In [ ]:
```
# Copyright 2025 Google LLC
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     https://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
```

Protecting Sensitive Data in Gen AI model responses

Overview

Sensitive Data Protection is a fully managed service designed to discover, classify, and protect your sensitive data wherever it resides. It uses a variety of methods to identify sensitive data including regular expressions, dictionaries, and contextual elements. Once sensitive data is identified, Sensitive Data Protection (Cloud Data Loss Prevention) can take several actions to either classify, mask, encrypt, or even delete it.

Sensitive Data Protection can be accessed via Google Cloud console and used to scan data within Cloud Storage, BigQuery and other Google Cloud services. The following notebook demonstrates using the Python Client for Cloud Data Loss Prevention to incorporate Sensitive Data Protection capabilities directly with Generative AI enabled applications.

With this Python client, you define custom functions that can identify and take corrective action on sensitive data within Large Language Models (LLM) responses in real time. Throughout this notebook, you generate example text with sensitive data and run the results through custom Python functions that redact the sensitive data from Gemini 2.5 Flash model responses, so you can see this functionality in action on example data.

After learning how to work with the Python client, you can adapt these same Python functions for Gen AI applications in your organization to protect sensitive data across your workflows.

Notebook credit: Jim Miller, Google

Objectives

In this lab, you learn how to use Sensitive Data Protection through the Python Client for Cloud Data Loss Prevention and explore how to identify and redact sensitive data within responses from the Gemini 2.5 Flash model.

The steps performed include:

- Installing the Python packages for Vertex AI and Cloud Data Loss Prevention (DLP) API
- Generating examples with sensitive data using Gemini 2.5 Flash model
- Defining and running Python functions to redact different types of sensitive data in Gemini 2.5 Flash model responses using the DLP API

Costs

This tutorial uses billable components of Google Cloud:

- Vertex AI
- Sensitive Data Protection (Cloud Data Loss Prevention)

Learn about Vertex AI pricing and Sensitive Data Protection. Use the Pricing Calculator to generate a cost estimate based on your projected usage.

Getting started with this notebook

Below are few steps to get your environment ready including installing a few key Python packages and setting your environmental variables (project ID and region).

Be sure to run each cell in consecutive order using the `Run` button (play arrow) at the top of this notebook.

Install necessary packages

In [1]:
```
# Install Gen AI
!pip install --upgrade google-genai

# Install Cloud Data Loss Prevention
!pip install google-cloud-dlp --upgrade --user
```

```
Requirement already satisfied: google-genai in /opt/conda/lib/python3.10/sit
e-packages (1.57.0)
Collecting google-genai
  Downloading google_genai-1.59.0-py3-none-any.whl.metadata (53 kB)
Requirement already satisfied: anyio<5.0.0,>=4.8.0 in /opt/conda/lib/python
3.10/site-packages (from google-genai) (4.12.1)
Requirement already satisfied: google-auth<3.0.0,>=2.47.0 in /opt/conda/lib/
python3.10/site-packages (from google-auth[requests]<3.0.0,>=2.47.0->google-
genai) (2.47.0)
Requirement already satisfied: httpx<1.0.0,>=0.28.1 in /opt/conda/lib/python
3.10/site-packages (from google-genai) (0.28.1)
Requirement already satisfied: pydantic<3.0.0,>=2.9.0 in /opt/conda/lib/pyth
on3.10/site-packages (from google-genai) (2.12.5)
Requirement already satisfied: requests<3.0.0,>=2.28.1 in /opt/conda/lib/pyt
hon3.10/site-packages (from google-genai) (2.32.5)
Requirement already satisfied: tenacity<9.2.0,>=8.2.3 in /opt/conda/lib/pyth
on3.10/site-packages (from google-genai) (9.1.2)
Requirement already satisfied: websockets<15.1.0,>=13.0.0 in /opt/conda/lib/
python3.10/site-packages (from google-genai) (15.0.1)
Requirement already satisfied: typing-extensions<5.0.0,>=4.11.0 in /opt/cond
a/lib/python3.10/site-packages (from google-genai) (4.15.0)
Requirement already satisfied: distro<2,>=1.7.0 in /opt/conda/lib/python3.1
0/site-packages (from google-genai) (1.9.0)
Requirement already satisfied: sniffio in /opt/conda/lib/python3.10/site-pac
kages (from google-genai) (1.3.1)
Requirement already satisfied: exceptiongroup>=1.0.2 in /opt/conda/lib/pytho
n3.10/site-packages (from anyio<5.0.0,>=4.8.0->google-genai) (1.3.1)
Requirement already satisfied: idna>=2.8 in /opt/conda/lib/python3.10/site-p
ackages (from anyio<5.0.0,>=4.8.0->google-genai) (3.11)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /opt/conda/lib/pytho
n3.10/site-packages (from google-auth<3.0.0,>=2.47.0->google-auth[requests]<
3.0.0,>=2.47.0->google-genai) (0.4.2)
Requirement already satisfied: rsa<5,>=3.1.4 in /opt/conda/lib/python3.10/si
te-packages (from google-auth<3.0.0,>=2.47.0->google-auth[requests]<3.0.0,>=
2.47.0->google-genai) (4.9.1)
Requirement already satisfied: certifi in /opt/conda/lib/python3.10/site-pac
kages (from httpx<1.0.0,>=0.28.1->google-genai) (2026.1.4)
Requirement already satisfied: httpcore==1.* in /opt/conda/lib/python3.10/si
te-packages (from httpx<1.0.0,>=0.28.1->google-genai) (1.0.9)
Requirement already satisfied: h11>=0.16 in /opt/conda/lib/python3.10/site-p
ackages (from httpcore==1.*->httpx<1.0.0,>=0.28.1->google-genai) (0.16.0)
Requirement already satisfied: annotated-types>=0.6.0 in /opt/conda/lib/pyth
on3.10/site-packages (from pydantic<3.0.0,>=2.9.0->google-genai) (0.7.0)
Requirement already satisfied: pydantic-core==2.41.5 in /opt/conda/lib/pytho
n3.10/site-packages (from pydantic<3.0.0,>=2.9.0->google-genai) (2.41.5)
Requirement already satisfied: typing-inspection>=0.4.2 in /opt/conda/lib/py
thon3.10/site-packages (from pydantic<3.0.0,>=2.9.0->google-genai) (0.4.2)
Requirement already satisfied: charset_normalizer<4,>=2 in /opt/conda/lib/py
thon3.10/site-packages (from requests<3.0.0,>=2.28.1->google-genai) (3.4.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /opt/conda/lib/python3.
10/site-packages (from requests<3.0.0,>=2.28.1->google-genai) (2.6.3)
Requirement already satisfied: pyasn1>=0.1.3 in /opt/conda/lib/python3.10/si
te-packages (from rsa<5,>=3.1.4->google-auth<3.0.0,>=2.47.0->google-auth[req
uests]<3.0.0,>=2.47.0->google-genai) (0.6.1)
Downloading google_genai-1.59.0-py3-none-any.whl (719 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 719.1/719.1 kB 17.1 MB/s  0:00:0
```

```
0
Installing collected packages: google-genai
  Attempting uninstall: google-genai
    Found existing installation: google-genai 1.57.0
    Uninstalling google-genai-1.57.0:
      Successfully uninstalled google-genai-1.57.0
Successfully installed google-genai-1.59.0
Collecting google-cloud-dlp
  Downloading google_cloud_dlp-3.34.0-py3-none-any.whl.metadata (9.9 kB)
Requirement already satisfied: google-api-core!=2.0.*,!=2.1.*,!=2.10.*,!=2.
2.*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,!=2.8.*,!=2.9.*,<3.0.0,>=1.34.1
in /opt/conda/lib/python3.10/site-packages (from google-api-core[grpc]!=2.0.
*,!=2.1.*,!=2.10.*,!=2.2.*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,!=2.8.*,!
=2.9.*,<3.0.0,>=1.34.1->google-cloud-dlp) (2.29.0)
Requirement already satisfied: google-auth!=2.24.0,!=2.25.0,<3.0.0,>=2.14.1
in /opt/conda/lib/python3.10/site-packages (from google-cloud-dlp) (2.47.0)
Requirement already satisfied: grpcio<2.0.0,>=1.33.2 in /opt/conda/lib/pytho
n3.10/site-packages (from google-cloud-dlp) (1.76.0)
Requirement already satisfied: proto-plus<2.0.0,>=1.22.3 in /opt/conda/lib/p
ython3.10/site-packages (from google-cloud-dlp) (1.27.0)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!
=4.21.4,!=4.21.5,<7.0.0,>=3.20.2 in /opt/conda/lib/python3.10/site-packages
(from google-cloud-dlp) (6.33.4)
Requirement already satisfied: googleapis-common-protos<2.0.0,>=1.56.2 in /o
pt/conda/lib/python3.10/site-packages (from google-api-core!=2.0.*,!=2.1.*,!
=2.10.*,!=2.2.*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,!=2.8.*,!=2.9.*,<3.
0.0,>=1.34.1->google-api-core[grpc]!=2.0.*,!=2.1.*,!=2.10.*,!=2.2.*,!=2.3.
*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,!=2.8.*,!=2.9.*,<3.0.0,>=1.34.1->google-cl
oud-dlp) (1.72.0)
Requirement already satisfied: requests<3.0.0,>=2.18.0 in /opt/conda/lib/pyt
hon3.10/site-packages (from google-api-core!=2.0.*,!=2.1.*,!=2.10.*,!=2.2.
*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,!=2.8.*,!=2.9.*,<3.0.0,>=1.34.1->g
oogle-api-core[grpc]!=2.0.*,!=2.1.*,!=2.10.*,!=2.2.*,!=2.3.*,!=2.4.*,!=2.5.
*,!=2.6.*,!=2.7.*,!=2.8.*,!=2.9.*,<3.0.0,>=1.34.1->google-cloud-dlp) (2.32.
5)
Requirement already satisfied: grpcio-status<2.0.0,>=1.33.2 in /opt/conda/li
b/python3.10/site-packages (from google-api-core[grpc]!=2.0.*,!=2.1.*,!=2.1
0.*,!=2.2.*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,!=2.8.*,!=2.9.*,<3.0.0,>
=1.34.1->google-cloud-dlp) (1.76.0)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /opt/conda/lib/pytho
n3.10/site-packages (from google-auth!=2.24.0,!=2.25.0,<3.0.0,>=2.14.1->goog
le-cloud-dlp) (0.4.2)
Requirement already satisfied: rsa<5,>=3.1.4 in /opt/conda/lib/python3.10/si
te-packages (from google-auth!=2.24.0,!=2.25.0,<3.0.0,>=2.14.1->google-cloud
-dlp) (4.9.1)
Requirement already satisfied: typing-extensions~=4.12 in /opt/conda/lib/pyt
hon3.10/site-packages (from grpcio<2.0.0,>=1.33.2->google-cloud-dlp) (4.15.
0)
Requirement already satisfied: charset_normalizer<4,>=2 in /opt/conda/lib/py
thon3.10/site-packages (from requests<3.0.0,>=2.18.0->google-api-core!=2.0.
*,!=2.1.*,!=2.10.*,!=2.2.*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,!=2.8.*,!
=2.9.*,<3.0.0,>=1.34.1->google-api-core[grpc]!=2.0.*,!=2.1.*,!=2.10.*,!=2.2.
*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,!=2.8.*,!=2.9.*,<3.0.0,>=1.34.1->g
oogle-cloud-dlp) (3.4.4)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.10/sit
e-packages (from requests<3.0.0,>=2.18.0->google-api-core!=2.0.*,!=2.1.*,!=
```

```
2.10.*,!=2.2.*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,!=2.8.*,!=2.9.*,<3.0.
0,>=1.34.1->google-api-core[grpc]!=2.0.*,!=2.1.*,!=2.10.*,!=2.2.*,!=2.3.*,!=
2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,!=2.8.*,!=2.9.*,<3.0.0,>=1.34.1->google-cloud-
dlp) (3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in /opt/conda/lib/python3.
10/site-packages (from requests<3.0.0,>=2.18.0->google-api-core!=2.0.*,!=2.
1.*,!=2.10.*,!=2.2.*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,!=2.8.*,!=2.9.
*,<3.0.0,>=1.34.1->google-api-core[grpc]!=2.0.*,!=2.1.*,!=2.10.*,!=2.2.*,!=
2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,!=2.8.*,!=2.9.*,<3.0.0,>=1.34.1->googl
e-cloud-dlp) (2.6.3)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.
10/site-packages (from requests<3.0.0,>=2.18.0->google-api-core!=2.0.*,!=2.
1.*,!=2.10.*,!=2.2.*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,!=2.8.*,!=2.9.
*,<3.0.0,>=1.34.1->google-api-core[grpc]!=2.0.*,!=2.1.*,!=2.10.*,!=2.2.*,!=
2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,!=2.8.*,!=2.9.*,<3.0.0,>=1.34.1->googl
e-cloud-dlp) (2026.1.4)
Requirement already satisfied: pyasn1>=0.1.3 in /opt/conda/lib/python3.10/si
te-packages (from rsa<5,>=3.1.4->google-auth!=2.24.0,!=2.25.0,<3.0.0,>=2.14.
1->google-cloud-dlp) (0.6.1)
Downloading google_cloud_dlp-3.34.0-py3-none-any.whl (220 kB)
Installing collected packages: google-cloud-dlp
Successfully installed google-cloud-dlp-3.34.0
```

Restart current runtime

To use the newly installed packages in this Jupyter runtime, you must restart the runtime. You can do this by running the cell below, which will restart the current kernel.

In [2]:
```
# Restart kernel after installs so that your environment can access the new
import IPython

app = IPython.Application.instance()
app.kernel.do_shutdown(True)
```

Out[2]:
```
{'status': 'ok', 'restart': True}
```

⚠ **The kernel is going to restart. Please wait until it is finished before continuing to the next step.** ⚠

When prompted, click OK to continue.

Set your project ID and region

In [1]:
```
from google import genai
import os
PROJECT_ID = "YOUR_PROJECT_ID "
LOCATION = os.environ.get("GOOGLE_CLOUD_REGION", "global")
client = genai.Client(vertexai=True, project=PROJECT_ID, location=LOCATION)
```

Generate simple example text with personally identifiable information (full name) using Gemini 2.5 Flash model

The Gemini 2.5 Flash (gemini-2.5-flash) model is designed to handle natural language tasks, multi-turn text and code chat, and code generation.

In this section, you use the the model to generate examples of text with personally identifiable information (PII) and then define a custom Python function to redact this sensitive data from the model responses.

In [2]:
```
# Create the API client
client = genai.Client(vertexai=True, project=PROJECT_ID, location=LOCATION)
model = "gemini-2.5-flash"
```

In [3]:
```
# Write a prompt that generates a simple example of personally identifiable
prompt = f"""Who is the CEO of Google?
  """

# Run model with prompt
response_name = client.models.generate_content(
    model=model,
    contents=prompt
)

# Print response without deidentification (full name is visible)
response_name
```

Out[3]:
```
GenerateContentResponse(
  automatic_function_calling_history=[],
  candidates=[
    Candidate(
      avg_logprobs=-1.496058428729022,
      content=Content(
        parts=[
          Part(
            text="""The CEO of Google is **Sundar Pichai**.

He is also the CEO of Google's parent company, Alphabet Inc."""
          ),
        ],
        role='model'
      ),
      finish_reason=<FinishReason.STOP: 'STOP'>
    ),
  ],
  create_time=datetime.datetime(2026, 1, 18, 8, 31, 42, 571430, tzinfo=TzIn
fo(0)),
  model_version='gemini-2.5-flash',
  response_id='bppsaabwIsv1hMIP1MqkuAM',
  sdk_http_response=HttpResponse(
    headers=<dict len=9>
  ),
  usage_metadata=GenerateContentResponseUsageMetadata(
    candidates_token_count=27,
    candidates_tokens_details=[
      ModalityTokenCount(
        modality=<MediaModality.TEXT: 'TEXT'>,
        token_count=27
      ),
    ],
    prompt_token_count=9,
    prompt_tokens_details=[
      ModalityTokenCount(
        modality=<MediaModality.TEXT: 'TEXT'>,
        token_count=9
      ),
    ],
    thoughts_token_count=151,
    total_token_count=187,
    traffic_type=<TrafficType.ON_DEMAND: 'ON_DEMAND'>
  )
)
```

Define and run a Python function to deidentify Gemini 2.5 Flash model responses using built-in global infotypes

Sensitive Data Protection uses information types, or infoTypes, to define what it scans for. An infoType is a type of sensitive data, such as a name, telephone number, or identification number.

In the cell below, you define a Python function that identifies and redacts that specific infoTypes that you provide as input, based on the list of built-in global infoTypes that are available in Sensitive Data Protection. Global infoTypes include general and globally applicable infoTypes such as names, date of birth, and credit card numbers.

When you apply the function to model responses, you specify a few key built-in infoTypes to redact, such as `PERSON_NAME`, `DATE_OF_BIRTH`, and `CREDIT_CARD_NUMBER`. You can review the documentation to see the full list of built-in infoTypes.

Run the code block below without modifications.

In [4]:
```
# Define function to inspect and deidentify output with Sensitive Data Prote
import google.cloud.dlp
from typing import List

def deidentify_with_replace_infotype(
    project: str, item: str, info_types: List[str]
) -> None:
    """Uses the Data Loss Prevention API to deidentify sensitive data in a
    string by replacing it with the info type.
    Args:
        project: The Google Cloud project id to use as a parent resource.
        item: The string to deidentify (will be treated as text).
        info_types: A list of strings representing info types to look for.
            A full list of info type categories can be fetched from the API.
    Returns:
        None; the response from the API is printed to the terminal.
    """

    # Instantiate a client
    dlp = google.cloud.dlp_v2.DlpServiceClient()

    # Convert the project id into a full resource id.
    parent = f"projects/{PROJECT_ID}"

    # Construct inspect configuration dictionary
    inspect_config = {"info_types": [{"name": info_type} for info_type in ir

    # Construct deidentify configuration dictionary
    deidentify_config = {
        "info_type_transformations": {
            "transformations": [
                {"primitive_transformation": {"replace_with_info_type_config
            ]
        }
    }

    # Call the API
    response = dlp.deidentify_content(
        request={
            "parent": parent,
            "deidentify_config": deidentify_config,
```

```
            "inspect_config": inspect_config,
            "item": {"value": item},
        }
    )

    # Print results
    print(response.item.value)
```

```
/opt/conda/lib/python3.10/site-packages/google/api_core/_python_version_supp
ort.py:275: FutureWarning: You are using a Python version (3.10.19) which Go
ogle will stop supporting in new releases of google.cloud.dlp_v2 once it rea
ches its end of life (2026-10-04). Please upgrade to the latest Python versi
on, or at least Python 3.11, to continue receiving updates for google.cloud.
dlp_v2 past that date.
  warnings.warn(message, FutureWarning)
```

In [5]:
```
# Deidentify model response that includes a person's name (full name is reda
deidentify_with_replace_infotype(PROJECT_ID, response_name.text, ["PERSON_NA
```

```
The CEO of Google is **[PERSON_NAME]**.

He is also the CEO of Google's parent company, Alphabet Inc.
```

Generate and de-identify example text with more personally identifiable information (date of birth) using Gemini 2.5 Flash model

In this example, you generate an example with more personally identifiable information in the form of a medical visit log, which can include other sensitive data such date of birth.

When you run the de-identification function, you provide `PERSON_NAME` and `DATE_OF_BIRTH` as the infoTypes to redact.

In [6]:
```
# Write a prompt that generates an example with more personally identifiable
prompt = f"""Generate an example medical after-visit log with faux personall
  """

# Run model with prompt
response_visitlog = client.models.generate_content(
    model=model,
    contents=prompt
)

# Print response without deidentification (full names and date of birth are
response_visitlog
```

Out[6]:
```
GenerateContentResponse(
  automatic_function_calling_history=[],
  candidates=[
    Candidate(
      avg_logprobs=-0.5369249722252973,
      content=Content(
        parts=[
          Part(
            text="""Okay, here is an example of a medical after-visit log w
ith faux personally identifiable information (PII).

---

**[CLINIC LOGO/NAME]**
**Evergreen Family Practice**
123 Wellness Way, Anytown, CA 90210
Phone: (555) 888-9999 | Fax: (555) 888-9998
www.evergreenfp.com

---

**AFTER-VISIT SUMMARY**

**Patient Information:**
*   **Patient Name:** Eleanor Vance
*   **Date of Birth:** October 15, 1978 (Age: 45)
*   **Medical Record #: EV101578
*   **Contact Phone:** (555) 123-4567
*   **Email:** eleanor.vance@example.com
*   **Address:** 456 Oak Avenue, Pleasantville, NY 10001

**Visit Details:**
*   **Date of Visit:** December 5, 2023
*   **Time of Visit:** 10:30 AM
*   **Provider:** Dr. Sarah Chen, MD
*   **Reason for Visit (Chief Complaint):** Follow-up on persistent headach
es and generalized fatigue.

**Summary of Visit:**
Ms. Vance presented for follow-up regarding her persistent mild headaches
(primarily frontal/temporal) and generalized fatigue over the past 3 month
s. She reports headaches are generally 3/10 severity, non-throbbing, and oc
cur several times a week. Fatigue is described as constant low energy, not
significantly relieved by sleep. She denies fever, weight loss, visual chan
ges, or focal weakness. Reviewed recent lab results (CBC, CMP, Thyroid Pane
l, Vitamin D) which were all within normal limits.

Physical exam today was unremarkable. Blood pressure 120/80, HR 72, Temp 9
8.6°F. Neurological exam (cranial nerves, motor, sensory, reflexes, coordin
ation) was normal. Funduscopic exam was also normal.

**Assessment/Diagnosis:**
1.  Tension-type headaches (G44.2)
2.  Fatigue, generalized (R53.81)
    *   *Likely contributing factors: Stress and sleep hygiene.*
```

```
**Treatment Plan & Recommendations:**

*   **Medications:**
    *   Continue with Ibuprofen 400mg as needed for headaches, limit to 2-3
days per week to avoid medication overuse headaches.
    *   No new prescriptions issued today.

*   **Lifestyle Modifications:**
    *   **Exercise:** Recommended incorporating a daily brisk walk (30 minu
tes) or other moderate activity.
    *   **Sleep Hygiene:** Focus on a consistent sleep schedule, ensuring 7
-8 hours of sleep per night. Avoid screens 1 hour before bed.
    *   **Stress Reduction:** Suggested exploring stress reduction techniqu
es such as mindfulness apps, deep breathing exercises, or gentle yoga.
    *   **Hydration:** Increase daily water intake.
    *   **Diet:** Consider keeping a food diary for 2 weeks to identify pot
ential dietary triggers for headaches.

*   **Referrals:**
    *   No specialist referrals indicated at this time given normal physica
l exam and lab results. Will re-evaluate if symptoms persist or worsen.

*   **Further Testing:**
    *   No additional laboratory or imaging studies ordered at this visit.

**Next Steps:**

*   **Follow-up Appointment:** Please schedule a follow-up visit in **4-6 w
eeks** to reassess symptoms and review progress. Call the office at (555) 8
88-9999 to schedule.
*   **When to Call the Office:**
    *   If headaches worsen significantly, become more frequent, or are ass
ociated with new neurological symptoms (e.g., weakness, numbness, visual lo
ss, difficulty speaking).
    *   If fatigue becomes debilitating or is accompanied by new concerning
symptoms.
    *   If you have any questions or concerns regarding your treatment pla
n.
*   **Emergency Instructions:**
    *   If you experience a severe, sudden-onset headache (often described
as "the worst headache of your life"), loss of consciousness, seizure, or a
cute vision changes, **call 911 immediately or go to the nearest emergency
room.**

---
**Provider Signature:**


_____
Dr. Sarah Chen, MD
Evergreen Family Practice
Date: December 5, 2023

---"""
          ),
        ],
        role='model'
```

```
      ),
      finish_reason=<FinishReason.STOP: 'STOP'>
    ),
  ],
  create_time=datetime.datetime(2026, 1, 18, 8, 34, 38, 825601, tzinfo=TzIn
fo(0)),
  model_version='gemini-2.5-flash',
  response_id='HptsaYGyMtWBqsMPjdvC8Qk',
  sdk_http_response=HttpResponse(
    headers=<dict len=9>
  ),
  usage_metadata=GenerateContentResponseUsageMetadata(
    candidates_token_count=988,
    candidates_tokens_details=[
      ModalityTokenCount(
        modality=<MediaModality.TEXT: 'TEXT'>,
        token_count=988
      ),
    ],
    prompt_token_count=21,
    prompt_tokens_details=[
      ModalityTokenCount(
        modality=<MediaModality.TEXT: 'TEXT'>,
        token_count=21
      ),
    ],
    thoughts_token_count=1643,
    total_token_count=2652,
    traffic_type=<TrafficType.ON_DEMAND: 'ON_DEMAND'>
  )
)
```

In [7]:
```
# Deidentify model response that includes an example medical visit log (full
deidentify_with_replace_infotype(PROJECT_ID, response_visitlog.text, ["PERSO
```

Okay, here is an example of a medical after-visit log with faux personally identifiable information (PII).

[CLINIC LOGO/NAME]
Evergreen Family Practice
123 Wellness Way, Anytown, CA 90210
Phone: (555) 888-9999 | Fax: (555) 888-9998
www.evergreenfp.com

AFTER-VISIT SUMMARY

Patient Information:
*   **Patient Name:** [PERSON_NAME]
*   **Date of Birth:** [DATE_OF_BIRTH] (Age: 45)
*   **Medical Record #: EV101578
*   **Contact Phone:** (555) 123-4567
*   **Email:** [PERSON_NAME].[PERSON_NAME]@example.com
*   **Address:** 456 Oak Avenue, Pleasantville, NY 10001

Visit Details:
*   **Date of Visit:** December 5, 2023
*   **Time of Visit:** 10:30 AM
*   **Provider:** Dr. [PERSON_NAME], MD
*   **Reason for Visit (Chief Complaint):** Follow-up on persistent headaches and generalized fatigue.

Summary of Visit:
[PERSON_NAME] presented for follow-up regarding her persistent mild headaches (primarily frontal/temporal) and generalized fatigue over the past 3 months. She reports headaches are generally 3/10 severity, non-throbbing, and occur several times a week. Fatigue is described as constant low energy, not significantly relieved by sleep. She denies fever, weight loss, visual changes, or focal weakness. Reviewed recent lab results (CBC, CMP, Thyroid Panel, Vitamin D) which were all within normal limits.

Physical exam today was unremarkable. Blood pressure 120/80, HR 72, Temp 98.6°F. Neurological exam (cranial nerves, motor, sensory, reflexes, coordination) was normal. Funduscopic exam was also normal.

Assessment/Diagnosis:
1. Tension-type headaches (G44.2)
2. Fatigue, generalized (R53.81)
    *   *Likely contributing factors: Stress and sleep hygiene.*

Treatment Plan & Recommendations:

*   **Medications:**
    *   Continue with Ibuprofen 400mg as needed for headaches, limit to 2-3 days per week to avoid medication overuse headaches.
    *   No new prescriptions issued today.

*   **Lifestyle Modifications:**

```
    *   **Exercise:** Recommended incorporating a daily brisk walk (30 minut
es) or other moderate activity.
    *   **Sleep Hygiene:** Focus on a consistent sleep schedule, ensuring 7-
8 hours of sleep per night. Avoid screens 1 hour before bed.
    *   **Stress Reduction:** Suggested exploring stress reduction technique
s such as mindfulness apps, deep breathing exercises, or gentle yoga.
    *   **Hydration:** Increase daily water intake.
    *   **Diet:** Consider keeping a food diary for 2 weeks to identify pote
ntial dietary triggers for headaches.

*   **Referrals:**
    *   No specialist referrals indicated at this time given normal physical
exam and lab results. Will re-evaluate if symptoms persist or worsen.

*   **Further Testing:**
    *   No additional laboratory or imaging studies ordered at this visit.

**Next Steps:**

*   **Follow-up Appointment:** Please schedule a follow-up visit in **4-6 we
eks** to reassess symptoms and review progress. Call the office at (555) 888
-9999 to schedule.
*   **When to Call the Office:**
    *   If headaches worsen significantly, become more frequent, or are asso
ciated with new neurological symptoms (e.g., weakness, numbness, visual los
s, difficulty speaking).
    *   If fatigue becomes debilitating or is accompanied by new concerning
symptoms.
    *   If you have any questions or concerns regarding your treatment plan.
*   **Emergency Instructions:**
    *   If you experience a severe, sudden-onset headache (often described a
s "the worst headache of your life"), loss of consciousness, seizure, or acu
te vision changes, **call 911 immediately or go to the nearest emergency roo
m.**

---
**Provider Signature:**

_____
Dr. [PERSON_NAME], MD
Evergreen Family Practice
Date: December 5, 2023

---
```

Generate example text with credit card information using Gemini 2.5 Flash model

In the previous examples, you generated example text with personally identifiable information such as full name and date of birth.

In this example, you start with generating example text with credit card information with the prompt provided below. Then, you apply what you have learned in the previous

examples to run the function to redact credit card information.

In [9]:
```
# Write a prompt that generates an example with a credit card number
prompt = f"""Is 4111 1111 1111 1111 an example of a credit card number?
  """

# Run model with prompt
response_creditcard = client.models.generate_content(
    model=model,
    contents=prompt
)

# Print response without deidentification (credit card number is visible)
response_creditcard
```

Out[9]:
```
GenerateContentResponse(
  automatic_function_calling_history=[],
  candidates=[
    Candidate(
      avg_logprobs=-0.7434920387706537,
      content=Content(
        parts=[
          Part(
            text="""Yes, from a purely technical standpoint, **it meets sev
eral key criteria to be *considered* an example of a credit card number**,
specifically a Visa card, and it passes the Luhn algorithm checksum.

Here's why:

1.  **Length:** It has 16 digits, which is a standard length for Visa and M
astercard.
2.  **Starting Digit:** It starts with a '4', which is the Major Industry I
dentifier (MII) for Visa cards.
3.  **Luhn Algorithm (Mod 10 Check):** Let's quickly check it:
    *   Number: 4111 1111 1111 1111
    *   Reverse and double every second digit (from the right, skipping the
first):
        *   1 (no double)
        *   1 (x2 = 2)
        *   1 (no double)
        *   1 (x2 = 2)
        *   1 (no double)
        *   1 (x2 = 2)
        *   1 (no double)
        *   1 (x2 = 2)
        *   1 (no double)
        *   1 (x2 = 2)
        *   1 (no double)
        *   1 (x2 = 2)
        *   1 (no double)
        *   1 (x2 = 2)
        *   1 (no double)
        *   4 (x2 = 8)
    *   Sum all digits (if doubling results in a two-digit number, sum its
digits, e.g., 12 becomes 1+2=3. In this case, no such numbers):
        1 + 2 + 1 + 2 + 1 + 2 + 1 + 2 + 1 + 2 + 1 + 2 + 1 + 2 + 1 + 8 = 30
    *   Since 30 is a multiple of 10, the number **passes the Luhn algorith
m**.

**However, it is extremely unlikely to be a *real, issued* credit card numb
er.**

*   **Repetitive Pattern:** Real credit card numbers have much more complex
and varied sequences of digits to ensure uniqueness and security. The "1111
1111 1111" pattern is highly unusual and would almost certainly never be is
sued by a bank.
*   **Purpose:** Numbers with such simple, repetitive patterns that pass th
e Luhn algorithm are often used for testing, placeholder data, or examples
in documentation.

So, while it **technically fits the format and passes the checksum for a Vi
```

```
sa card**, it is not a number that would ever be found on a legitimate, act
ive credit card."""
          ),
        ],
        role='model'
      ),
      finish_reason=<FinishReason.STOP: 'STOP'>
    ),
  ],
  create_time=datetime.datetime(2026, 1, 18, 8, 36, 14, 379161, tzinfo=TzIn
fo(0)),
  model_version='gemini-2.5-flash',
  response_id='fptsaZmSF9qphMIP27iVqAI',
  sdk_http_response=HttpResponse(
    headers=<dict len=9>
  ),
  usage_metadata=GenerateContentResponseUsageMetadata(
    candidates_token_count=609,
    candidates_tokens_details=[
      ModalityTokenCount(
        modality=<MediaModality.TEXT: 'TEXT'>,
        token_count=609
      ),
    ],
    prompt_token_count=31,
    prompt_tokens_details=[
      ModalityTokenCount(
        modality=<MediaModality.TEXT: 'TEXT'>,
        token_count=31
      ),
    ],
    thoughts_token_count=1422,
    total_token_count=2062,
    traffic_type=<TrafficType.ON_DEMAND: 'ON_DEMAND'>
  )
)
```

Apply your skills using the built-in global infoType for credit card number

Now it's your turn to call the function `deidentify_with_replace_infotype` with the appropriate inputs to redact credit card numbers from model responses.

Hint: you can review the global infoTypes in the documentation to identify the appropriate infoType for credit card numbers.

For the full solution, return to the lab instructions and expand the **Hint** button.

In [15]:
```
# Deidentify model response that includes an example credit card number (cre

# ADD YOUR CODE BELOW
deidentify_with_replace_infotype(PROJECT_ID, response_creditcard.text, ["CRE
```

Yes, from a purely technical standpoint, **it meets several key criteria to be *considered* an example of a credit card number**, specifically a Visa card, and it passes the Luhn algorithm checksum.

Here's why:

1. **Length:** It has 16 digits, which is a standard length for Visa and Mastercard.
2. **Starting Digit:** It starts with a '4', which is the Major Industry Identifier (MII) for Visa cards.
3. **Luhn Algorithm (Mod 10 Check):** Let's quickly check it:
    * Number: [CREDIT_CARD_NUMBER]
    * Reverse and double every second digit (from the right, skipping the first):
        * 1 (no double)
        * 1 (x2 = 2)
        * 1 (no double)
        * 1 (x2 = 2)
        * 1 (no double)
        * 1 (x2 = 2)
        * 1 (no double)
        * 1 (x2 = 2)
        * 1 (no double)
        * 1 (x2 = 2)
        * 1 (no double)
        * 1 (x2 = 2)
        * 1 (no double)
        * 1 (x2 = 2)
        * 1 (no double)
        * 4 (x2 = 8)
    * Sum all digits (if doubling results in a two-digit number, sum its digits, e.g., 12 becomes 1+2=3. In this case, no such numbers):
        1 + 2 + 1 + 2 + 1 + 2 + 1 + 2 + 1 + 2 + 1 + 2 + 1 + 2 + 1 + 8 = 30
    * Since 30 is a multiple of 10, the number **passes the Luhn algorithm**.

However, it is extremely unlikely to be a *real, issued* credit card number.

* **Repetitive Pattern:** Real credit card numbers have much more complex and varied sequences of digits to ensure uniqueness and security. The "1111 1111 1111 1111" pattern is highly unusual and would almost certainly never be issued by a bank.
* **Purpose:** Numbers with such simple, repetitive patterns that pass the Luhn algorithm are often used for testing, placeholder data, or examples in documentation.

So, while it **technically fits the format and passes the checksum for a Visa card**, it is not a number that would ever be found on a legitimate, active credit card.

Redefine the Python function to block Gemini 2.5 Flash model responses based on specific infotypes for documents

In addition to its ability to scan and classify information contained within documents, Sensitive Data Protection can classify documents into multiple enterprise-specific categories. When combined with sensitive data inspection, this classification can be useful for document risk assessment, policy enforcement, and similar use cases.

In this section, you redefine the the original function to take advantage of this classification functionality and use it to block output for two specific document infoTypes: source code and patents.

In the code block below for the function, notice the new code lines after `# Add conditional return for document infoTypes for source code and patent`.

Run the code block below without modifications.

In [11]:
```
# Redefine original function to inspect and deidentify output with Sensitive
import google.cloud.dlp
from typing import List

def deidentify_with_replace_infotype(
    project: str, item: str, info_types: List[str]
) -> None:
    """Uses the Data Loss Prevention API to deidentify sensitive data in a
    string by replacing it with the info type.
    Args:
        project: The Google Cloud project id to use as a parent resource.
        item: The string to deidentify (will be treated as text).
        info_types: A list of strings representing info types to look for.
            A full list of info type categories can be fetched from the API.
    Returns:
        None; the response from the API is printed to the terminal.
    """

    # Instantiate a client
    dlp = google.cloud.dlp_v2.DlpServiceClient()

    # Convert the project id into a full resource id.
    parent = f"projects/{PROJECT_ID}"

    # Construct inspect configuration dictionary
    inspect_config = {"info_types": [{"name": info_type} for info_type in ir

    # Construct deidentify configuration dictionary
    deidentify_config = {
        "info_type_transformations": {
            "transformations": [
                {"primitive_transformation": {"replace_with_info_type_config
            ]
        }
    }

    # Call the API for deidentify
```

```
    response = dlp.deidentify_content(
        request={
            "parent": parent,
            "deidentify_config": deidentify_config,
            "inspect_config": inspect_config,
            "item": {"value": item},
        }
    )

    return_payload = response.item.value

    # Add conditional return to block responses containing document infoType
    info_types = ["DOCUMENT_TYPE/R&D/SOURCE_CODE","DOCUMENT_TYPE/R&D/PATENT"
    inspect_config = {"info_types": [{"name": info_type} for info_type in ir

    response = dlp.inspect_content(
        request={
            "parent": parent,
            "inspect_config": inspect_config,
            "item": {"value": item},
        }
    )

    if response.result.findings:
        for finding in response.result.findings:
            if finding.info_type.name == "DOCUMENT_TYPE/R&D/SOURCE_CODE":
                return_payload = '[Blocked due to category: Source Code]'
            elif finding.info_type.name == "DOCUMENT_TYPE/R&D/PATENT":
                return_payload = '[Blocked due to category: Patent Related]'

    # Print results
    print(return_payload)
```

Generate an example with source code using Gemini 2.5 Flash model and block results

In the previous examples, you generated example text with personally identifiable information.

In this example, you generate examples with document infoTypes including source code and patent information. Then, you apply what you have learned in the previous examples to run the function to block responses based on these document infoTypes. Also, in this example, when generating the response, you keep the temperature to `0`, so that the generated output is invariable.

In [16]:
```
# Create prompt that generates an example of Java code
prompt = f"""Show me an example of Java code
  """

# Run model with prompt
response_sourcecode = client.models.generate_content(
    model=model,
```

```
    contents=prompt,
    config={
        "temperature": 0,
        }
)

# Print response without blocking it (code is visible)
response_sourcecode
```

Out[16]:
```
GenerateContentResponse(
  automatic_function_calling_history=[],
  candidates=[
    Candidate(
      avg_logprobs=-0.17896677762849283,
      content=Content(
        parts=[
          Part(
            text="""Okay, here's a simple and common example of Java code.
This program will:

1.  Print a "Hello, World!" message.
2.  Ask the user for their name.
3.  Greet the user by name.
4.  Ask for two numbers.
5.  Calculate and display their sum and product.
6.  Use a simple `if-else` statement.
7.  Demonstrate a basic method (function).

---

```java
// 1. Package declaration (optional for simple single-file programs, but go
od practice)
// package com.example.myfirstapp;

// 2. Import necessary classes from the Java standard library
import java.util.Scanner; // Used for reading user input from the console

/**
 * This is a simple Java program that demonstrates basic input/output,
 * variable usage, arithmetic operations, conditional statements,
 * and method creation.
 */
public class MyFirstJavaApp {

    // 3. The main method: This is the entry point of any Java application.
    //    - public: The method is accessible from anywhere.
    //    - static: The method belongs to the class, not to an instance of
the class.
    //    - void: The method does not return any value.
    //    - String[] args: An array of String objects that can be used to p
ass
    //                     command-line arguments to the program.
    public static void main(String[] args) {

        // --- Section 1: Basic Output and User Greeting ---

        // Print a simple message to the console.
        // System.out.println() prints the string and then moves to the nex
t line.
        System.out.println("Hello, Java World!");
        System.out.println("Welcome to your first Java program!");

        // Create a Scanner object to read input from the console (System.i
n).
```
```

```
        Scanner scanner = new Scanner(System.in);

        // Prompt the user for their name.
        // System.out.print() prints the string without moving to the next
line,
        // so the user's input appears on the same line as the prompt.
        System.out.print("What's your name? ");

        // Read the entire line of input from the user and store it in a St
ring variable.
        String userName = scanner.nextLine();

        // Greet the user using their input.
        // The '+' operator concatenates (joins) strings.
        System.out.println("Nice to meet you, " + userName + "!");

        // --- Section 2: Arithmetic Operations and Variables ---

        System.out.println("\nLet's do some simple math, " + userName +
"!");

        // Prompt for the first number.
        System.out.print("Enter the first whole number: ");
        // Read an integer from the user.
        int number1 = scanner.nextInt(); // 'int' is a primitive data type
for whole numbers

        // Prompt for the second number.
        System.out.print("Enter the second whole number: ");
        int number2 = scanner.nextInt();

        // Perform arithmetic operations
        int sum = number1 + number2;
        int product = number1 * number2;

        // Display the results
        System.out.println("The sum of " + number1 + " and " + number2 + "
is: " + sum);
        System.out.println("The product of " + number1 + " and " + number2
+ " is: " + product);

        // --- Section 3: Conditional Statement (if-else) ---

        System.out.println("\nNow for a little logic...");
        if (sum > 100) {
            System.out.println("Wow, that's a big sum!");
        } else if (sum > 50) {
            System.out.println("That's a moderately large sum.");
        } else {
            System.out.println("That's a smaller sum.");
        }

        // --- Section 4: Calling a Custom Method ---

        System.out.println("\nLet's use a custom method to find the differe
nce.");
```

```
        // Call the 'calculateDifference' method defined below
        int difference = calculateDifference(number1, number2);
        System.out.println("The difference between " + number1 + " and " +
number2 + " is: " + difference);

        // It's good practice to close the scanner when you're done with it
        // to release system resources.
        scanner.close();

        System.out.println("\nProgram finished. Goodbye!");
    }

    // 4. A custom method (function)
    //    - public: Accessible from anywhere.
    //    - static: Belongs to the class, can be called directly from main.
    //    - int: The method returns an integer value.
    //    - (int a, int b): The method takes two integer parameters.
    public static int calculateDifference(int a, int b) {
        // Return the absolute difference to ensure a positive result
        return Math.abs(a - b);
    }
}
```

How to Compile and Run This Java Code:

1. **Save:** Save the code in a file named `MyFirstJavaApp.java`. The file name *must* exactly match the public class name.

2. **Open Terminal/Command Prompt:** Navigate to the directory where you saved the file.

3. **Compile:** Use the Java compiler (`javac`) to compile the source code into bytecode.
    ```bash
    javac MyFirstJavaApp.java
    ```
    If there are no errors, this will create a file named `MyFirstJavaApp.class` in the same directory. This `.class` file contains the Java bytecode that the Java Virtual Machine (JVM) can understand.

4. **Run:** Use the Java Virtual Machine (`java`) to execute the compiled bytecode.
    ```bash
    java MyFirstJavaApp
    ```

5. **Interact:** The program will then prompt you for your name and numbers in the console.

This example covers the very basics and should give you a good starting point for understanding Java's structure and fundamental concepts!"""
          ),
        ],

```
        role='model'
      ),
      finish_reason=<FinishReason.STOP: 'STOP'>
    ),
  ],
  create_time=datetime.datetime(2026, 1, 18, 8, 48, 34, 379379, tzinfo=TzIn
fo(0)),
  model_version='gemini-2.5-flash',
  response_id='Yp5safOTF5DyhMIP0LDkgQk',
  sdk_http_response=HttpResponse(
    headers=<dict len=9>
  ),
  usage_metadata=GenerateContentResponseUsageMetadata(
    candidates_token_count=1389,
    candidates_tokens_details=[
      ModalityTokenCount(
        modality=<MediaModality.TEXT: 'TEXT'>,
        token_count=1389
      ),
    ],
    prompt_token_count=9,
    prompt_tokens_details=[
      ModalityTokenCount(
        modality=<MediaModality.TEXT: 'TEXT'>,
        token_count=9
      ),
    ],
    thoughts_token_count=1587,
    total_token_count=2985,
    traffic_type=<TrafficType.ON_DEMAND: 'ON_DEMAND'>
  )
)
```

In [17]:
```
# Block model response that include source code (response is not available)
# Notice that the infoType that you request is a different infoType
# Results are still blocked because the model response is identified contain
deidentify_with_replace_infotype(PROJECT_ID, response_sourcecode.text, ["EMA
```

```
[Blocked due to category: Source Code]
```

Apply your skills using the built-in document infoType for patents

Now it's your turn to call the function `deidentify_with_replace_infotype` with the appropriate inputs to block patent information in model responses.

Hint: review the previous two cells for generating an example with source code and calling the function, and then modify both to block the model response because it contains patent information.

For the full solution, return to the lab instructions and expand the **Hint** button.

In [21]:
```
# Create prompt that generates example patent

prompt = f"""Show me an example patent

"""

# Run model with prompt

# Name the output as response_patent

response_patent = client.models.generate_content(model=model, contents=promp

# Print response without blocking it (patent information provided)

response_patent
```

Out[21]:
```
GenerateContentResponse(
  automatic_function_calling_history=[],
  candidates=[
    Candidate(
      avg_logprobs=-0.21693781701383422,
      citation_metadata=CitationMetadata(
        citations=[
          Citation(
            end_index=13792,
            start_index=13510,
            uri='https://patents.google.com/patent/US20150038605A1/en'
          ),
        ]
      ),
      content=Content(
        parts=[
          Part(
            text="""Okay, let's create a **fictional, simplified example of
a utility patent** for an invention. Real patents are often hundreds of pag
es long and incredibly detailed, but this will give you a good idea of the
structure and content.

**Invention:** A "Smart Pet Feeding and Activity Monitoring System"

---

## UNITED STATES PATENT

**Patent No.:** US 11,234,567 B2
**Date of Patent:** Dec. 12, 2023

**Title:** SMART PET FEEDING AND ACTIVITY MONITORING SYSTEM

**Inventors:**
*   Dr. Anya Sharma, San Francisco, CA (US)
*   Mr. Ben Carter, Seattle, WA (US)

**Assignee:** Pawsitive Innovations Inc., San Francisco, CA (US)

**Filing Date:** May 15, 2022
**Appl. No.:** 17/123,456

**International Class:** A01K 5/02 (2006.01); G06F 19/00 (2018.01)

---

### ABSTRACT

A smart pet feeding and activity monitoring system is disclosed, comprising
an automated pet feeder with an integrated activity sensor and a communicat
ion module. The system is configured to dispense pet food according to a cu
stomizable schedule, monitor the pet's activity levels, and transmit data t
o a remote computing device (e.g., smartphone, tablet). The system further
includes a processor for analyzing feeding patterns and activity data, and
for generating health insights and alerts for the pet owner. Remote control
of feeding, two-way audio communication, and integrated camera functionalit
```

y may also be provided.

BACKGROUND OF THE INVENTION

1. Field of the Invention
The present invention relates generally to pet care devices, and more particularly to automated pet feeding systems integrated with pet activity monitoring and remote management capabilities.

2. Description of Related Art
Traditional pet feeding often involves manual dispensing of food, which can be inconvenient for owners with busy schedules or those who travel. Automated pet feeders exist, but many are basic, offering only timed dispensing without intelligence or remote control. Separately, pet activity trackers are available, typically worn on a collar, which monitor a pet's movement. However, these systems are often disparate, requiring owners to manage multiple devices and data streams. There is a need for an integrated solution that combines smart feeding with comprehensive activity monitoring and remote management to promote pet health and owner convenience. Existing solutions often fail to provide a holistic view of a pet's dietary intake and physical activity, leading to potential issues like overfeeding, under-exercising, and difficulty in identifying health trends.

SUMMARY OF THE INVENTION

The present invention overcomes the limitations of the prior art by providing a comprehensive Smart Pet Feeding and Activity Monitoring System. The system integrates an automated food dispenser with an activity sensor, a communication module, and a central processing unit. This integration allows for precise, scheduled feeding, real-time monitoring of a pet's physical activity, and remote control and data access via a user-friendly application on a smartphone or other computing device.

In one embodiment, the system includes a food hopper, a dispensing mechanism, a weight sensor to measure dispensed food, an accelerometer or gyroscope to track pet movement, a camera for visual monitoring, and a microphone/speaker for two-way audio. A processor analyzes collected data to provide insights into the pet's health, such as calorie intake vs. expenditure, and can alert owners to unusual activity patterns or feeding irregularities. The system aims to improve pet health outcomes by ensuring proper nutrition and adequate exercise, while offering unparalleled convenience and peace of mind to pet owners.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWINGS

*   **FIG. 1** is a perspective view of the Smart Pet Feeding and Activity Monitoring System, illustrating its external components.
*   **FIG. 2** is a block diagram illustrating the internal components and data flow within the Smart Pet Feeding and Activity Monitoring System.
*   **FIG. 3** is a schematic diagram illustrating the interaction between

the Smart Pet Feeding and Activity Monitoring System and a remote computing device via a network.
*   **FIG. 4** is a flowchart illustrating a method for monitoring pet activity and adjusting feeding schedules based on collected data.
*   **FIG. 5** is a user interface screenshot of the mobile application for controlling and monitoring the system.

DETAILED DESCRIPTION OF THE INVENTION

The present invention provides a Smart Pet Feeding and Activity Monitoring System, generally referred to as system 100. As shown in FIG. 1, system 100 comprises a main housing 102, a food hopper 104, a food bowl 106, and a control panel 108.

Food Dispensing Mechanism: The food hopper 104 stores dry pet food. A dispensing mechanism 110 (shown schematically in FIG. 2), such as an auger or rotating paddle, is located at the base of the hopper 104. This mechanism 110 is controlled by a processor 120 to dispense precise portions of food into the food bowl 106. A weight sensor 112 (FIG. 2) is integrated beneath the food bowl 106 to accurately measure the amount of food dispensed and consumed, providing feedback to the processor 120.

Activity Monitoring: An activity sensor 114 (FIG. 2), such as a 3-axis accelerometer and/or gyroscope, is integrated within the housing 102 or, in an alternative embodiment, as a detachable collar-mounted unit that wirelessly communicates with the main housing 102. This sensor 114 continuously monitors the pet's movement, including walking, running, sleeping, and other activities. Data from the activity sensor 114 is transmitted to the processor 120.

Communication and Control: A communication module 116 (FIG. 2), such as a Wi-Fi or Bluetooth module, enables system 100 to connect to a local network and the internet. This allows for remote control and data transmission to a remote computing device 130 (FIG. 3), such as a smartphone or tablet, running a dedicated mobile application 132. The mobile application 132 allows pet owners to:
*   Set and modify feeding schedules and portion sizes.
*   Manually dispense food remotely.
*   View real-time and historical feeding data.
*   Monitor pet activity levels and trends.
*   Receive alerts for unusual activity, low food levels, or system malfunctions.

Processor and Data Analysis: The processor 120 (FIG. 2) is the central control unit of system 100. It receives data from the weight sensor 112 and activity sensor 114. The processor 120 executes algorithms to:
*   Manage feeding schedules and control the dispensing mechanism 110.
*   Calculate calorie intake based on dispensed food type and quantity.
*   Analyze activity data to estimate calorie expenditure and identify activity patterns (e.g., periods of high activity, prolonged inactivity).
*   Compare intake and expenditure to provide health insights and recommendations.
*   Generate alerts based on predefined thresholds (e.g., pet not eating, unusually low activity).

Optional Features:
*   **Camera 118 (FIG. 2):** An integrated camera allows pet owners to visually check on their pet remotely.
*   **Microphone and Speaker 119 (FIG. 2):** Enables two-way audio communication, allowing owners to speak to their pet and hear their pet's responses.
*   **Environmental Sensors:** Temperature and humidity sensors can be included to monitor the pet's environment.

Operation:
As illustrated in the flowchart of FIG. 4, the system continuously monitors pet activity (step 400). Feeding schedules are pre-programmed or remotely adjusted (step 402). At scheduled times, food is dispensed (step 404), and the amount is measured (step 406). Activity and feeding data are transmitted to the cloud and the mobile application (step 408). The processor analyzes this data (step 410) to provide insights and generate alerts (step 412), which are displayed on the mobile application (FIG. 5). Based on analysis, the system or owner can adjust feeding or activity goals (step 414).

The foregoing description of the embodiments has been provided for purposes of illustration and description. It is not intended to be exhaustive or to limit the invention. Individual elements or features of a particular embodiment are generally not limited to that particular embodiment, but are applicable to other embodiments where applicable.

CLAIMS

What is claimed is:

1. A smart pet feeding and activity monitoring system, comprising:
    a. a food hopper configured to store pet food;
    b. a dispensing mechanism coupled to the food hopper, configured to dispense pet food;
    c. a weight sensor positioned to measure an amount of dispensed pet food;
    d. an activity sensor configured to monitor physical activity of a pet;
    e. a communication module configured to transmit data to and receive commands from a remote computing device over a network; and
    f. a processor coupled to the dispensing mechanism, the weight sensor, the activity sensor, and the communication module, wherein the processor is configured to:
        i. control the dispensing mechanism to dispense pet food according to a schedule;
        ii. receive activity data from the activity sensor;
        iii. receive dispensed food data from the weight sensor; and
        iv. transmit the activity data and the dispensed food data to the remote computing device via the communication module.

2. The system of claim 1, wherein the processor is further configured to analyze the activity data and the dispensed food data to generate health insights for the pet.

3. The system of claim 2, wherein the health insights include a comparison of estimated calorie intake from dispensed food and estimated calorie expenditure based on pet activity.

4. The system of claim 1, further comprising a camera configured to capture images or video of the pet, wherein the processor is configured to transmit the images or video to the remote computing device.

5. The system of claim 1, further comprising a microphone and a speaker, wherein the processor is configured to enable two-way audio communication between the remote computing device and the pet.

6. The system of claim 1, wherein the activity sensor comprises at least one of an accelerometer or a gyroscope.

7. The system of claim 1, wherein the processor is further configured to generate an alert to the remote computing device if the pet's activity level deviates from a predefined threshold.

8. The system of claim 1, wherein the processor is further configured to receive commands from the remote computing device to manually initiate food dispensing.

9. A method for smart pet feeding and activity monitoring, comprising:
    a. storing pet food in a food hopper of an automated feeder;
    b. dispensing pet food from the food hopper according to a schedule via a dispensing mechanism;
    c. measuring an amount of dispensed pet food using a weight sensor;
    d. monitoring physical activity of a pet using an activity sensor;
    e. transmitting activity data from the activity sensor and dispensed food data from the weight sensor to a processor; and
    f. transmitting the activity data and the dispensed food data from the processor to a remote computing device over a network.

10. The method of claim 9, further comprising analyzing the activity data and the dispensed food data by the processor to generate health insights for the pet.

11. The method of claim 10, wherein analyzing includes comparing estimated calorie intake from dispensed food and estimated calorie expenditure based on pet activity.

12. The method of claim 9, further comprising capturing images or video of the pet using a camera and transmitting the images or video to the remote computing device.

13. The method of claim 9, further comprising enabling two-way audio communication between the remote computing device and the pet via a microphone and a speaker.

14. The method of claim 9, further comprising generating an alert to the remote computing device if the pet's activity level deviates from a predefined threshold.

15. The method of claim 9, further comprising receiving commands from the remote computing device to manually initiate food dispensing.

```
---

### Explanation of Patent Sections:

*   **Patent No. / Date of Patent:** The unique identifier and the date the
patent was granted.
*   **Title:** A concise name for the invention.
*   **Inventors:** The individuals who conceived the invention.
*   **Assignee:** The company or entity that owns the patent rights (often
the inventors' employer).
*   **Filing Date / Appl. No.:** When the application was submitted and its
unique number.
*   **International Class:** Classification codes that categorize the inven
tion based on its technical field.
*   **Abstract:** A brief summary of the invention, typically 150 words or
less, highlighting its key features and purpose.
*   **Background of the Invention:** Explains the problem the invention sol
ves, discusses existing solutions (prior art), and why the new invention is
needed.
*   **Summary of the Invention:** A more detailed overview of the inventio
n, often describing its advantages and how it overcomes the problems mentio
ned in the background.
*   **Brief Description of the Several Views of the Drawings:** A list and
short description of each figure in the accompanying drawings (which would
be separate pages in a real patent).
*   **Detailed Description of the Invention:** This is the "how-to" guide.
It describes the invention in full detail, enabling someone skilled in the
art to make and use it. It refers to the drawings and explains the componen
ts and their operation.
*   **Claims:** **This is the most important part of a patent.** The claims
legally define the scope of the invention for which protection is sought.
    *   **Independent Claims (e.g., Claim 1, Claim 9):** Broadest claims, s
tanding alone, defining the core inventive concept. They use "comprising" t
o indicate that the invention *includes* these elements but isn't limited t
o *only* these elements.
    *   **Dependent Claims (e.g., Claim 2, 3, 4...):** Refer back to an ind
ependent claim (or another dependent claim) and add further specific detail
s or limitations, making them narrower in scope.

This example illustrates the typical structure and content you'd find in a
utility patent. Remember, real patents are much more complex, with highly s
pecific legal language and extensive technical detail."""
          ),
        ],
        role='model'
      ),
      finish_reason=<FinishReason.STOP: 'STOP'>
    ),
  ],
  create_time=datetime.datetime(2026, 1, 18, 8, 57, 21, 74746, tzinfo=TzInf
o(0)),
  model_version='gemini-2.5-flash',
  response_id='caBsafrHBNqphMIP27iVqAI',
  sdk_http_response=HttpResponse(
    headers=<dict len=9>
```

```
    ),
    usage_metadata=GenerateContentResponseUsageMetadata(
      candidates_token_count=3176,
      candidates_tokens_details=[
        ModalityTokenCount(
          modality=<MediaModality.TEXT: 'TEXT'>,
          token_count=3176
        ),
      ],
      prompt_token_count=6,
      prompt_tokens_details=[
        ModalityTokenCount(
          modality=<MediaModality.TEXT: 'TEXT'>,
          token_count=6
        ),
      ],
      thoughts_token_count=1299,
      total_token_count=4481,
      traffic_type=<TrafficType.ON_DEMAND: 'ON_DEMAND'>
    )
  )
```

In [22]:
```
# Block model response that includes patent information (patent information

deidentify_with_replace_infotype(PROJECT_ID, response_patent.text, ["EMAIL_A
```

```
[Blocked due to category: Patent Related]
```

In [ ]:
```

```