In [11]:
```
import vertexai
from vertexai.generative_models import GenerativeModel, Part
import os
from dotenv import load_dotenv
load_dotenv(override=True)
import pprint
from google import genai
from google.genai import types
from IPython.display import Markdown
```

In [3]:
```
genai_client = genai.Client(
    vertexai=True, project=os.getenv("PROJECT_ID"), location='us-central1'
)
```

Readme from prommt

In [9]:
```
prompt = """
        You are a professional Readme file writer, write a readme.md file the ex
        The project aims to to provide a bunch of python notebook that interact
        this is a sample project that is based on the following documentation
        https://cloud.google.com/vertex-ai/generative-ai/docs/samples/generative

        Markdown:

"""

# Query the model
response = genai_client.models.generate_content(
    model='gemini-2.0-flash-exp',
    contents=[prompt],
    config=types.GenerateContentConfig(
        temperature=0
    )

)
```

In [12]:
```
display(Markdown(response.text))
```

Gemini AI API Python Notebook Examples

This repository contains a collection of Python notebooks demonstrating how to interact wit
h the Gemini generative AI API using Google Cloud's Vertex AI. These notebooks provide pr
actical examples for various use cases, allowing you to quickly get started with exploring the
capabilities of Gemini.

This project is based on the official Google Cloud documentation and samples found here:
[https://cloud.google.com/vertex-ai/generative-ai/docs/samples/generativeaionvertexai-ge
mini-pro-example](https://cloud.google.com/vertex-ai/generative-ai/docs/samples/generati
veaionvertexai-gemini-pro-example)

Project Overview

The notebooks in this repository showcase how to:

* **Set up your environment:** Configure your Google Cloud project and authentication fo
r accessing the Gemini API.
* **Generate text:** Use the Gemini Pro model to generate text based on prompts.
* **Explore different parameters:** Experiment with various parameters like temperature, t
op-p, and max output tokens to control the generation process.
* **Handle different input types:** Learn how to provide text prompts and potentially othe
r input types (depending on the specific notebook).
* **Understand the API response:** Interpret the API response and extract the generated t
ext.
* **Implement basic use cases:** Demonstrate practical applications of the Gemini API, suc
h as creative writing, summarization, and question answering.

Getting Started

To use these notebooks, you will need:

1. **A Google Cloud Platform (GCP) project:** If you don't have one, you can create a free a
ccount.
2. **Vertex AI API enabled:** Ensure the Vertex AI API is enabled for your GCP project.
3. **Python 3.7+:** Make sure you have Python 3.7 or a later version installed.
4. **Required Python libraries:** Install the necessary libraries using pip:

   ```bash
   pip install google-cloud-aiplatform
   pip install google-auth
   pip install ipykernel
   ```

5. **Authentication:** You will need to authenticate with your Google Cloud account. The r

ecommended way is to use Application Default Credentials (ADC). You can set this up by following the instructions in the Google Cloud documentation.

Notebook Structure

The repository is organized as follows:

* `notebooks/`: This directory contains the Jupyter Notebook files.
  * `gemini_pro_text_generation.ipynb`: A basic example of text generation using the Gemini Pro model.
  * `gemini_pro_parameter_exploration.ipynb`: A notebook that explores different parameters for text generation.
  * `gemini_pro_use_cases.ipynb`: A notebook that demonstrates various use cases for the Gemini API.
  * `...`: (Add more notebooks as you create them)

How to Use the Notebooks

1. **Clone the repository:**
   ```bash
   git clone https://github.com/your-username/your-repo-name.git
   cd your-repo-name
   ```
2. **Install the required libraries:**
   ```bash
   pip install -r requirements.txt
   ```
3. **Navigate to the `notebooks` directory:**
   ```bash
   cd notebooks
   ```
4. **Start Jupyter Notebook:**
   ```bash
   jupyter notebook
   ```
5. **Open and run the desired notebook:** Select the notebook you want to explore and execute the cells.

Contributing

Contributions are welcome! If you have any improvements, bug fixes, or new notebooks to add, please feel free to submit a pull request.

License

This project is licensed under the [MIT License](LICENSE).

Disclaimer

This project is intended for educational and demonstration purposes. Please refer to the offi cial Google Cloud documentation for the most up-to-date information and best practices.

Contact

If you have any questions or feedback, please feel free to reach out.

In [13]:
```
with open("readme.md", "w") as file:
    file.write(response.text)
```

Readme from existing file

In [ ]:
```
notebook_to_convert = "generate_readme_doc.ipynb"  # Replace with your notebook
output_folder = "output_pdfs" # Replace with the folder you want the pdf in

if not os.path.exists(output_folder):
    os.makedirs(output_folder)

convert_ipynb_to_pdf(notebook_to_convert, output_dir=output_folder)
# Convert to a different filename
#convert_ipynb_to_pdf(notebook_to_convert, output_dir=output_folder, output_file
```

```
Running nbconvert command:
jupyter nbconvert --to pdf --output output_pdfs\generate_readme_doc.pdf generate_
readme_doc.ipynb
Error during conversion: Command '['jupyter', 'nbconvert', '--to', 'pdf', '--outp
ut', 'output_pdfs\\generate_readme_doc.pdf', 'generate_readme_doc.ipynb']' return
ed non-zero exit status 1.
Stdout:
Stderr: [NbConvertApp] Converting notebook generate_readme_doc.ipynb to pdf
[NbConvertApp] Writing 22529 bytes to notebook.tex
[NbConvertApp] Building PDF
Traceback (most recent call last):
  File "<frozen runpy>", line 198, in _run_module_as_main
  File "<frozen runpy>", line 88, in _run_code
  File "c:\Users\User\gemini-examples\.venv\Scripts\jupyter-nbconvert.EXE\__main_
_.py", line 7, in <module>
  File "c:\Users\User\gemini-examples\.venv\Lib\site-packages\jupyter_core\applic
ation.py", line 283, in launch_instance
    super().launch_instance(argv=argv, **kwargs)
  File "c:\Users\User\gemini-examples\.venv\Lib\site-packages\traitlets\config\ap
plication.py", line 1075, in launch_instance
    app.start()
  File "c:\Users\User\gemini-examples\.venv\Lib\site-packages\nbconvert\nbconvert
app.py", line 420, in start
    self.convert_notebooks()
  File "c:\Users\User\gemini-examples\.venv\Lib\site-packages\nbconvert\nbconvert
app.py", line 597, in convert_notebooks
    self.convert_single_notebook(notebook_filename)
  File "c:\Users\User\gemini-examples\.venv\Lib\site-packages\nbconvert\nbconvert
app.py", line 563, in convert_single_notebook
    output, resources = self.export_single_notebook(
                        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "c:\Users\User\gemini-examples\.venv\Lib\site-packages\nbconvert\nbconvert
app.py", line 487, in export_single_notebook
    output, resources = self.exporter.from_filename(
                        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "c:\Users\User\gemini-examples\.venv\Lib\site-packages\nbconvert\exporters
\templateexporter.py", line 386, in from_filename
    return super().from_filename(filename, resources, **kw)  # type:ignore[return
-value]
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "c:\Users\User\gemini-examples\.venv\Lib\site-packages\nbconvert\exporters
\exporter.py", line 201, in from_filename
    return self.from_file(f, resources=resources, **kw)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "c:\Users\User\gemini-examples\.venv\Lib\site-packages\nbconvert\exporters
\templateexporter.py", line 392, in from_file
    return super().from_file(file_stream, resources, **kw)  # type:ignore[return-
value]
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "c:\Users\User\gemini-examples\.venv\Lib\site-packages\nbconvert\exporters
\exporter.py", line 220, in from_file
    return self.from_notebook_node(
           ^^^^^^^^^^^^^^^^^^^^^^^^
  File "c:\Users\User\gemini-examples\.venv\Lib\site-packages\nbconvert\exporters
\pdf.py", line 197, in from_notebook_node
    self.run_latex(tex_file)
  File "c:\Users\User\gemini-examples\.venv\Lib\site-packages\nbconvert\exporters
\pdf.py", line 166, in run_latex
    return self.run_command(
           ^^^^^^^^^^^^^^^^^
```

```
  File "c:\Users\User\gemini-examples\.venv\Lib\site-packages\nbconvert\exporters
\pdf.py", line 120, in run_command
    raise OSError(msg)
OSError: xelatex not found on PATH, if you have not installed xelatex you may nee
d to do so. Find further instructions at https://nbconvert.readthedocs.io/en/late
st/install.html#installing-tex.
```

Make sure LaTeX is installed on your system

In [14]:

```python
def pdf_to_bytes(pdf_path):
    """
    Reads a local PDF file and converts its contents into bytes.

    :param pdf_path: Path to the local PDF file.
    :return: Byte content of the PDF file.
    """
    with open(pdf_path, 'rb') as pdf_file:
        pdf_bytes = pdf_file.read()
    return pdf_bytes


prompt ="""
    Understand the Context:
    You will receive a PDF file that contains a Jupyter notebook.
    Your task is to create a README.md file that provides a clear and concise ex
    Key Sections of the README.md:
    Title: The name of the notebook.
    Description: A brief description of what the notebook does.
    Installation: Instructions on how to set up the environment to run the noteb
    Usage: Steps on how to execute the notebook.
    Contents: An overview of the main sections and functionalities within the no
    Examples: Provide examples of outputs or plots generated by the notebook (if
    Contributing: Guidelines for contributing to the notebook.
    License: Include a standard license section.
    Expected Output Format:
    The output should be in Markdown format.
    Use appropriate Markdown syntax for headers, lists, code blocks, etc.

    PDF Content Analysis:
    Extract the title from the first cell or heading of the notebook.
    Summarize each section of the notebook, including code cells, markdown cells
    Identify any dependencies or libraries used in the notebook.
    Note any specific instructions or parameters mentioned in the notebook.

    Example Output:

    # [Notebook Title]

    ## Description
    [Brief description of what the notebook does. Include the main objective and

    ## Installation
    To run this notebook, you will need to have the following libraries installe

    ```bash
    pip install [library1] [library2] [library3]
    ```

"""
```

```
bytes = pdf_to_bytes("inputs/generate_readme_doc.pdf")

# Query the model
response = genai_client.models.generate_content(
    model='gemini-2.0-flash-exp',
    contents=[
            prompt,
            types.Part.from_bytes(bytes)
        ],
    config=types.GenerateContentConfig(
        temperature=0
    )

)
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[14], line 53
     10     return pdf_bytes
     13 prompt ="""
     14     Understand the Context:
     15     You will receive a PDF file that contains a Jupyter notebook.
   (...)
     49
     50 """
---> 53 bytes = pdf_to_bytes("inputs/generate_readme_doc.pdf")
     55 # Query the model
     56 response = genai_client.models.generate_content(
     57     model='gemini-2.0-flash-exp',
     58     contents=[
   (...)
     65
     66 )

Cell In[14], line 8, in pdf_to_bytes(pdf_path)
      1 def pdf_to_bytes(pdf_path):
      2     """
      3     Reads a local PDF file and converts its contents into bytes.
      4
      5     :param pdf_path: Path to the local PDF file.
      6     :return: Byte content of the PDF file.
      7     """
----> 8     with open(pdf_path, 'rb') as pdf_file:
      9         pdf_bytes = pdf_file.read()
     10     return pdf_bytes

File c:\Users\User\gemini-examples\.venv\Lib\site-packages\IPython\core\interacti
veshell.py:324, in _modified_open(file, *args, **kwargs)
    317 if file in {0, 1, 2}:
    318     raise ValueError(
    319         f"IPython won't let you open fd={file} by default "
    320         "as it is likely to crash IPython. If you know what you are doin
g, "
    321         "you can use builtins' open."
    322     )
--> 324 return io_open(file, *args, **kwargs)

FileNotFoundError: [Errno 2] No such file or directory: 'inputs/generate_readme_d
oc.pdf'
```