Karthik Project: Gemini-powered Text Generation

1. Project Overview

This project utilizes Google Gemini's generative language API to provide text generation capabilities. It takes user prompts as input and returns generated text based on the prompt. The primary purpose is to demonstrate a simple integration with the Gemini API and showcase its text generation functionality. The target users are developers interested in exploring and utilizing the Google Gemini API.

2. Tech Stack

- **Programming Language:** Python
- **Framework:** Flask
- **Libraries:** `requests`, `flask_cors`, `dotenv`
- **API:** Google Gemini Generative Language API

3. Installation & Setup

Prerequisites:

1. **Python 3.7+:** Ensure Python is installed on your system.
2. **pip:** Python's package installer. (Usually installed with Python)
3. **Google Cloud Project & API Key:** You need a Google Cloud project with the Gemini API enabled and an API key generated. This key will be used as an environment variable.

Installation Steps:

1. **Clone the repository:**

   ```
   git clone <repository_url>
   ```

2. **Create a `.env` file:** Create a file named `.env` in the project's root directory and add your Gemini API key:

   ```
   GEMINI_API_KEY=YOUR_GEMINI_API_KEY
   ```

3. **Install dependencies:**

   ```
   pip install -r requirements.txt
   ```

   (A `requirements.txt` file should be created containing the listed libraries.)

4. Usage Guide

Running the application:

1. **Development:**

   ```
   python app.py
   ```

   This will start the Flask development server.

2. **Production:** (Adapt as needed for your production environment)

   ```
   gunicorn app:app
   ```

Making API Calls:

The application exposes a single endpoint: `/generate`. Send a POST request with a JSON payload containing the prompt:

```
curl -X POST -H "Content-Type: application/json" -d '{"prompt": "Write a short story about a robot learning to love."}'
http://localhost:5000/generate
```

The response will be a JSON object with the generated text:

```
{
  "text": "Once upon a time, in a world filled with gleaming chrome and whirring gears, lived a robot named Rusty.  ..."
}
```

5. Features

- **Text Generation:** Generates text based on user-provided prompts using the Google Gemini API.
- **Error Handling:** Includes basic retry logic for API calls.
- **Environment Variable Support:** Securely stores the API key using environment variables.
- **CORS Enabled:** Allows cross-origin requests for easier integration with front-end applications.

6. Codebase Walkthrough

Folder Structure:

- **`app.py`:** The main application file. Contains the Flask application, API endpoint, and Gemini API interaction logic.

Important Files:

- **`app.py`:** This file is the core of the application. It sets up the Flask application, handles incoming requests, calls the Google Gemini API, and returns the generated text. It also loads the API key from the `.env` file.

7. API Documentation

Endpoint: `/generate`

Method: POST

Request:

```
{
  "prompt": "Your text prompt here"
}
```

Response (Success):

```
{
  "text": "Generated text from Gemini API"
}
```

Response (Error): The response will contain an error message if the API call fails or the input is invalid. Appropriate HTTP status codes will be used to indicate errors (e.g., 400 Bad Request, 500 Internal Server Error).

8. Contribution Guide

1. **Fork the repository.**
2. **Create a new branch:** `git checkout -b my-new-feature`
3. **Make your changes.**
4. **Commit your changes:** `git commit -m "Add my new feature"`
5. **Push your branch:** `git push origin my-new-feature`
6. **Create a pull request.**

Coding Style: Follow PEP 8 style guidelines for Python code.

9. License & Credits

This project is currently unlicensed. Please add a license if you intend to open source it.

Credits: Google for providing the Gemini API.