
AI Text Generation Service with Google Gemini API

This project uses the Google Gemini API to generate summaries from text. It provides a service that accepts a text prompt, sends it to the API, and returns the generated text response.

Table of Contents

Overview

This project implements a simple web service that leverages the Google Gemini API to generate text based on a given prompt. It allows users to submit a request with a prompt, get the text summary, and display the result.

We deploy this application on an AWS EC2 instance to make it publicly accessible.

Prerequisites

Before getting started, ensure you have the following:

- **Node.js** installed on your local machine (version 14 or higher).
- **Google Cloud Account** and access to the Gemini API.
- **AWS Account** for deploying the application to EC2.
- **AWS CLI** installed and configured on your local machine.

Tools and Services

1. **Google Cloud API (Gemini)**: API key to access the Gemini model.
2. **Node.js**: Server-side JavaScript runtime.
3. **Express.js**: Web framework for Node.js.
4. **AWS EC2**: Cloud computing platform to host the service.
5. **Postman** or **curl** for testing the API.

Setup Instructions

1. Clone the Repository

Start by cloning the project repository to your local machine.

git clone https://github.com/Rachit2912 /BlogOn

cd BlogOn

2. Install Dependencies

Run the following command to install all necessary dependencies:

npm install @google/generative-ai express dotenv cors body-parser

3. Configure Google Cloud API Key

You need to get the API key for the Google Gemini service.

1. Go to the Google Cloud Console.
2. Enable the "Google Gemini API" for your project.
3. Generate an API key from the credentials section.

Create a .env file in the root of the project and add the API key like this:

GOOGLE_API_KEY=YOUR_API_KEY

Make sure to replace YOUR_API_KEY with the actual key you generated.

4. Run the Local Server

To test the service locally, run the following command:

npm start

This will start a local Express server on http://localhost:5000.

And in another terminal use :
cd frontend

npm start

This will start a react app on http://localhost:3000

Deployment Process

1. Create an EC2 Instance on AWS

1. Log in to your AWS Console.
2. Go to EC2 dashboard and create a new instance.
3. Choose an **Amazon Linux 2** or **Ubuntu** instance type.
4. Select a security group that allows **HTTP (port 80)**, **SSH (port 22)**, and **HTTPS (port 443)**.
5. Once the instance is created, connect to it via SSH:
6. ssh -i "your-key.pem" ec2-user@your-ec2-public-ip

2. Set Up the EC2 Instance

1. Install Node.js on EC2

Install Node.js by running the following commands on your EC2 instance:

```
# Update package index
sudo yum update -y

# Install Node.js
curl -sL https://rpm.nodesource.com/setup_16.x | sudo -E bash -
sudo yum install -y nodejs
```

2. Install Git and Clone the Repository

```
sudo yum install git -y
git clone https://github.com/your-username/ai-text-generation-service.git
cd ai-text-generation-service
```

3. Install Project Dependencies

On the EC2 instance, run the following:

```
npm install
```

4. Set Up the Google API Key on EC2

You can either add your API key directly to the .env file on the EC2 instance, or set it up as an environment variable:

```
echo "GOOGLE_API_KEY=YOUR_API_KEY" >> .env
```

Replace YOUR_API_KEY with your actual key.

5. Run the Application

Start the application on your EC2 instance:

```
npm start
```

The server will be available at the public IP of your EC2 instance (e.g., http://<your-ec2-public-ip>:3000).

---

3. Set Up Reverse Proxy (Optional)

If you want the application to run on port 80 (default HTTP port), use Nginx as a reverse proxy.

1. Install Nginx

```
sudo yum install nginx -y
```

2. Configure Nginx

Edit the Nginx configuration to route traffic to your Node.js application.

sudo vi /etc/nginx/nginx.conf

Add a server block to route traffic:

```
server {
    listen 80;
    server_name <your-ec2-public-ip>;

    location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

3. Start Nginx

sudo service nginx start

Architecture Overview

The architecture of the application is as follows:

1. Client Request

- A user sends an HTTP request with a text prompt to the application hosted on AWS EC2.
- The server runs Express.js and listens for incoming requests.

2. Backend API Call (Google Gemini)

- The server sends the prompt to the **Google Gemini API** using a **POST request**.
- The API processes the request and returns a generated text response.

3. Response Delivery

- The backend receives the response from the Gemini API and sends it back to the client.
- The generated text is displayed to the user as the final result.

4. Hosting on AWS EC2

- The application is deployed on an **AWS EC2 instance**.
- Nginx acts as a reverse proxy, forwarding HTTP requests to the Node.js application running on port 3000.

Testing the API

Once deployed, you can test the service by sending a request to the server. You can either use Postman or curl to send a **POST request**:

Example using curl:

```
curl -X POST \
  http://<your-ec2-public-ip>:80/generate-summary \
  -H "Content-Type: application/json" \
  -d '{"prompt": "Explain how AI works"}'
```

The API should respond with a generated text summary.

Conclusion

This project demonstrates how to create a simple AI-based text generation service using the Google Gemini API. The service is deployed to AWS EC2 and accessible publicly, enabling users to generate summaries based on text prompts.

Let me know if you need any further modifications or clarifications in the documentation!