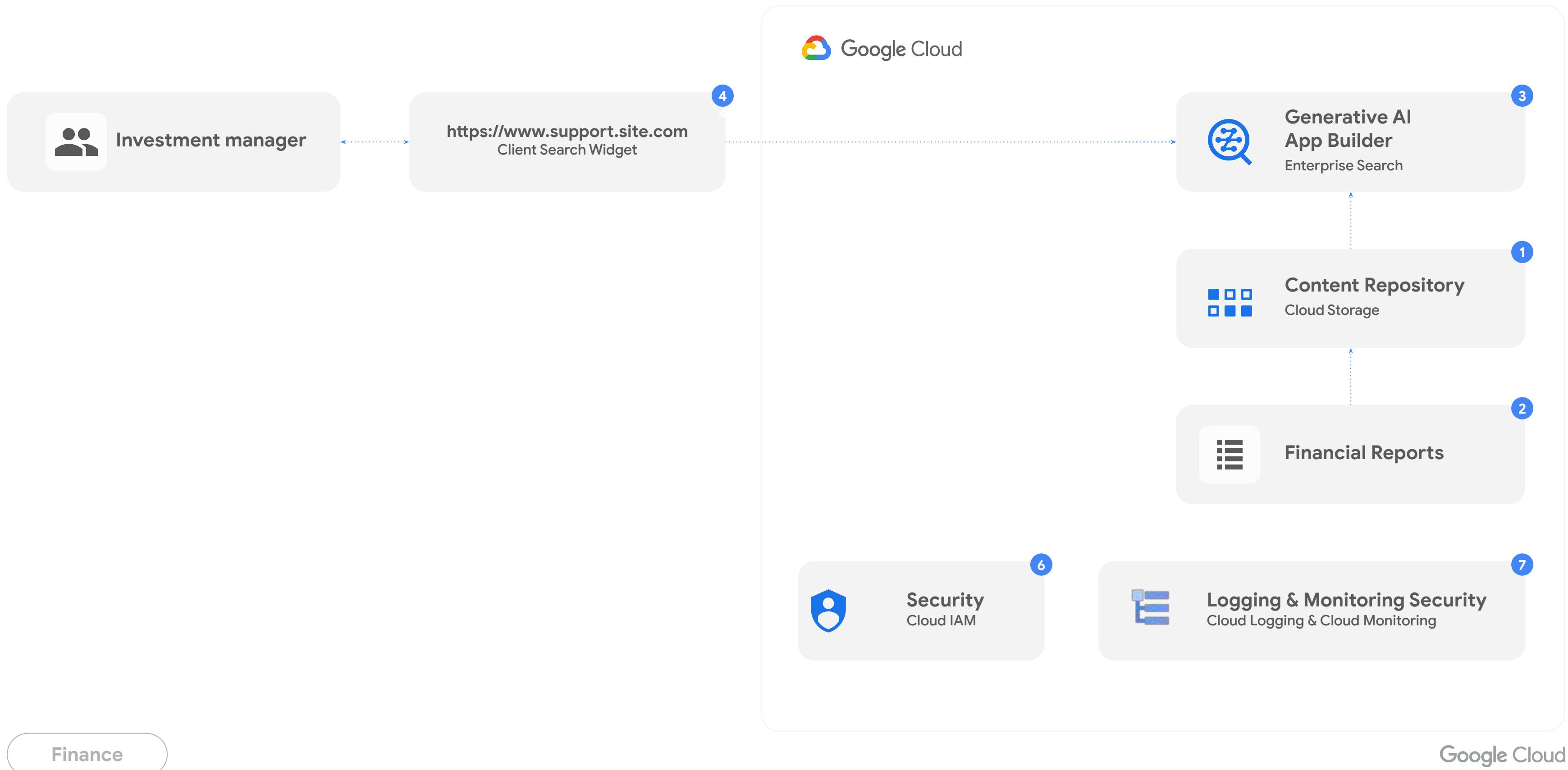


# Topics

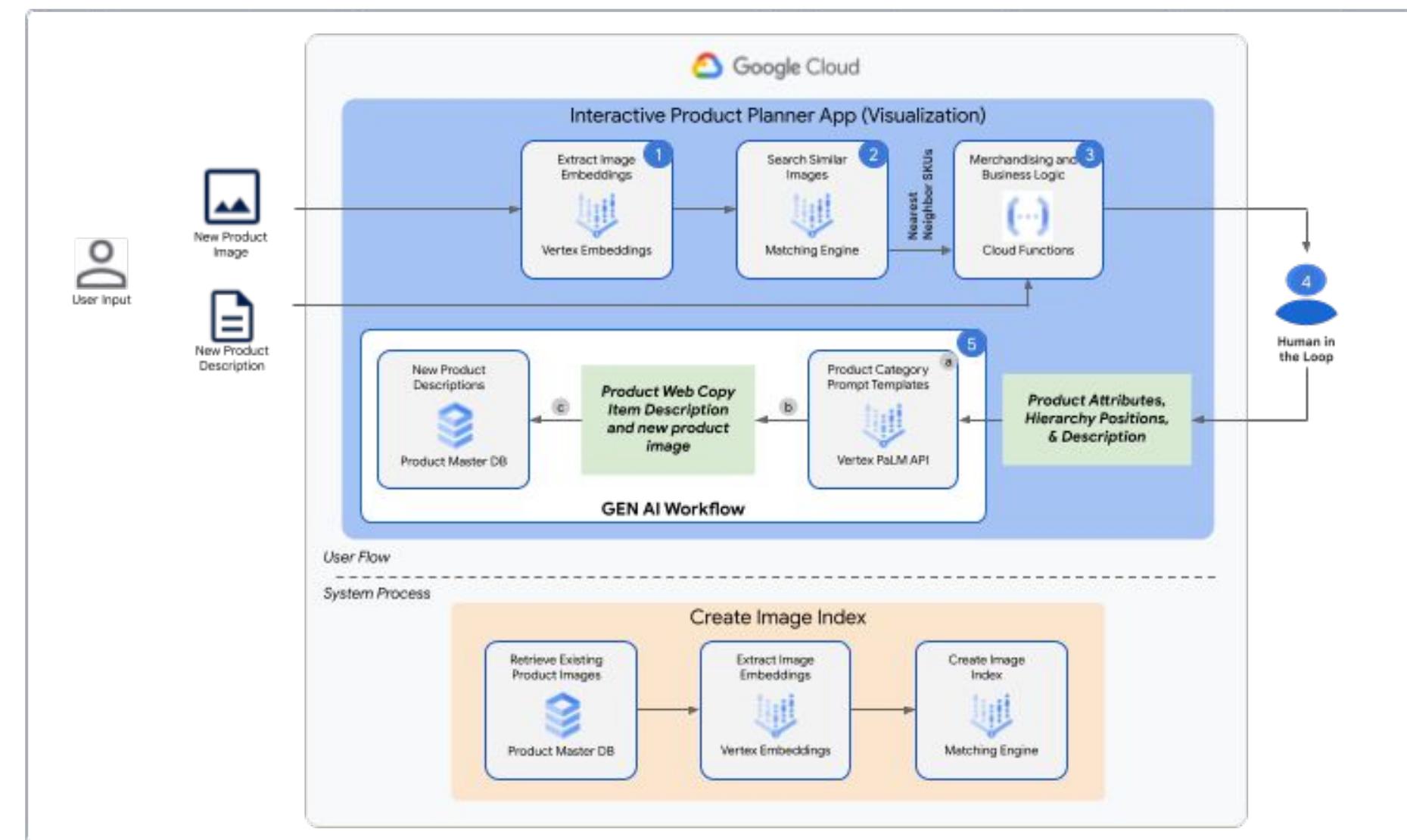
- 01 What is Generative AI?
- 02 Real-world examples
- 03 Example architectures



# Financial Services Research



# Ecommerce Site Development



# In this module, you learned...

01

What Generative AI (GenAI) can do

02

How Generative AI functionality can enable critical workflows across industries

03

How production-grade solutions are built by integrating multiple GenAI products







# Understanding Generative AI

# In this module, you learn ...

01

Fundamental Generative AI concepts

02

Best practices for responsible use of GenAI



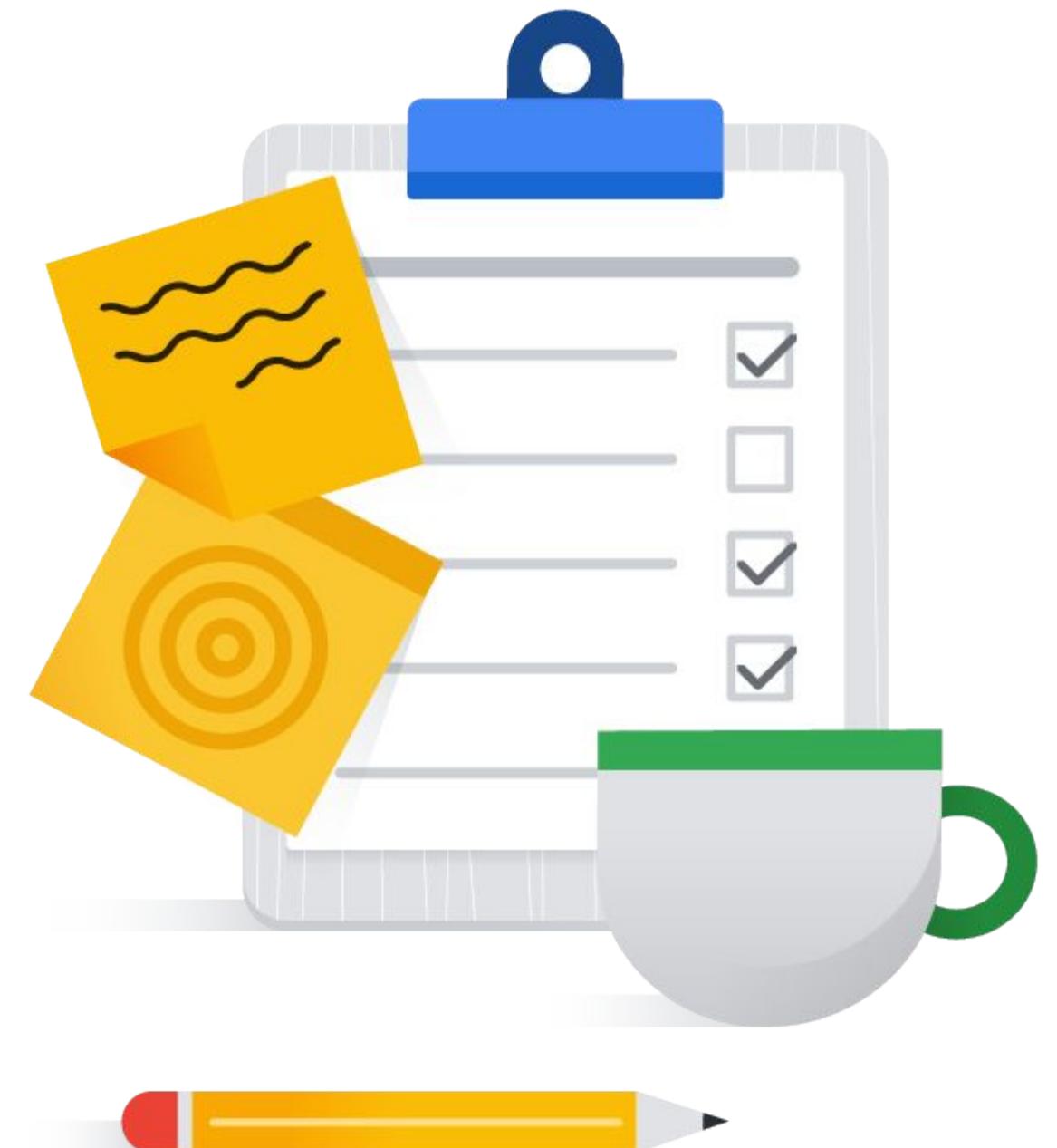
# Topics

01

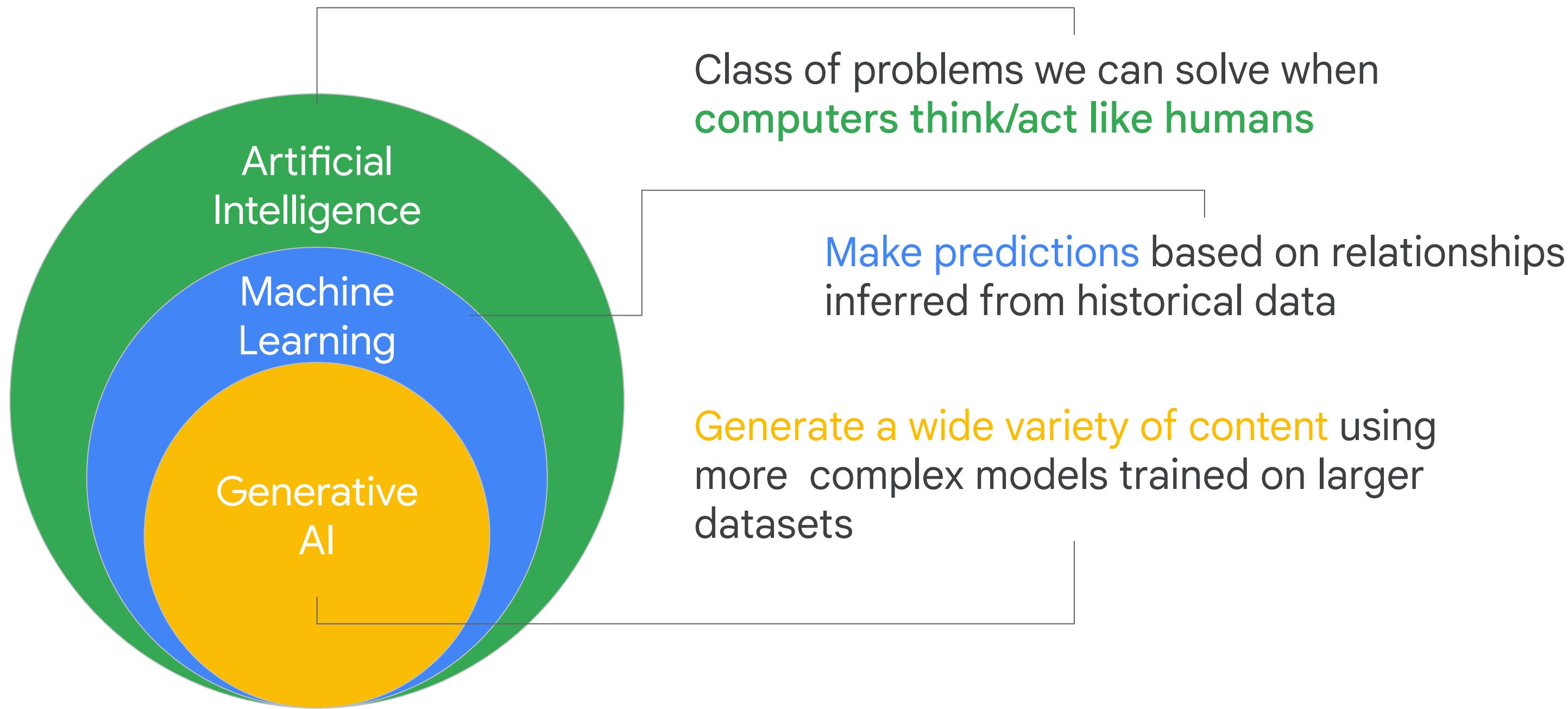
GenAI fundamentals

02

Best practices for responsible GenAI



# Machine Learning is a type of AI, and Generative AI (GenAI) is a type of machine learning



# Large language models are - large!



ML algorithms that can **recognize, predict, and generate** human languages



Pre-trained on petabyte scale text-based datasets resulting in large models with **10s to 100s of billions of parameters**



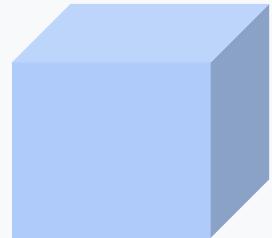
LLMs are normally **pre-trained on a large corpus of text** followed by fine-tuning on a specific task



LLMs can also be called **Large Models** (includes all types of data modality) and **Generative AI** (a model that produces content)

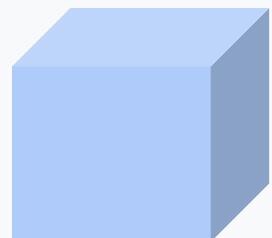


Go read this huuuuuuge pile of books.



So, you've learned about cats and millions of other concepts ... what's a cat?

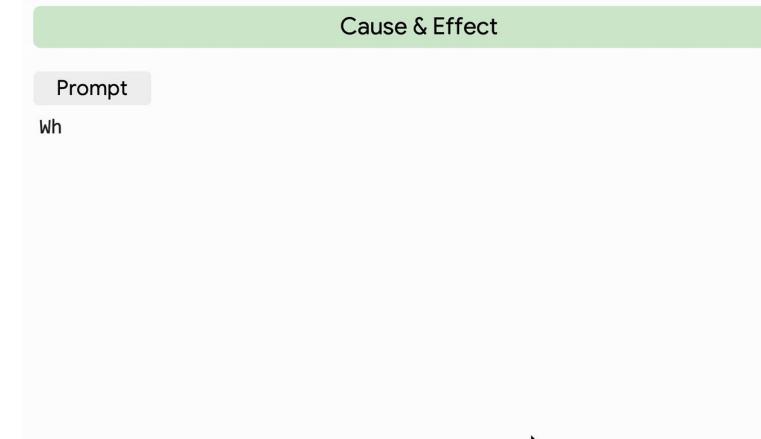
A cat is a small, domesticated carnivorous mammal.



**Generative language models**

LaMDA, PaLM, GPT-3, etc.

# Google has an entire suite of foundation models



## PaLM for text/chat

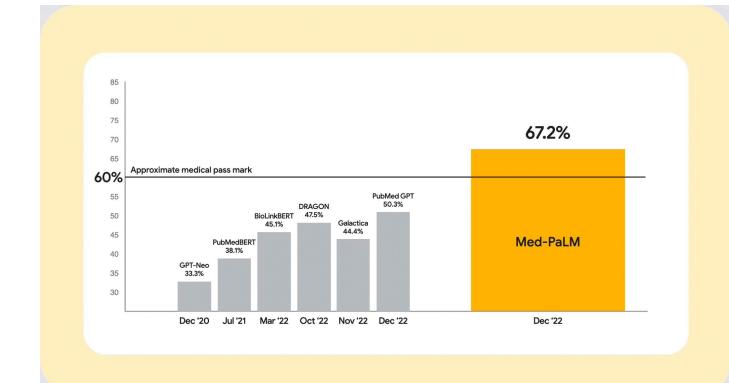
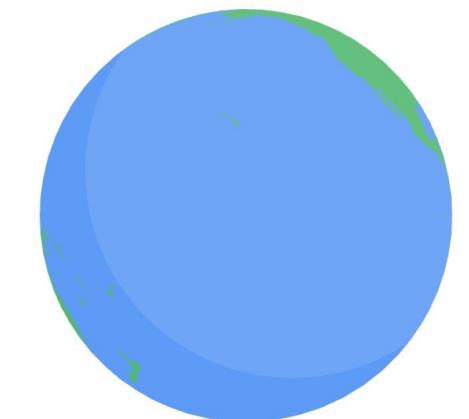
Custom language task & Multi-turn conversations with session context

## Imagen for Text to Image

Create and edit images from simple prompts

## Sec PaLM

Cybersecurity analysis



## Codey for Code Generation

Improve coding and debugging

## Chirp for Speech to Text

Build voice enabled applications

## Med PaLM

Accurately and safely answer medical questions

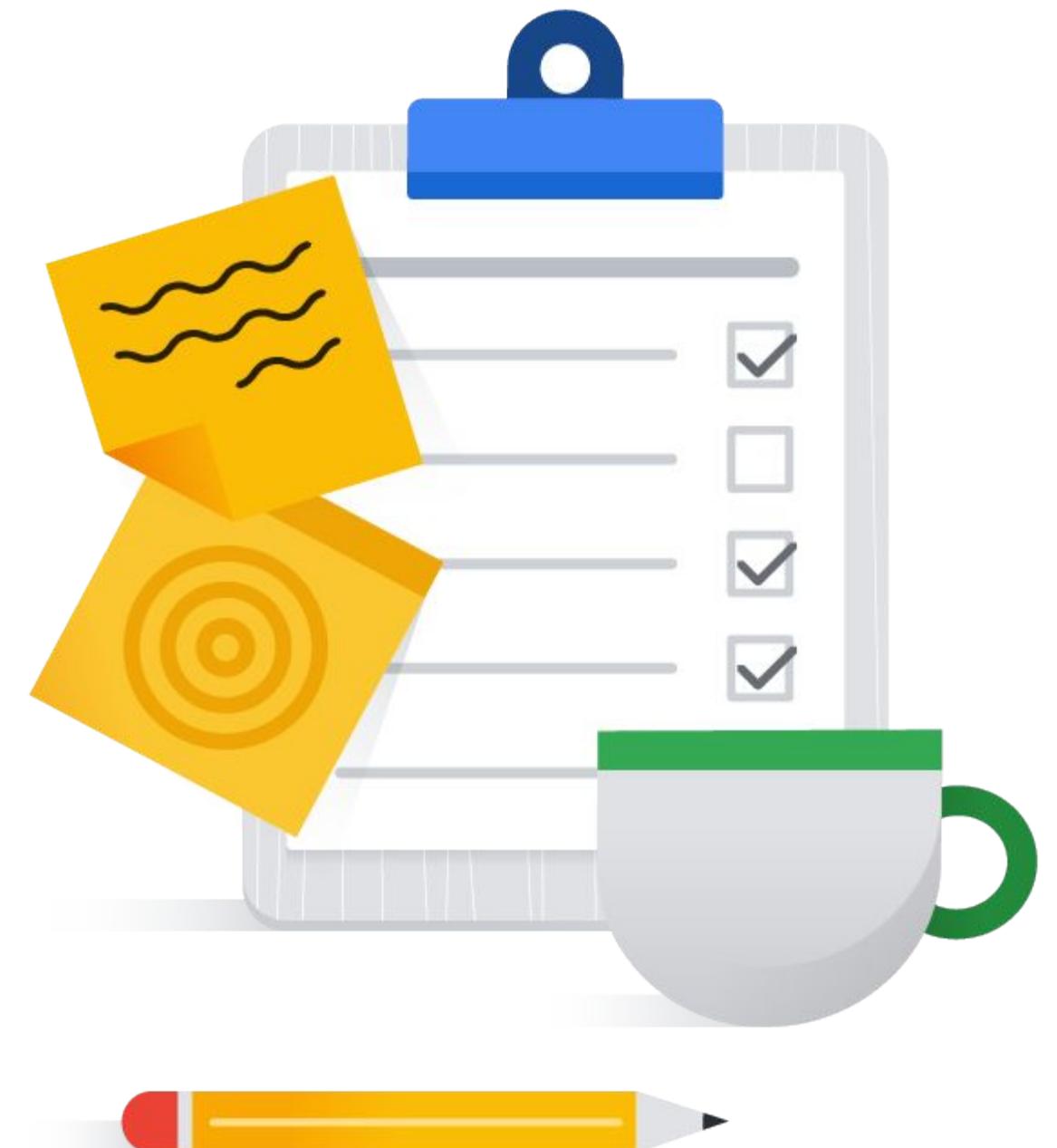
# Topics

01

GenAI fundamentals

02

Best practices for responsible GenAI

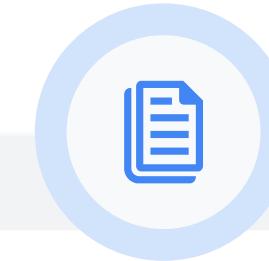


# Google strives to develop AI solutions responsibly



## Principles

1. Be socially beneficial
2. Avoid creating or reinforcing bias
3. Be built and tested for safety
4. Be accountable to people
5. Incorporate privacy design principles
6. Uphold high standards for scientific excellent
7. Be made available for uses that accord with these principles



## Practice

1. Examine your raw data and test (General)
2. Check the system for unfair bias (Fairness)
3. Communicate explanations to users (Interpretability)
4. Collect and handle data responsibly (Privacy)
5. Develop an approach to combat threats (Safety)

# End users also have responsibilities when using GenAI

01

## Avoid hallucinations

Models can generate plausible but incorrect content. You can reduce a model's "creativity" with carefully crafted prompts and setting the temperature of requests

02

## Fact check and test

Users should always check factual assertions provided by GenAI tools. Organizations should have standards and processes for testing results over time.

03

## Use GenAI as a starting point

In some cases, using content generated entirely by AI is inappropriate. In many cases, the results can be improved by human fine-tuning.

04

## Be careful with prompts

Avoid inclusion of sensitive information. Also, avoid using brand or celebrity names in prompts as these might result in answers that are subject to infringement allegations.

05

## Keep records

In situations where you might need to demonstrate the source of your work, keeping track of your prompts, results, and final products can be important.

06

## Disclose AI use

In many cases, it's a good idea to acknowledge when content has been generated by an AI tool.





# Generative AI on Google Cloud

# In this module, you learn...

- 01 Why it's important to use enterprise-ready GenAI tools
- 02 Which Google Cloud products offer GenAI functionality
- 03 Which tools are appropriate for which use cases



# Topics

01

Consumer vs. enterprise products

02

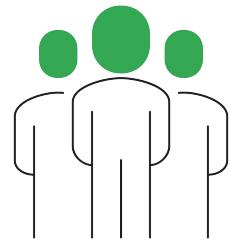
Google Cloud Platform GenAI products

03

Google Workspace GenAI features



# Consumers & enterprises have different needs....



## Consumers and enthusiasts

Bard, MakerSuite

- Help me plan a neighborhood block party
- Outline my blog post about summer mocktail recipes
- I want to write a novel. How do I get started?
- Draft a packing list for my weekend fishing and camping trip
- Give me a list of idioms for “let’s circle back” that aren’t cringe



## Enterprise

Vertex AI, Gen App Builder

- Allow data analysts to search and summarize market reports while **controlling our data**
- Help my customers understand my financial products while being **safe, explainable, and regulatory compliant**
- Is it easy to **integrate into our existing apps and platform?**
- Handle a customer service interaction with **accurate info**
- Can you guarantee we have **access to state-of-the-art models?**
- Can we generate content while **controlling costs?**

# Topics

01

Consumer vs. enterprise products

02

Google Cloud Platform GenAI products

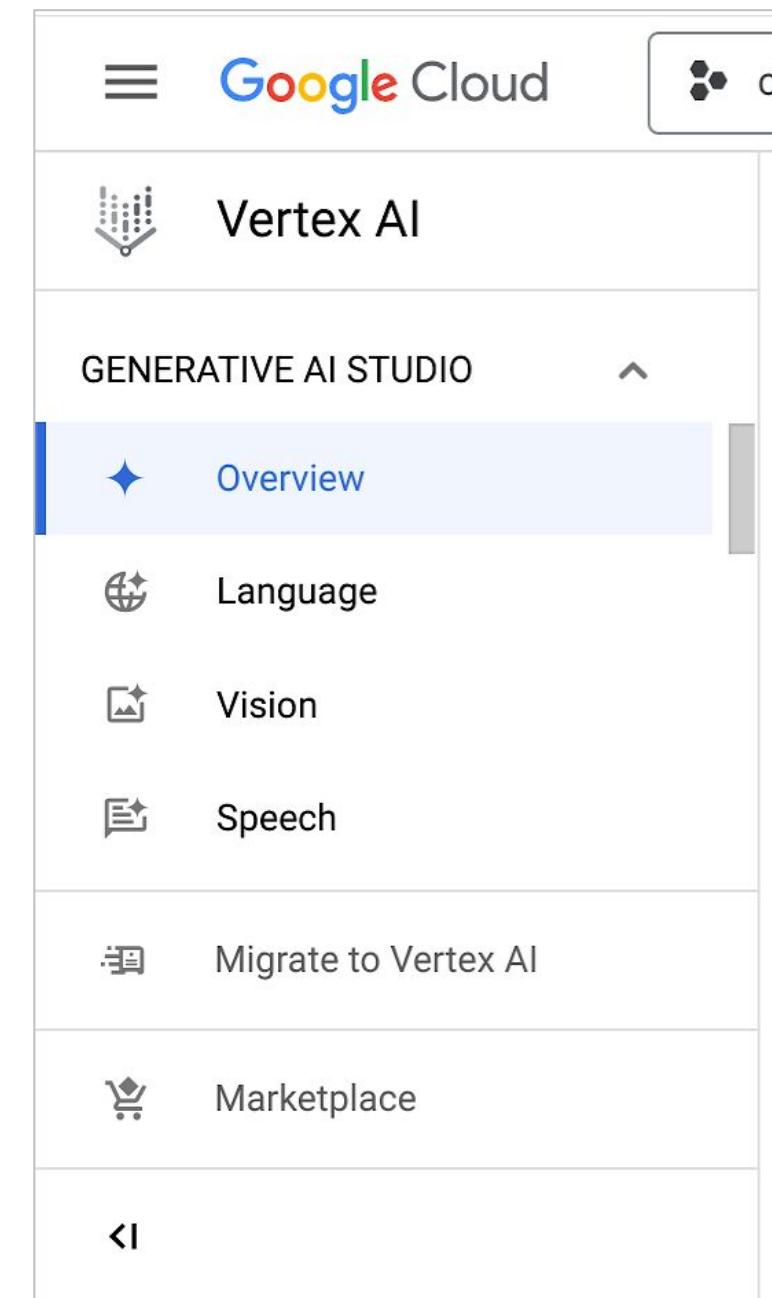
03

Google Workspace GenAI features



# Generative AI Studio simplifies prompting, tuning, and deploying Google foundation models

- Prompt design tools allow you to craft high-performing prompts
- To improve model performance, you can tune the model
- Generative AI studio offers tooling for three types of generation
  - Language
  - Vision
  - Speech



# The prompt UI makes it easy to develop effective prompts

**Context** [?](#)

Multi-choice problem: Define the category of the ticket?  
Categories:  
- Credit card  
- Bank account services  
- Loans and Mortgages

**Examples** [?](#)

INPUT	OUTPUT	Category
I lost my credit card numbered 12345. Can you help with deactivating the card?	Credit card	X
I would like to change the address associated with my account. I have been calling the bank multiple times but couldn't get through. Please help me.	Bank account services	X
good morning my name is xxxx xxxx and i appreciate it if you could help me put a stop to chase bank cardmember services. I wrote to chase asking for debt verification and what they sent me a statement which is not acceptable i am asking the bank to validate the debt instead i been receiving mail every month from them attempting to collect a debt i have a right to know this information as a consumer chase account xxxx xxxx xxxx xxxx thanks in advance for your help	Loans and Mortgages	X
<a href="#">+ Write an input to add a new row</a>	<a href="#">+ Write an output to add a new row</a>	

**Test** [?](#)

INPUT	OUTPUT	Category
my grand son give me check for i deposit it into my chase account after fund clear my chase bank closed my account never paid me my money they said they need to speak with my grand son check was clear money was taking by my chase bank refuse to pay me my money my grand son called chase times they told him i should call not him to verify the check owner he is out the country most the time date happen check number xxxx claim number is xxxx with chase	<a href="#">Model responses will appear here</a>	<a href="#">- Markdown</a>

We want your [feedback](#).

**Model** [text-bison@001](#)

**Temperature** [0.2](#)

**Token limit** [256](#)

**Top-K** [40](#)

**Top-P** [0.8](#)

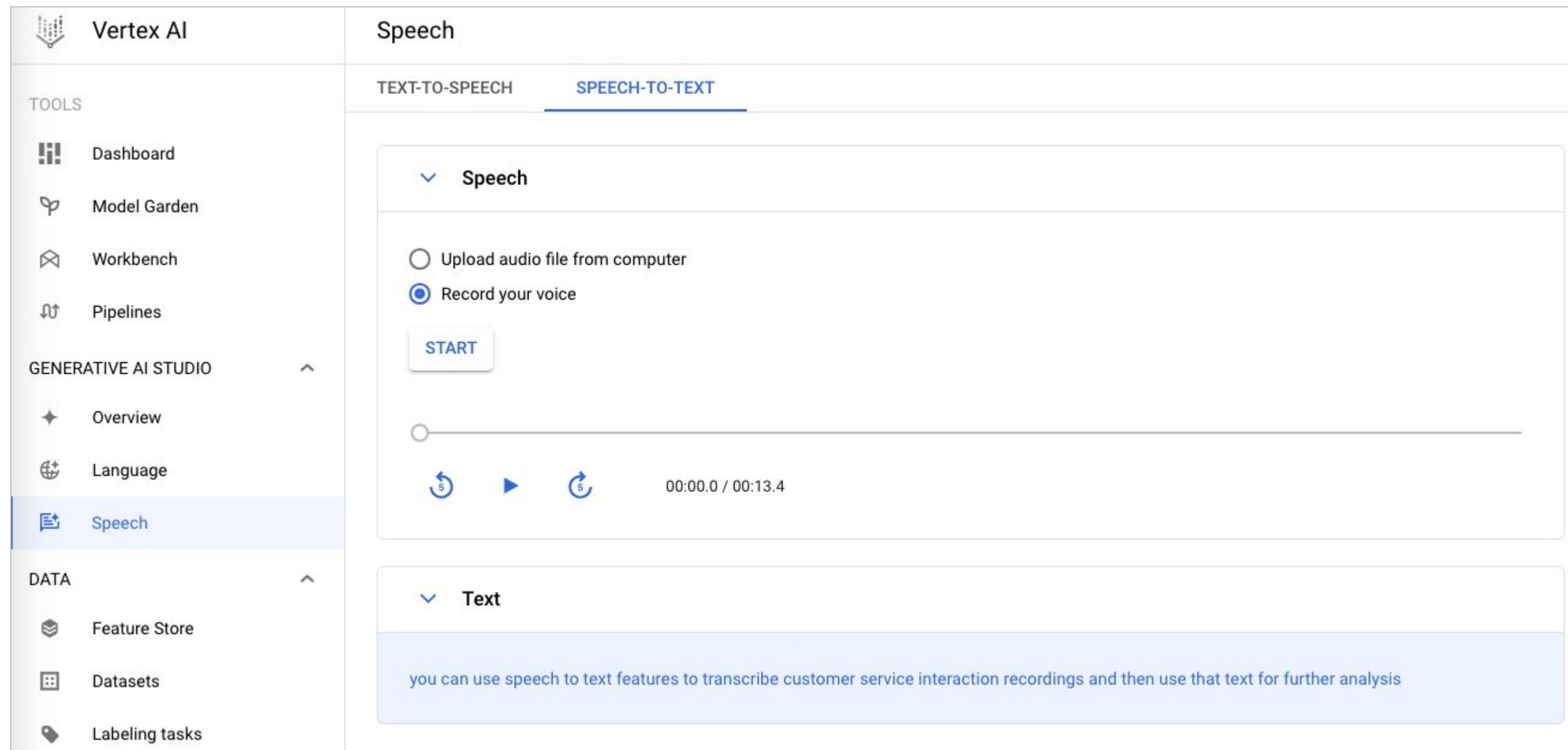
**Safety filter threshold** [Block few](#)

**SUBMIT** **RESET PARAMETERS**

# Vision tools allow you to create images or image-related text content

The screenshot displays the Vertex AI Vision interface. On the left, a sidebar lists various tools: Dashboard, Model Garden, Workbench, Pipelines, Overview, Language, **Vision** (which is selected), Speech, Feature Store, Datasets, Migrate to Vertex AI, and Marketplace. The main area shows a grid of eight generated images. Below the images is a text input field containing "Material design style icon for video conferencing appliance". To the right, there are several configuration sections: "Model options" set to "imagegeneration@001", "Aspect ratio" set to "Square" (with a note that resolution is 1024 by default), "Number of results" set to 8, "Negative prompt" (with a placeholder "Define what you don't want to see"), and "ADVANCED OPTIONS" which includes a checkbox for "Enable fine-tuned model" with a sub-note about customizing the model. At the bottom, there are buttons for GENERATE, EDIT, CAPTION, and VISUAL Q & A, along with a large blue GENERATE button.

# Speech tools allow users to easily convert text to speech, or speech-to-text



# GenAI Studio allows you to train and deploy fine-tuned (customized) models

- Allows you to teach the model more about what your expected output should be
- You specify a custom dataset which includes prompts along with the expected output
- The result of tuning a model is a set of small add-in layers that is used in conjunction with a foundation model
  - The adapters are what enable the foundation model to perform better for certain tasks and domains



**Tune a model**

Tune a model so it's better equipped for your use case, then deploy to an endpoint to get predictions or test it in prompt design. Take a tutorial on [creating a tuned model](#).

**+ CREATE TUNED MODEL**

# Model Garden helps you discover, customize, and deploy models from many sources

The screenshot shows the Vertex AI Model Garden interface. On the left, there's a sidebar with sections like TOOLS (Dashboard, Model Garden, Workbench, Pipelines), GENERATIVE AI STUDIO (Overview, Language, Vision, Speech), DATA (Feature Store, Datasets, Labeling tasks), and MODEL DEVELOPMENT (Training, Experiments). The main area is titled "Model Garden" with "EXPLORER GENERATIVE AI" and "VIEW MY MODELS" buttons. It features a search bar and a modalities section with categories: Language (23), Vision (46), Tabular (2), Documents (2), and Speech (1). Below these are sections for Tasks (Generation, Classification, Detection, Extraction, Recognition, Translation, Embedding, Segmentation) and Features (API available, 9). A central grid displays various models:

Model	Type	Description	Actions
Codey for Code Completion	Foundation, Language	Generates code based on code prompts. Good for code suggestions and minimizing bugs in code.	code-gecko@001, VIEW DETAILS
Codey for Code Generation	Foundation, Language	Generates code based on natural language input. Good for writing functions, classes, unit tests, and more.	code-bison@001, VIEW DETAILS
Codey for Code Chat	Foundation, Language	Get code-related assistance through natural conversation. Good for questions about an API, syntax in a supported language, and more.	codechat-bison@001, VIEW DETAILS
Stable Diffusion v1-5	Foundation, Vision	Latent text-to-image diffusion model capable of generating photo-realistic images given a text input.	runwayml/stable-diffusion-v1-5, VIEW DETAILS
BERT	Foundation, Language	Neural network-based technique for natural language processing. Use it to train your own question answering system and more.	google/bert-base-001, VIEW DETAILS
InstructPix2Pix	Foundation, Language, Vision	Given an input image and a text prompt that tells the model what to do, the instruct-pix2pix model follows the prompt to edit the image by generating a new one.	timbrooks/instruct-pix2pix, VIEW DETAILS

# Gen App Builder let's developers and power users build no-code apps that tap into Google's foundation models



## Enterprise Search

1. Quickly build custom search engines on your proprietary data
2. Deliver natural language understanding and semantic search
3. Automatically handle synonyms & misspellings
4. Include AI-generated summaries and enable conversational search
5. Incorporate search across structured and unstructured data



## Infobot

1. Easily create virtual agents that use GenAI
2. Agents can answer questions and have conversations with users based on information you provide
3. Infobot applications are built on top of Dialogflow CX technologies, but are easy to bootstrap
4. Infobot features are not currently available to the public

# Topics

01

Consumer vs. enterprise products

02

Google Cloud Platform GenAI products

03

Google Workspace GenAI features



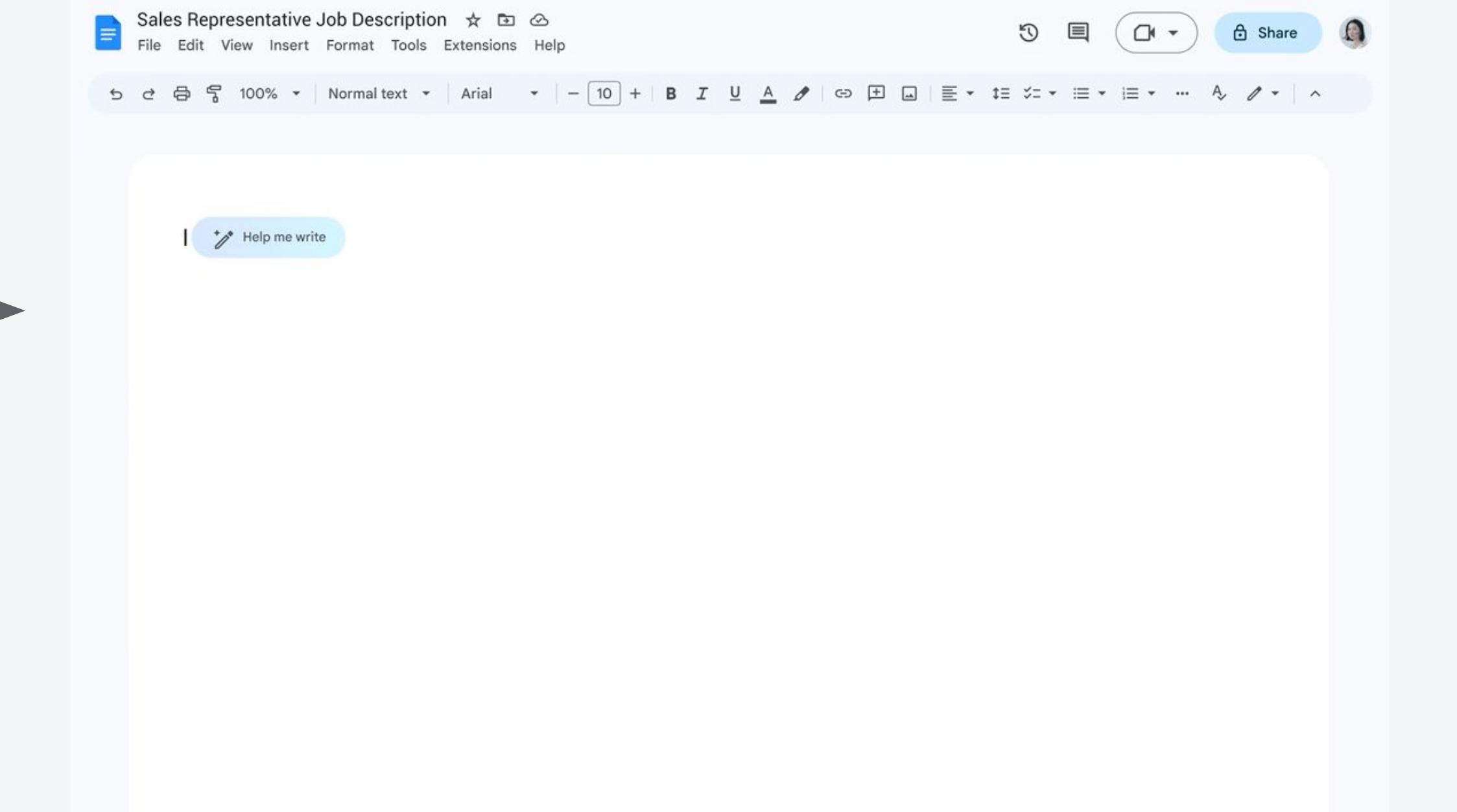
# Google Docs will provide features like “Help me write”



## Docs

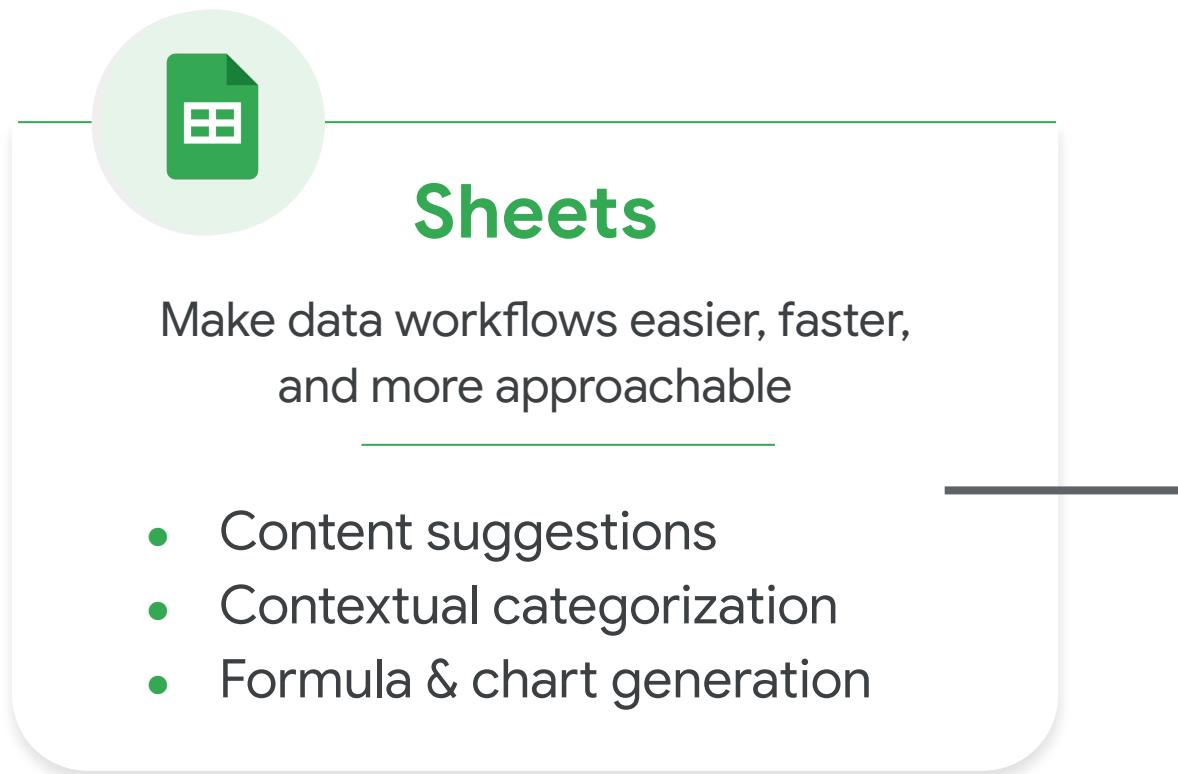
Think bigger, work faster, and supercharge your imagination

- Help write
- Proofread
- Summarize



A screenshot of the Google Docs interface for a document titled "Sales Representative Job Description". The toolbar at the top includes File, Edit, View, Insert, Format, Tools, Extensions, and Help. Below the toolbar are standard document controls for search, print, and zoom (100%). The text color is set to "Normal text" and the font is Arial, size 10. The toolbar also features bold (B), italic (I), underline (U), and strikethrough (A) buttons, along with various alignment and list tools. In the bottom-left corner of the document area, there is a small blue button with a pencil icon and the text "Help me write".

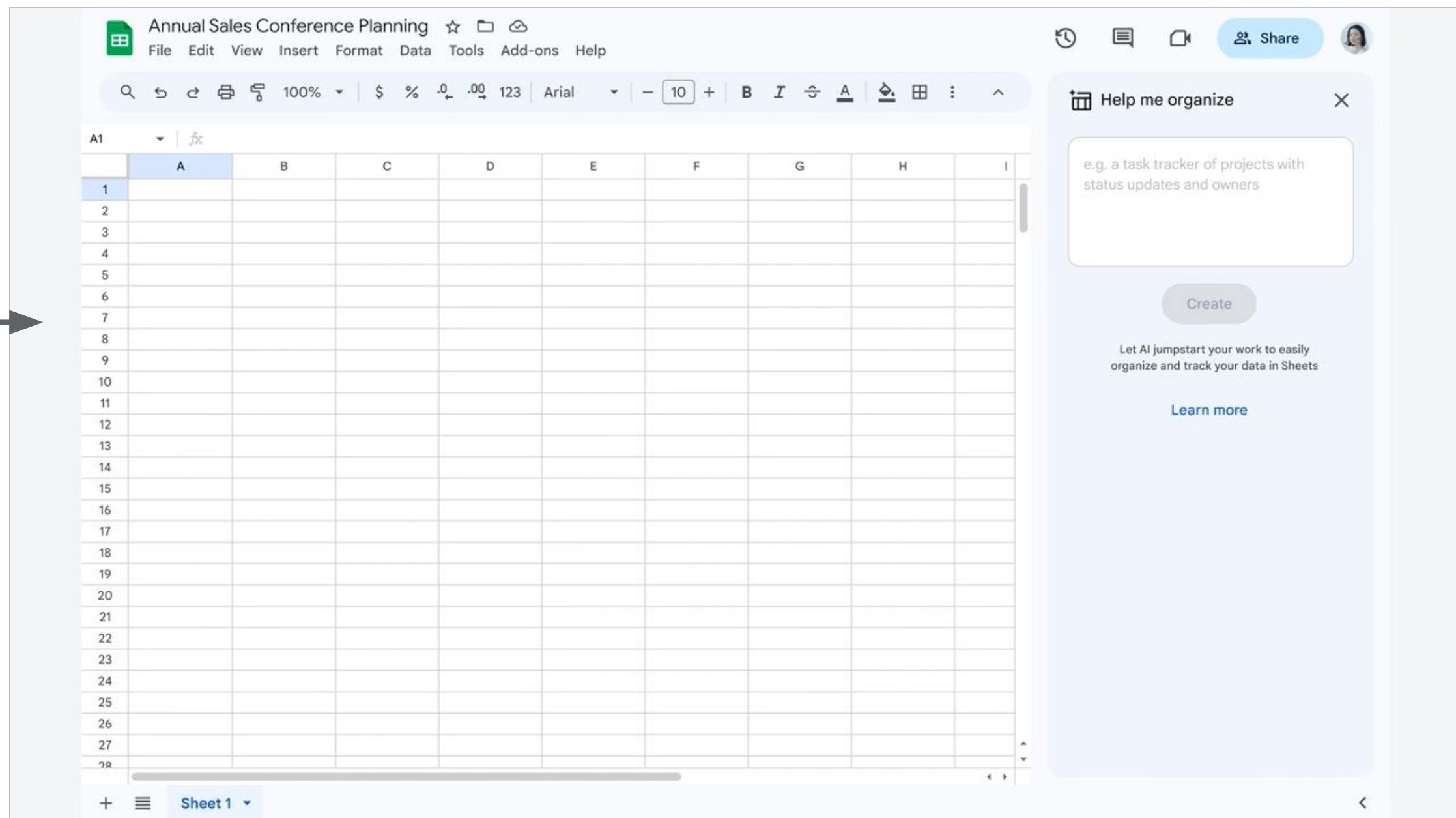
# Google Sheets will provide features like “Help me organize”



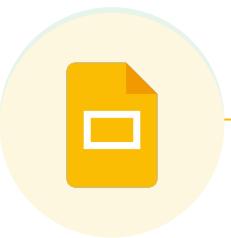
**Sheets**

Make data workflows easier, faster, and more approachable

- Content suggestions
- Contextual categorization
- Formula & chart generation



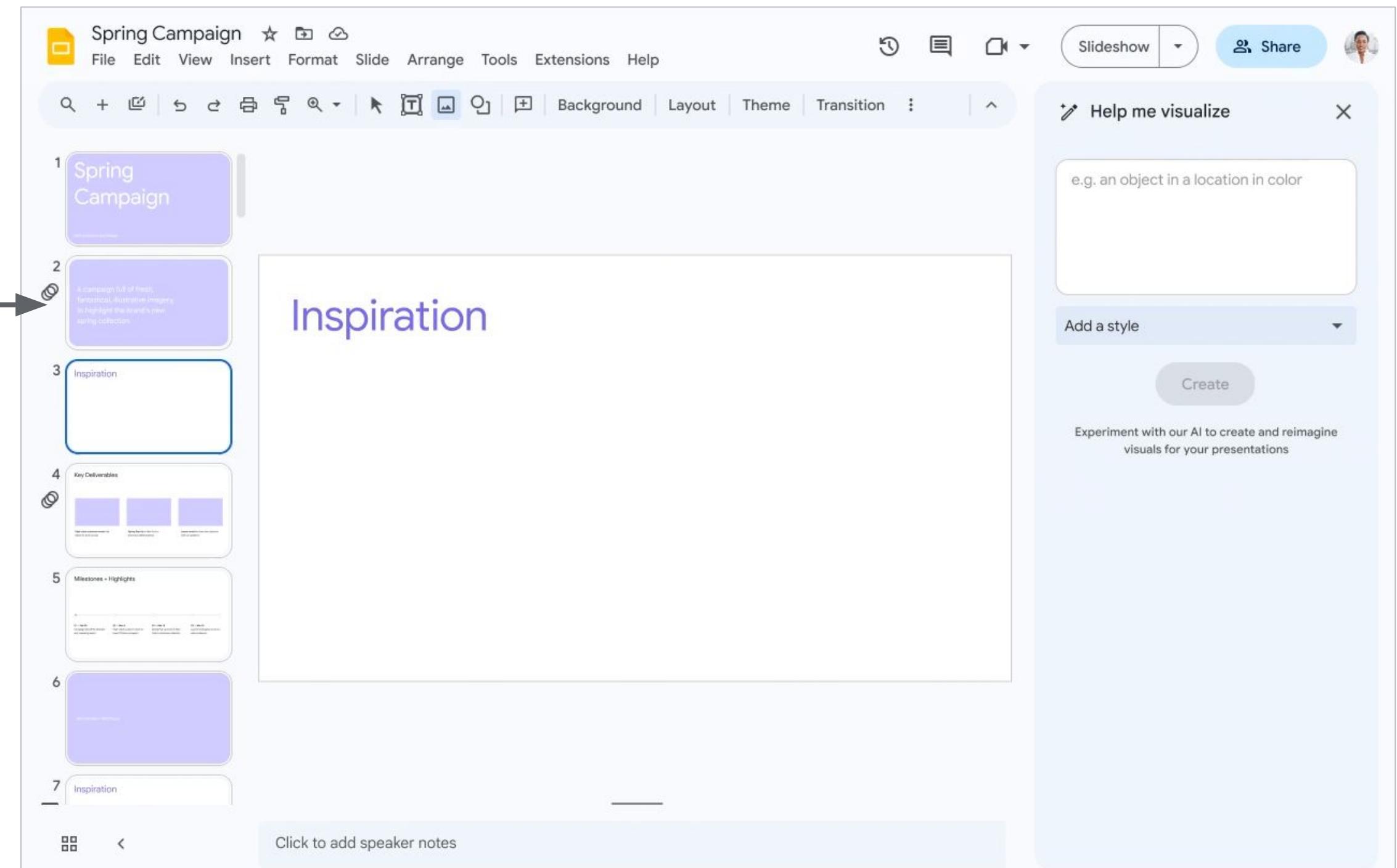
# Google Slides will provide features like “Help me visualize”



## Slides

Realize your creative vision easier and show up like a pro

- Generate images from text
- Rich content libraries
- Generate soundtracks







**GenAI in your  
organization**

# In this module, you learn to ...

01

Begin developing a process for identifying priority use cases across your company



# Topics

01

Finding and prioritizing use cases



# Start with the solution classes discussed earlier



## CREATE

Bring your thoughts and visions to life

### Use cases

- Generate product descriptions from images
- Generate images from text



## SUMMARIZE

Condense and summarize your knowledge base into a simple format

### Use cases

- Content/video summarization
- Intra-knowledge Q&A



## DISCOVER

Help your customers and employees find what they need at the right time

### Use cases

- Search for a document
- Machine-generated event monitoring



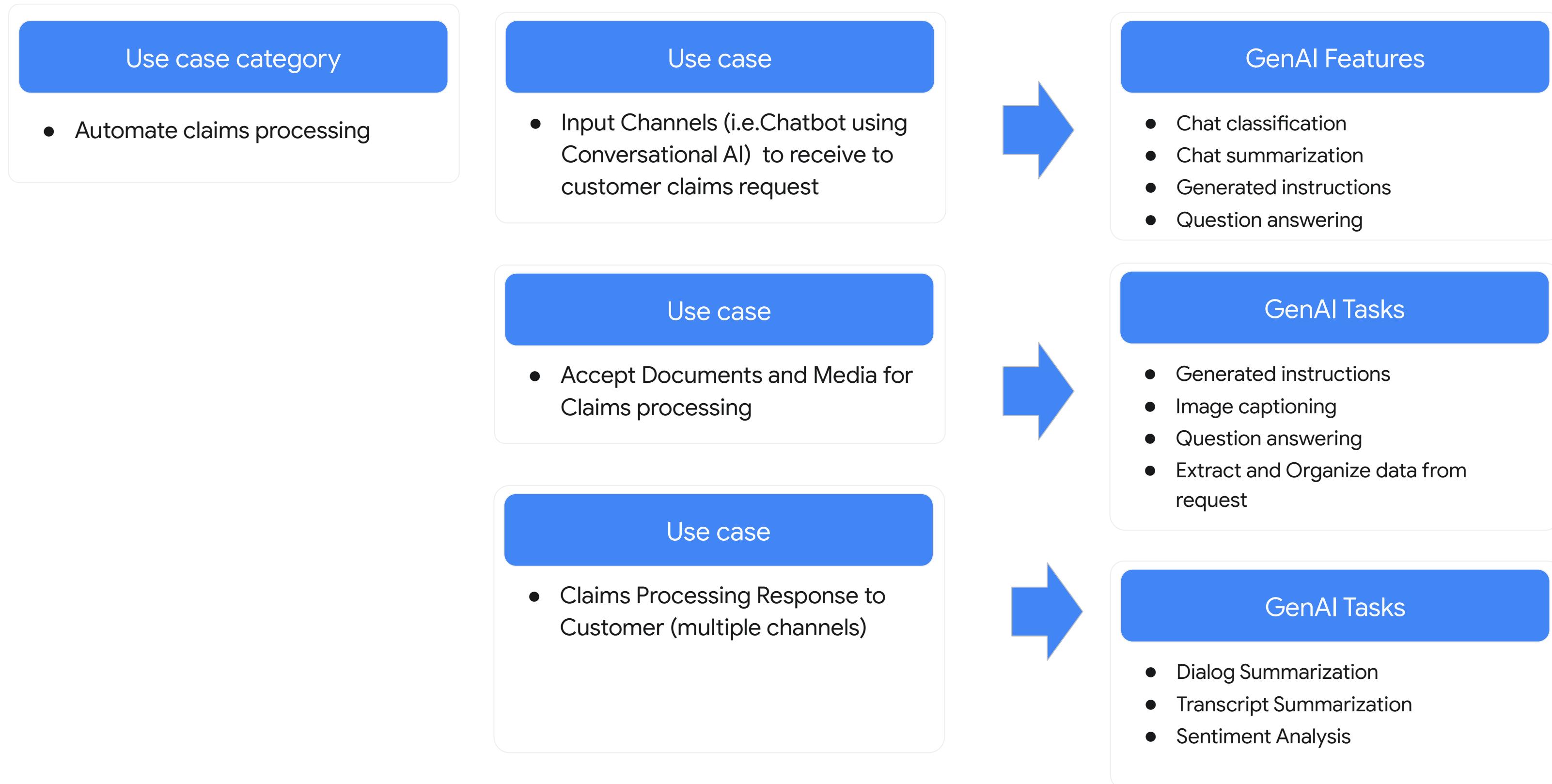
## AUTOMATE

Automate your customer service across multiple channels

### Use cases

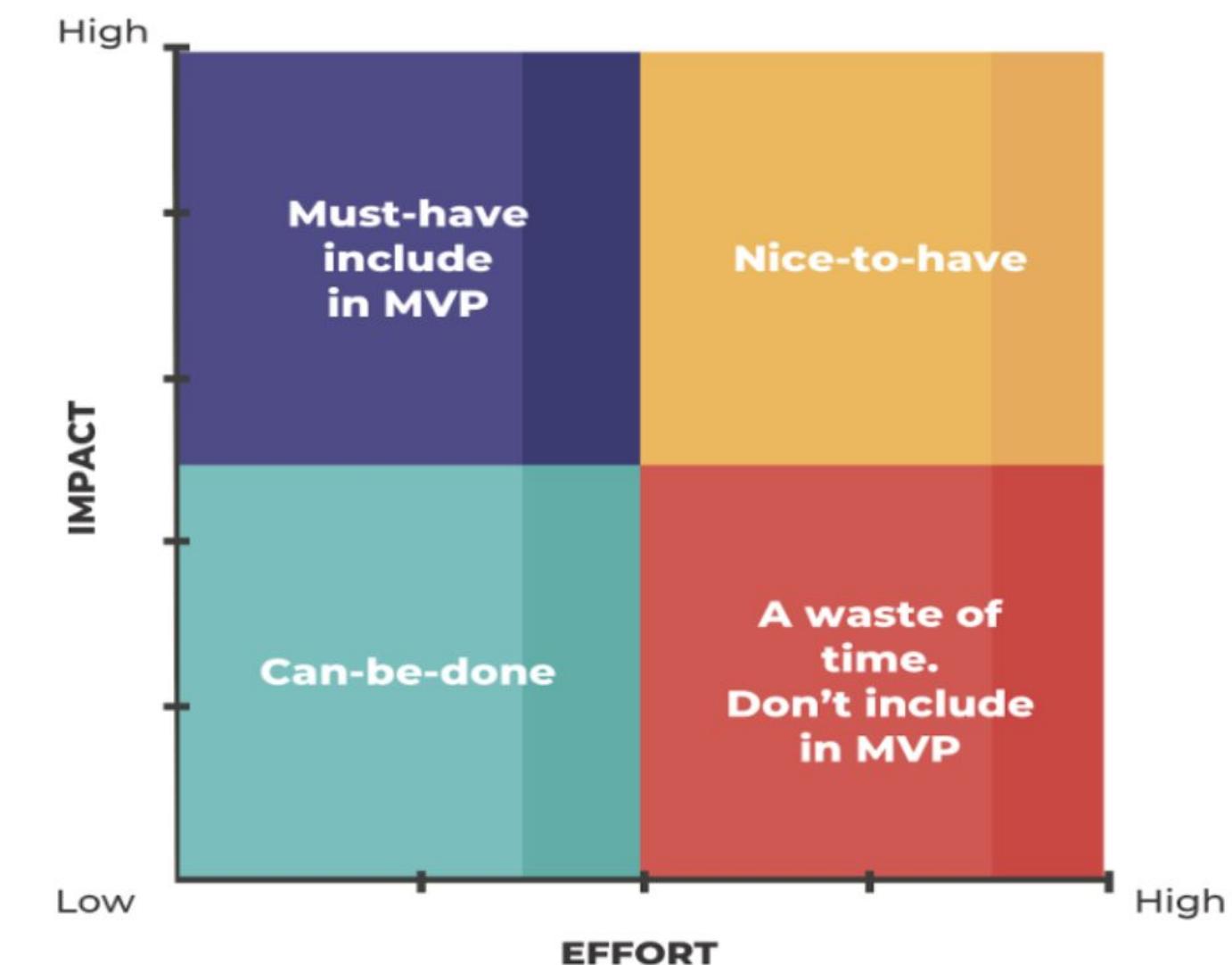
- Contract information extraction
- Feedback classification and ticket creation

# Category > Use case > GenAI features



# Identify priorities based on impact and effort

1. **Business Impact:** What GenAI applications would have the highest impact on a given business area?
  - a. Complex search
  - b. Infobots
  - c. Content generation
2. **Data Availability:** What data would be available?
  - a. Sample prompts and responses
  - b. Unstructured data sources
3. **Complexity:** What are the key challenges? (business/technical requirements, limitations, risks)
4. **Expectation:** How would you define your success criteria?



# A standard worksheet can help with prioritization

Criteria	Description	Score
Organization's interest/ readiness to test (flexibility to test a nascent api)	What is the level of relevance of Google GenAI capabilities to customer use cases. Willingness to engage in Trust Tester Program (TTP)?	
Product availability	Are there detailed APIs, or other GenAI product assets available within the required timeline?	
Product maturity	Does the product have the requisite level of maturity?	
Product depth	What is the level of detail and depth that the product has that is currently available for use?	
Tokens generated	Are there any challenges with token limitations?	
Input/Output token comparison / ratio	What is the ratio of tokens used for input prompts and output response? Any challenges or impact on prompt design, prompt tuning, quality of response?	
Customization requirements	Can we solve this with OOTB components, foundational models as is? Or requires detailed customization?	

# In this module, you learned to ...

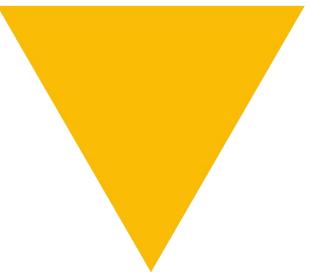
01

Begin developing a process for identifying priority use cases across your company





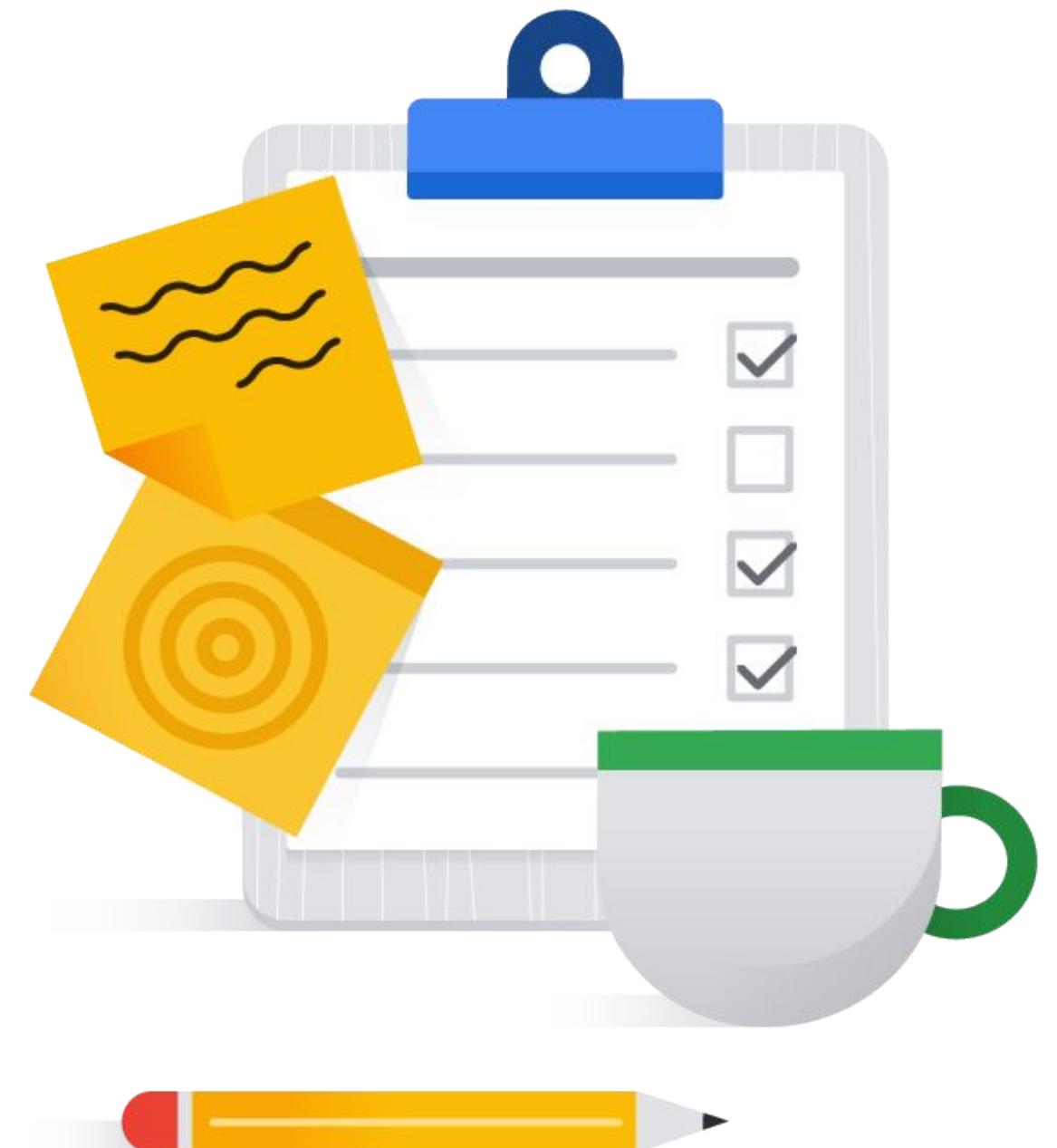
105



## Summary

# Topics

01	Summary
02	Next Steps
03	Q&A



# In this course, you learned ...

01

What Generative AI can do, and how it works

02

How Generative AI solutions can be built for real-world use cases in various industries

03

Which Google Cloud products offer Generative AI functionality, and when to use them

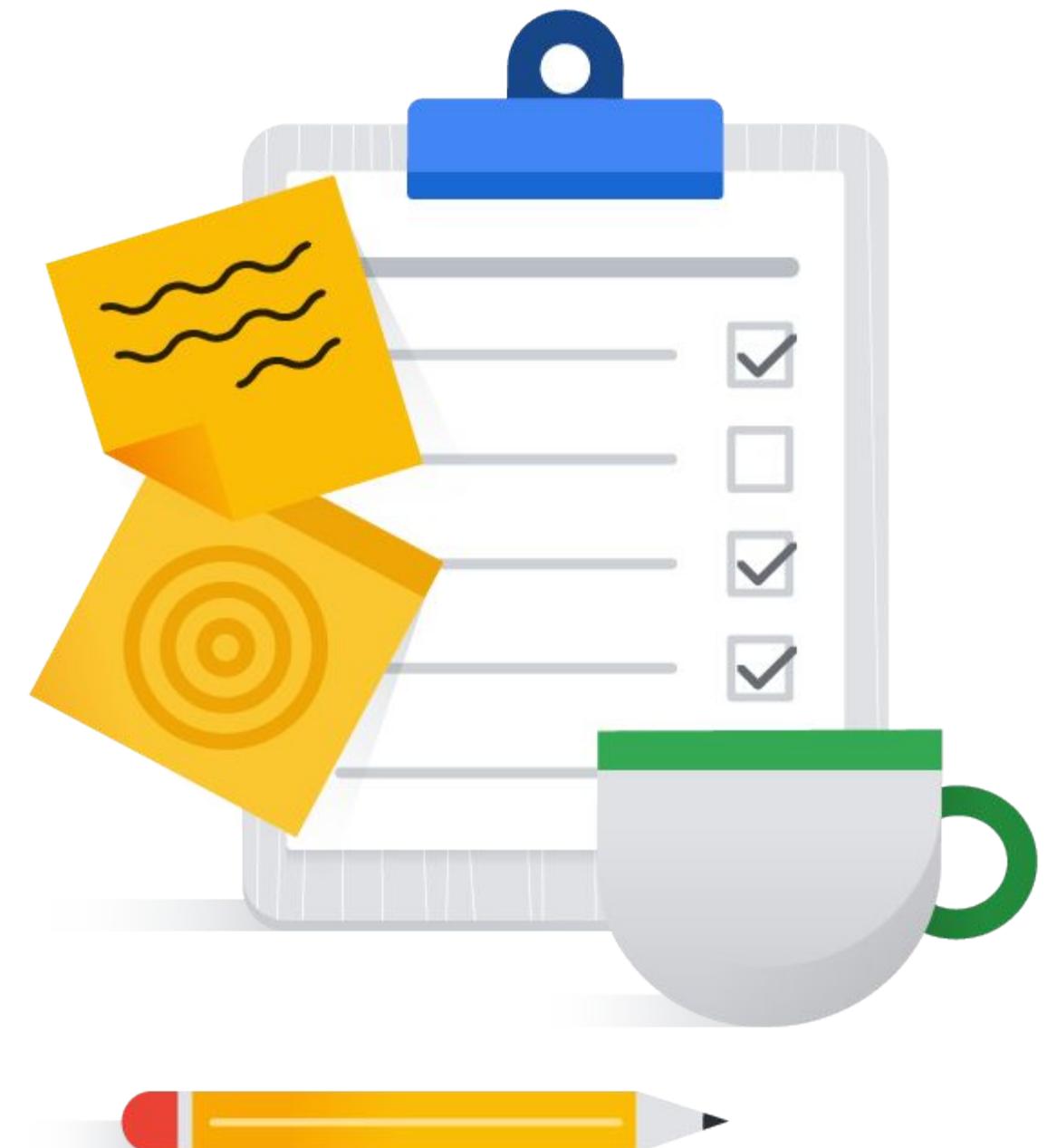
04

How to get started with Generative AI in your organization



# Topics

01	Summary
02	Next Steps
03	Q&A



Please mark your attendance at  
[goo.gle/genai-checkin](https://goo.gl/genai-checkin)

# Thank you for attending this training!

We would love your feedback! Please take 3-5 minutes to complete our survey and help inform content and program related improvements.

- 1 Scan our QR Code (<https://bit.ly/genai-survey>)
- 2 Enter the attendance code provided by your instructor
- 3 Complete the survey



# Take a structured approach to GenAI transformation



## Ramp your skills

Continue your learning journey and complete our [Generative AI Learning Path](#), free of charge, on Google Cloud Skills Boost



## Organizational readiness

Assess your organization's current status and business needs for generative & traditional AI capabilities



## Identify use cases

Select from one of our [Jumpstart GenAI offers](#), and work with Google Cloud to develop a technical design doc and sample code to solve the use case



## Test and scale

Purchase and implement generative AI solutions. Not all AI is built equal. POC often and fail fast to identify what works for your business.

[Contact your Google Cloud Representative to learn more](#)

# 4 GenAI Jumpstart offers - \$25k & 2 weeks per use case



## CREATE

Bring your thoughts and visions to life



## SUMMARIZE

Condense and summarize your knowledge base into a simple format



## DISCOVER

Help your customers and employees find what they need at the right time



## AUTOMATE

Automate your customer service across multiple channels

### Use cases:

- Images from text
- Product descriptions from images
- Blog post from content\*
- Email from content\*
- Release notes from content\*
- Report from content\*
- Press releases from content\*
- Personalized ads\*

### Use cases:

- Content/video summarization
- Intra-knowledge Q&A
- Explanations of code content\*
- External chatbot using internal data\*
- External chatbot using website data\*

### Use cases:

- Search for a document
- Machine-generated event monitoring
- File organization based on content\*
- Exam questions from content\*

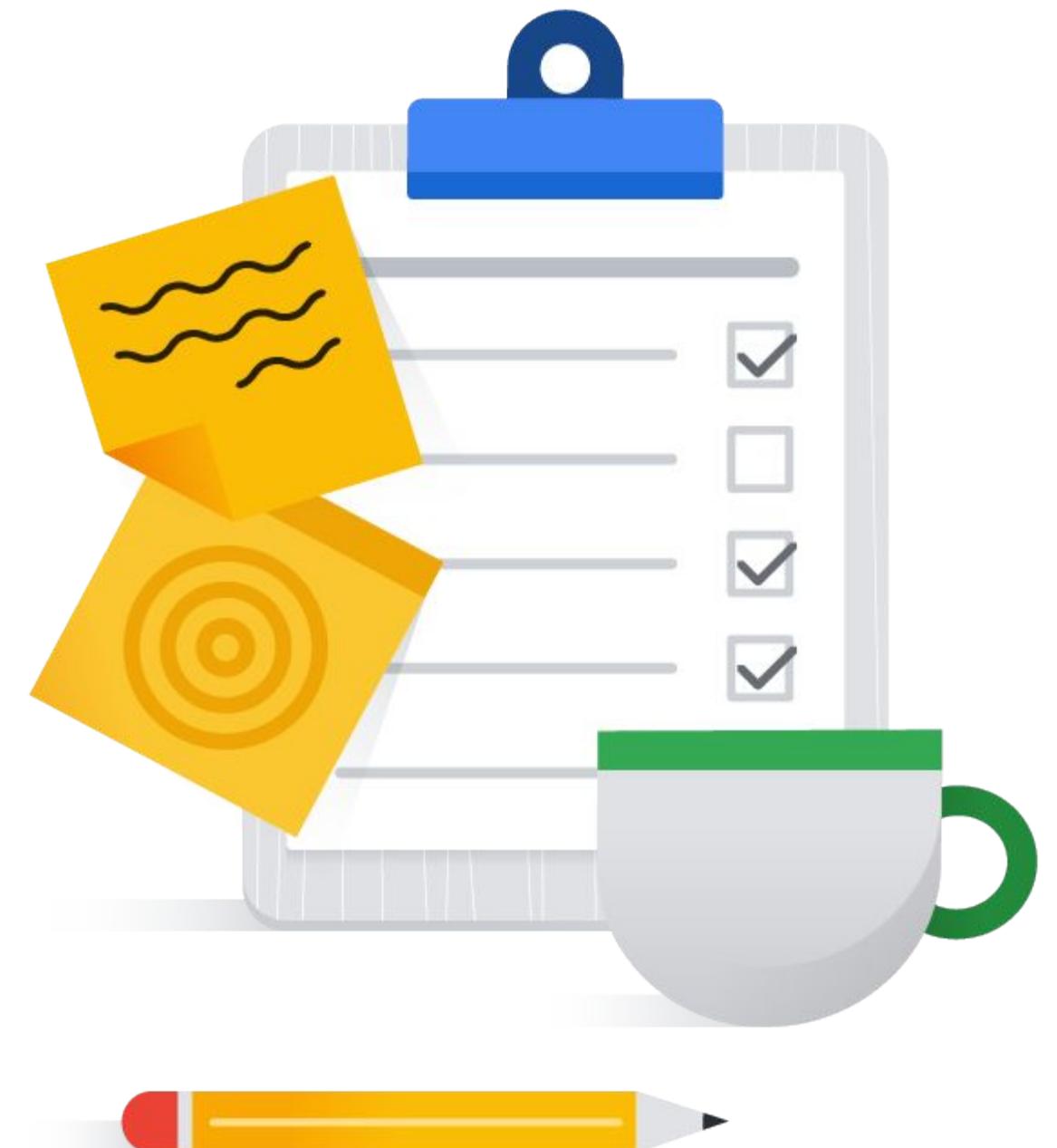
### Use cases:

- Contract information extraction
- Feedback classification and ticket creation
- Sentiment analysis\*
- Content translation\*
- Structured data extraction from file\*
- Media tagging\*
- Product tagging\*
- Content moderation \*

\*may require Responsible AI Review

# Topics

01	Summary
02	Next Steps
03	Q&A



# Is there anything else you want to ask?



