edunet foundation

LAB MANUAL

Text Generation app via Gemini API Integration

Text Generation app via Gemini API Integration

Objective:

To create a Python program that takes a text input from the user and generates AI-based text output using the Gemini API. This practical will help in understanding how to integrate Gemini API with Python for text generation tasks.

Equipment Required:

- A computer with internet connectivity.
- Python installed (version 3.9 or later).
- Gemini API key.
- Code editor such as Visual Studio Code or any IDE of your choice.

Prerequisites:

- Basic knowledge of Python programming.
- Understanding of how to run commands in a terminal or command prompt.
- An active Gemini API key created from the developer console.

Problem Statement:

Create a console-based text generation application that:

1. Accepts a text prompt from the user.
2. Sends the prompt to the Gemini API.
3. Displays the generated text returned by the model.

Procedure:

Setting up the Environment:

1. **Ensure Python is installed**
   Check Python installation: python --version

If not installed, download and install Python from the official website:
https://www.python.org/downloads/

2. **Install the Gemini client library**
   In the terminal or command prompt, run: pip install -U google-generativeai
3. **Obtain an API key for Gemini**
   - Sign in to the Gemini developer console.
   - Create a new API key.
   - Copy the key safely; you will need it in your code.

Accessing Gemini API:

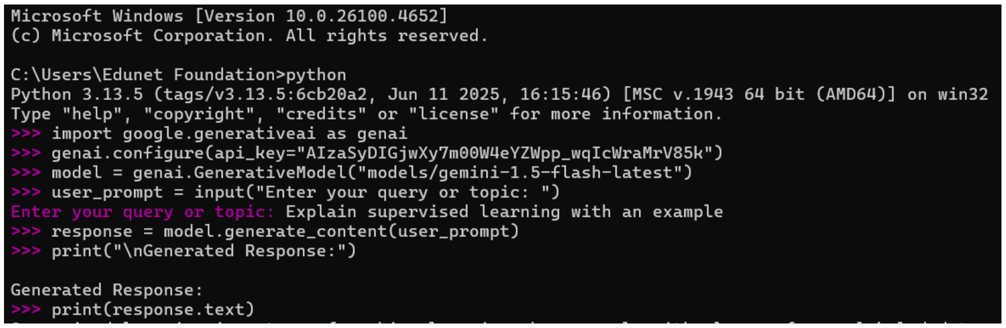
We will use Python with the official google-generativeai library to interact with Gemini. No third-party platforms are needed for this practical.

Interacting with Gemini API:

1. **Write the Python code in cmd *(Type one line at each time)***

```python
import google.generativeai as genai

# Step 1: Configure your API key
genai.configure(api_key="YOUR_API_KEY_HERE")

# Step 2: Create a model instance (flash model for higher quota)
model = genai.GenerativeModel("models/gemini-1.5-flash-latest")

# Step 3: Take user input
user_prompt = input("Enter your query or topic: ")

# Step 4: Generate AI text
response = model.generate_content(user_prompt)

# Step 5: Display the generated text
print("\nGenerated Response:")
print(response.text)
```

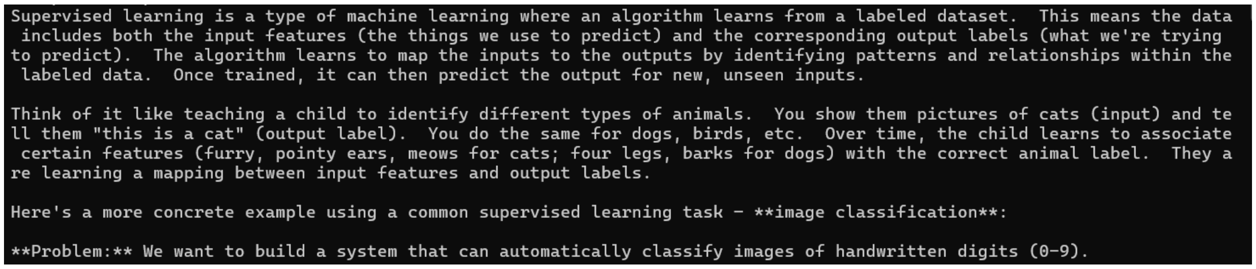
2. **Provide input**
   When prompted, type a query such as: Explain supervised learning with an example



```
Microsoft Windows [Version 10.0.26100.4652]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Edunet Foundation>python
Python 3.13.5 (tags/v3.13.5:6cb20a2, Jun 11 2025, 16:15:46) [MSC v.1943 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import google.generativeai as genai
>>> genai.configure(api_key="AIzaSyDIGjwXy7m00W4eYZWpp_wqIcWraMrV85k")
>>> model = genai.GenerativeModel("models/gemini-1.5-flash-latest")
>>> user_prompt = input("Enter your query or topic: ")
Enter your query or topic: Explain supervised learning with an example
>>> response = model.generate_content(user_prompt)
>>> print("\nGenerated Response:")

Generated Response:
>>> print(response.text)
```

Source: Screenshot

3. **View output**
   The generated text from Gemini will be displayed on the console.



```
Supervised learning is a type of machine learning where an algorithm learns from a labeled dataset.  This means the data
 includes both the input features (the things we use to predict) and the corresponding output labels (what we're trying
to predict).  The algorithm learns to map the inputs to the outputs by identifying patterns and relationships within the
 labeled data.  Once trained, it can then predict the output for new, unseen inputs.

Think of it like teaching a child to identify different types of animals.  You show them pictures of cats (input) and te
ll them "this is a cat" (output label).  You do the same for dogs, birds, etc.  Over time, the child learns to associate
 certain features (furry, pointy ears, meows for cats; four legs, barks for dogs) with the correct animal label.  They a
re learning a mapping between input features and output labels.

Here's a more concrete example using a common supervised learning task - **image classification**:

**Problem:** We want to build a system that can automatically classify images of handwritten digits (0-9).
```

Source: Screenshot

This completes the text generation app integration using Gemini API.