

GenAI Powered Daily Quotes Blog Generator

Introducing the Idea

Overview

This project was developed as an idea entry shortlisted in for [Project Saadhna](#). This is my first time attending Project Saadhna.

About Project Saadhna

To empower selected Code Vipassana developers to lead and continuously engage in self-paced building with Google Cloud and leading more developers in the community to learn, build and grow.

"Saadhna" is a Sanskrit word which means to practice what you have learnt. Similarly, in this program, the activated developers from Code Vipassana seasons will practice their hands-on learning by continuously building with Google Cloud.

Goal

To provide users with a fresh, AI-generated poem and accompanying visual every day, inspired by a daily quote.

Technology Stack

Google Cloud Platform (GCP): Google Cloud, Firebase, Gemini, Imagen2 etc

Any Compatible CMS of choice.

Generative AI Models: For image and poem generation.

Daily Quote API : (Free versions or Free Tiers service)

Table of Contents

- [Description of Post](#)
- [Progress of Project till now.](#)
- [Configure Project](#)
 - [Starting a GCP Project & Enable Billing](#)
 - [Create a Firebase Project](#)
- [Understand the Project Workflow](#)
- [Set Up Project](#)
 - [Download the Project zip file](#)
 - [Activate GCP Cloud Shell](#)
 - [Create and Set up a Virtual Environment.](#)
 - [Start a new Pelican Project](#)
 - [Set Up Firebase Hosted Web App](#)
- [Execution and Output Screenshot](#)
- [Understand the Capacity and Limits of Package](#)
- [Final Product](#)
 - [Final Project Diagram](#)
- [Potential Project Improvements](#)
- [Updates : Additional Changes Made](#)
- [Disclaimers from Developer.](#)
- [API & Package Documentations](#)

Description of Post

Each post will contain:

Date: Day, Month, Year

Quote of the Day: Sourced from a selected Daily Quote API

Poem of the Day: AI-generated using Gemini Vertex AI, inspired by the quote

Cover Image: AI-generated using Gemini Vertex AI, inspired by the quote

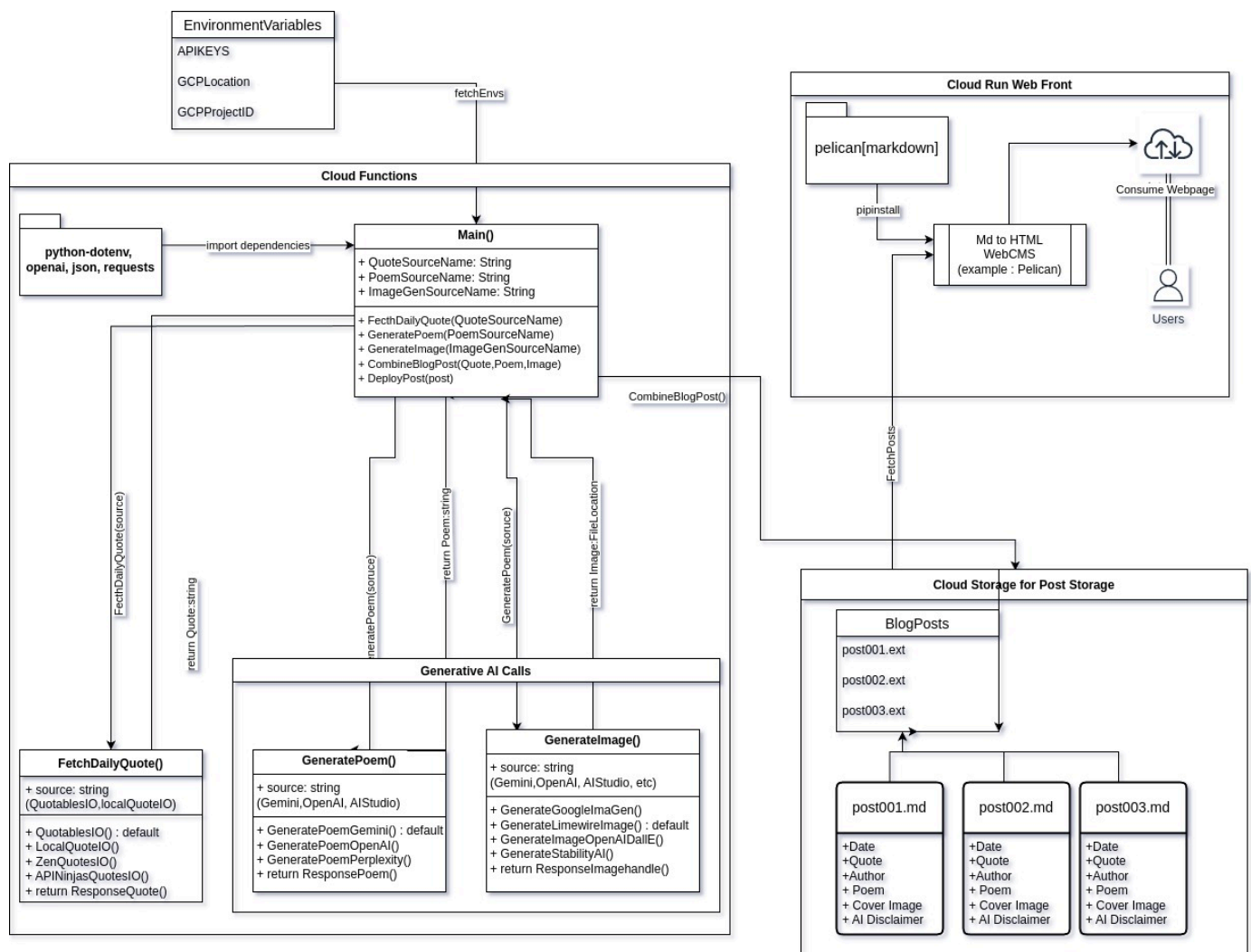
Disclaimers: Legal and AI-related disclaimers

The project's backend could be powered by Google Cloud Functions, which will execute daily.

The program will:

1. Fetch the quote of the day from any Daily Quote API (or other services).
2. Generate a cover image using Gemini AI (or other services) based on the quote.
3. Generate a poem using Gemini Vertex AI (or other services) based on the quote.
4. Create a blog post with the quote, poem, image, and date.
5. convert it to web consumable format (html,css,js).
6. Deploy the blog post to Hosting of choice. (Here we will use Firebase Hosting).
- 7.

Once I had most of my Individual parts of the project working. I started bringing it together.



This is what I had in mind.

Some aspects of the workflow has changed throughout the progress of this

Progress of Project till now.

Version V0.1 : Starting Code

- HTML/CSS/JS from Scratch.
- post.js to modify index.html with {postname}.md files on load.
- No WebServer or Hosting Configured yet.

Version V1.0 : Getting Each Module Working

- Basic Individual Python Scripts to
 - Get Quote
 - Generate Poem
 - Generate Image
 - Generate MD File
 - Basic MDfile to HTML convertor directory for Pelican.
- Pelican output to Cloud Storage Mover Script.
- Hosting on Google Cloud Storage.

Version V2.0 : Single Entry Point Script for Terminal. app.py

- Switched to Firebase as Hosting.
- Included Extensibility options for each Step to use different services. Some are under progress.
- Combined Application for running {python app.py}
- Scripted Pelican creation, rebuilds on new run, deploying to Firebase with app.py
- Included Logs to audit app run and Errors.

CURRENT VERSION

Version V2.1 : Text User Interface. appTUI.py

- Introduced a very good looking Text User Interface 'appTUI.py' to make informed execution in each step.
 - Asks which service to use for Quote, Poem, Image
 - Reminds to set API_KEY
 - Enabled VertexAI only for Imagen tasks
 - Asks confirmations before running pelican export and firebase deployment.
 - Parameterised 'app.py' Code
 - merged functionalities of **appTUI.py** and Parameterized **app.py** into a unified **main.py**
 - Further reduced error points with Exception handling.
-

Version V3.0 : Build with App Engine / Cloud Run

- Bring Better TUI Features such as full screen item selection, progress bar, cron setup manager.
- Port out and modify project to work as an App Engine Code or Containerize with Cloud Run .
- Add Cloud Scheduler to Automate Task after porting to AppEngine/CloudRun. Cloud Scheduler cannot trigger code in Cloudshell.
- Fine-tune the Poem Generation / Cover Image Generation prompt to be cohesive with the quote. Currently, there is a lack of cohesion between what the quote says and what the model interprets as the essence of the quote.

Current project I have is made to work from within Cloud Shell and deployed to Firebase. I have never used Pelican before, so I could not get the project running in App Engine initially, so, I settled for Cloud Shell. I want to port this out into a cloud native solution such as App Engine with the opportunity to automate with Cloud Scheduler. I skipped trying to meet this within the program deadline as I reached Cloud Shell Quota after spending 50 Hour Weekly Cloud Shell Usage Quota Limit on my billing enabled account and had to wait for 2 days to reset quota. Remaining developments were done in an alternate account without billing. To Be Honest, this is the first time I found out Cloud Shell has a weekly quota, so that's one more additional knowledge I gained this week. Will work on improvements after Quota Resets.

Cloud Shell quota

Cloud Shell has weekly usage limits. If you reach these limits you will need to wait before you can use Cloud Shell again.

50 of 50 hours used

Your quota resets on Jul 28, 2024, 7:21 pm

You are approaching your usage limits. If you believe that this is an error, please [contact support](#).

CLOSE

Please Refer Final Pages of PDF to See Any Updates in this

Configure Project

Starting a GCP Project & Enable Billing

Create a Google Cloud Project & Firebase Project

Decide on a Project ID for your project. This must be Globally-unique.

I have created the project with id **project-saadhna-blog-generator** and name **Project-Saadhna**

This will be referred throughout the codelab as '**GCP_PROJECT_ID**'

I will be creating a Firebase project in the same project id.

Create a Firebase Project

Note : You should remember that when you use Add Firebase to Cloud Project, this step is non reversible. Google Cloud, User Roles, Permissions and billing for your project will be shared. If you delete your Firebase Project, it also deletes your Google Cloud Project Too. You can modify the code to use a separate firebase project but this is not implemented for the current project.

Go to <https://console.firebase.google.com/>.

Click **Create a Project**

Click on "**Add Firebase to Cloud Project**" Button below the Text Area.

If asked for Analytics, you can enable or disable it. I will be disabling it.

Connect firebase to the already created GCP Project.

Click **Continue**

If you have enabled billing, your project will ask you to select Blaze Plan. click Confirm Plan

See [Pricing Page](#) for details of Sparks Plan vs Blaze Plan.

About using Cloud Shell

Next steps, you can either perform on your local System, GCP VM Instance or Cloud Shell. I will be using Cloud Shell. Some commands in this project are based on Cloud Shell.

Your 5GB home directory will persist across sessions, but the VM is ephemeral and will be reset approximately 20 minutes after your session ends. No system-wide change will persist beyond that. If you delete your \$HOME directory, or if you don't access Cloud Shell for 120 days, Cloud Shell automatically deletes your \$HOME directory and you can't recover your files. There is also a 50 hour per week Quota Limit to use it which i did not know before this. Ideally, if you want to host this project on a long term basis, you should consider performing this on a VM Instance, App Engine, or Cloud Run. Since this is a demo app, I will be showcasing it in Cloud Shell.

Advantages of Cloud Shell includes fast easy access, no cost processing, Built-In Cloud Code Editor. Access to Gemini Code Assist and pre-installed packages.

Disadvantages include restrictions on persisting changes through session, limitations on setting customization. Persisting changes through terminal reload.

Understand the Project Workflow

This project has several steps in which you can opt for the service of your choosing. Some are not yet optimized or tested.

The Project is split into several key milestones :

1. Start and Set Up.
2. **Get Quote of the Day** from API Services. *(We can use any of these 3)*
 1. **Zen Quotes API** (zenquotes.io/api/random)
No Registration needed for using this lab.
 2. **Quotable** free, open source REST API (api.quotable.io/random)
No Registration needed for using this lab.
 3. Collection of **Local Quotes** incase of Failure (~/.Backup_Repository/local_quotes.json)
Limited number of quotes just for testing/failover.
3. **Generate 12 line Poem** based on Quote of the Day. *(We will use Gemini which is tested and working)*
 1. Generate Poem using **Google Gemini 1.0 Pro** Call (models/gemini-1.0-pro)
We will use the free Gemini AI Studio AI Key for this.
aistudio.google.com "Get API Key"
 2. Generate Poem using OpenAI Chat Completion (api.openai.com/v1/completions)
This requires you to have a Paid Plan to use OpenAI.
 3. Generate Poem using Perplexity AI Calls (api.perplexity.ai)
This requires you to have a Paid Plan to use Perplexity.
4. **Generate a Cover Image** based on Quote of the Day. *(We will use Limewire Tree Tier)*
 1. Generate Image using **Limewire API** (limited to 10 Credits per day)
This requires you to register on limewire.com
limewire.com/u/subscriptions/api?tier=API_FREE
 2. Generate Image using Google Imagen 2 Model. (Available on request but closed to general public calling.) This requires you to be in an allowlist by Google Imagen Team. I got my application accepted to do experimentation and I got the Imagen Image Generation working. So the code will work if you are in their Allowlist. Requires Enabling Vertex AI API and has Billing Enabled.
 3. Generate Image using StabilityAI StableDiffusion Model. (limited trial)
This requires you to have an API Key from StabilityAI.
limited to 25 Free Credit in total.
5. Use Pelican **Markdown to HTML** package.
 1. Combine the Quote,Poem & Cover to a Markdown file.
 2. Use Pelican Python Framework to generate Static Websites.
 3. Jekyll, Hugo, Ghost, Nikola, Django etc are other options to consider.

6. Host your **Static Website Blog**.

1. Use Firebase Hosting to deploy your website.

Set Up Project

Download the Project zip file

[Github Repo](#)

Activate GCP Cloud Shell

Upload the Zip file to GCP Cloud Shell in a folder. (folder name "PS/GeminiBlog-Version:2.0" in my case.) Unzip the project to the folder

```
unzip GeminiBlog.zip -d GeminiBlog
```

Get into the project folder.

```
cd GeminiBlog
```

Create and Set up a Virtual Environment.

```
sudo apt update
sudo apt install python3-venv python3-pip
python3 -m venv GPBenv
source GPBenv/bin/activate
```

We should consider adding an Environment Set-Up file to avoid having to set Path variables on each session. We only have to deal with this in a cloud shell as it is an ephemeral VM.

```
echo $PATH
```

Refer : https://cloud.google.com/shell/docs/configuring-cloud-shell#environment_customization

But I could not get that working to append the \$PATH variable. I have worked a way around it in script later on.

Install dependencies

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

Create some environment variables based on which services you are using and which services you have access to.

If the directory contains a .env file, run 'edit .env' to modify the file and add API KEY. Enter API keys for whichever services you plan on using later.

```
edit .env
```

If you don't have a .env inside GeminiBlog folder, run :

While the pip package dotenv is imported, it does not work randomly on different systems. I found that a secret manager is a better option to save variables. You should opt for some secret manager when using it in production. But for now, we'll use a workaround and install python-dotenv.

```
touch .env
echo "# Poem generation using OpenAI" >> .env
echo "OPENAI_API_KEY=" >> .env
echo "# Poem generation using PerplexityAI" >> .env
echo "PERPLEXITY_API_KEY=" >> .env
echo "# Poem generation using Gemini AI Studio" >> .env
echo "GEMINI_API_KEY=" >> .env
echo "# Image generation using StabilityAI" >> .env
echo "StabilityAI_API_KEY=" >> .env
echo "# Image generation using Google Imagen2" >> .env
echo "GOOGLE_API_KEY=" >> .env
echo "# Image generation using LimeWire" >> .env
echo "LimeWire_API_KEY=" >> .env
echo "" >> .env
echo "# Used as google cloud project id and firebase project id" >> .env
echo "GCP_PROJECT_ID=" >> .env
```

Make sure you do not share or any other config files when you export or upload your project publicly.

```
touch .gitignore
echo ".env" >> .gitignore
echo "firebase.json" >> .gitignore
echo "__pycache__" >> .gitignore
echo "*.log" >> .gitignore
echo "**/*.*" >> .gitignore
echo "**/node_modules/**" >> .gitignore
```

Start a new Pelican Project

Go into the pelican-CMS folder and start a Pelican Project. If the pelican-CMS folder is not present, run 'mkdir pelican-CMS' to create one.

Note : During every folder name creation step in this walk-through, please follow the same name used here. Some of these are called by the same name in the project script.

```
cd pelican-CMS
```

```
pelican-quickstart
```

You should see a setup guide starting with "Welcome to pelican-quickstart..."

In some cases, you get an error. *Command not Found*. This is because you are having issues with \$PATH. see if your \$PATH contains a local bin as an entry. If not, run the following script when you reset your session or virtual environment. replace with your Cloud Shell User Name. This might be easily resolvable somehow. I just didn't see a reliable way to prevent it.

```
export PATH=$PATH:/home/{yourcloudshellusername}/.local/bin/
```

Run again and Answer the Quick-Start Set-Up Questions.

```
pelican-quickstart
> Where do you want to create your new web site? [.] _site
> What will be the title of this web site? Gemini Powered Blog
> Who will be the author of this web site? < your name >
> What will be the default language of this web site? [en]
> Do you want to specify a URL prefix? e.g., https://example.com (Y/n) Y
> What is your URL prefix? (see above example; no trailing slash) <replace
with your gcpid>.web.app
> Do you want to enable article pagination? (Y/n) Y
> How many articles per page do you want? [10] 10
> What is your time zone? [Europe/Rome] Asia/Calcutta
> Do you want to generate a tasks.py/Makefile to automate generation and
publishing? (Y/n) Y
> Do you want to upload your website using FTP? (y/N) N
> Do you want to upload your website using SSH? (y/N) N
> Do you want to upload your website using Dropbox? (y/N) N
> Do you want to upload your website using S3? (y/N) N
> Do you want to upload your website using Rackspace Cloud Files? (y/N) N
> Do you want to upload your website using GitHub Pages? (y/N) N
```

You will see the following message

Your new project is available at /home/{CloudShellUserName}/GeminiBlog-Version:2.0/pelican-CMS/_site

Inside _site folder, open "pelicanconf.py".

Modify/Add the following at the bottom and save it.

```
RELATIVE_URLS = True
STATIC_PATHS = ['media']
```

```
cd _site
pelican content
```

If you see this message, we are good to proceed :

Done: Processed 0 articles, 0 drafts, 0 hidden articles, 0 pages, 0 hidden pages and 0 draft pages in 0.06 seconds.

Pelican is meant to run a server on Port 8000 by default but since we are using firebase hosted static website method, we will not cover it here or use that method. However, if you are interested in seeing how it is, run this.

START USING

```
pelican --listen --port 8000
```

Click on serving URL to view port 8000

END USING

```
Ctrl + Z
```

Feel free to go through Pelican Documentation later on to see further Themes, Customization, Automation, Serving, Plugin options. docs.getpelican.com/en/latest

You can also use Google Cloud Storage as your hosting repo but it requires you to set up things, have a registered verified domain and DNS propagation also takes time. See this post by Priyanka Vergadia (The Cloud Girl) if you are interested. [Hosting a static website on Google Cloud using Google Cloud Storage](#)

We will now proceed to set-up firebase hosting.

Set Up Firebase Hosted Web App

While staying inside `_site`

Initialize Firebase

```
firebase login --no-localhost
```

Complete the Authorization Procedure if this is your first time using Firebase in Cloud Shell.
After authorization, you should see this message :

```
✓ Success! Logged in as {email id}
```

```
firebase init hosting
```

- Use an Existing Project
- Select your created Project
- Answer Questions

```
What do you want to use as your public directory? **output**
Configure as a single-page-app? **No**
Set Up Automatic Builds and Deploys with Github? **No**
```

(ignore 404 for now.)

File output/index.html already exists. Overwrite? **No**

In Code Editor Side Menu

open **firebase.json** that was just created and set the following. Some of these files would not be present in the output folder but I just keep mentioning them just in case of code change.

```
{
  "hosting": {
    "public": "output",
    "ignore": ["firebase.json", "**/.*", "**/node_modules/**", ".env",
    "__pycache__"],
    "rewrites": [
      {
        "source": "**",
        "destination": "/index.html"
      }
    ]
  }
}
```

Deploy Firebase

firebase serve # To see site locally before deployment

firebase **deploy** # To deploy to the firebase project created.

Run Project

Make sure you have added environment variables in the .env file.

go back to main project Directory

```
cd ..
cd ..
# you should be in the same directory with main.py
```

Now, before you run app

1. Remember to set the PATH variable in case you exited the Virtual Environment and Restarted it. (only for Cloud Shell).
2. Set the API Keys for the services you wish to use in the .env file in the project directory. Also save GCP_PROJECT_ID in .env. ('edit .env' to modify file)

Execution and Output Screenshot

Proceed to run the app

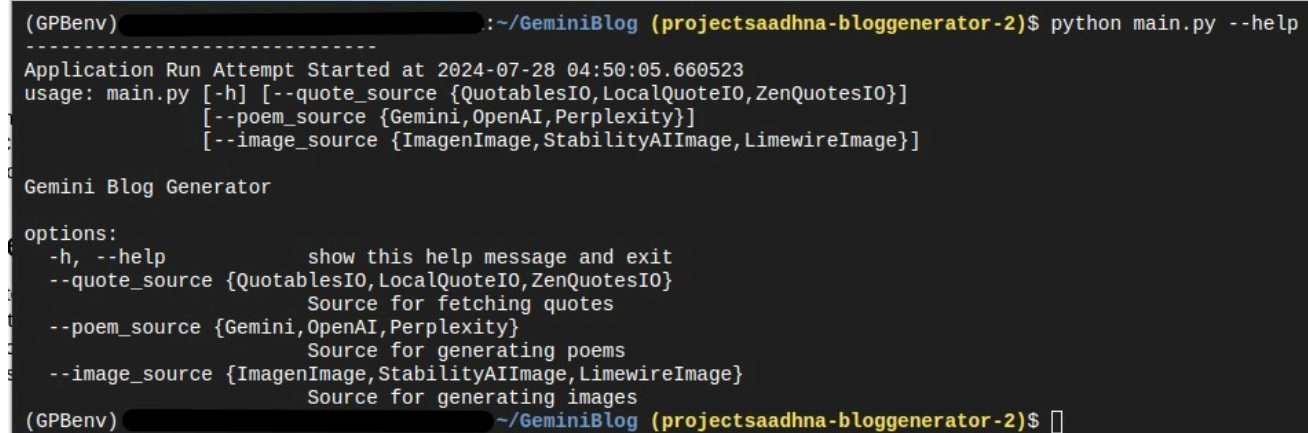
A properly executed commands will look like this

run **main.py --help** to see how to use the script.

1 : Get Help

```
python main.py --help
```

Output from Feature 1

A screenshot of a terminal window showing the output of the command 'python main.py --help'. The terminal has a dark background with light-colored text. The output includes the application name 'Gemini Blog Generator', the start time '2024-07-28 04:50:05.660523', and a usage section with options for quote_source, poem_source, and image_source. The prompt is '(GPBenv) ~/GeminiBlog (projectsaadhna-bloggenerator-2)\$'.

```
(GPBenv) ~/GeminiBlog (projectsaadhna-bloggenerator-2)$ python main.py --help
-----
Application Run Attempt Started at 2024-07-28 04:50:05.660523
usage: main.py [-h] [--quote_source {QuotablesIO,LocalQuoteIO,ZenQuotesIO}]
               [--poem_source {Gemini,OpenAI,Perplexity}]
               [--image_source {ImagenImage,StabilityAIImage,LimewireImage}]

Gemini Blog Generator

options:
  -h, --help            show this help message and exit
  --quote_source {QuotablesIO,LocalQuoteIO,ZenQuotesIO}
                        Source for fetching quotes
  --poem_source {Gemini,OpenAI,Perplexity}
                        Source for generating poems
  --image_source {ImagenImage,StabilityAIImage,LimewireImage}
                        Source for generating images
(GPBenv) ~/GeminiBlog (projectsaadhna-bloggenerator-2)$
```

2 : Run with parameters

```
python main.py --quote_source QuotablesIO --poem_source Gemini --
image_source ImagenImage
```

Output from Feature 2

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
(bash - GeminiBlog + v [ ] [ ] ... v x)

(GPEnv) ~/GeminiBlog (projectsaadhna-blogger-2)$ python main.py --quote_source QuotablesIO --poem_source Gemini --image_source LimewireImage

Application Run Attempt Started at 2024-07-28 00:50:29.583752
Setting Up Dire 'ory'
-----
Setting User Home Directory: /home/
Setting Pelican Content Directory: /home/
Setting Pelican Attached Media Directory: /home/
Setting Pelican Deployer Directory: /home/
Adding ./local/bin to PATH if not already present
Arguments Detected with main.py Running the blog generation process in script mode.
-----
Fetching Daily Quote
-----
Generating Poem
-----
Generating Image
Image Generation - Success!
You have 4.0 credits remaining.
Image downloaded and saved to: /home/
-----
Generating Markdown File
-----
Pelican Deployer Directory: /home/
Running Pelican MakeFile
"pelican" "/home/
Done: Processed 5 articles, 0 drafts, 0 hidden articles, 0 pages, 0 hidden pages and 0 draft pages in 0.18 seconds.
Blog Post Generation Complete!
-----
Deploy Pelican output to Firebase.

=== Deploying to 'projectsaadhna-blogger-2'...

i deploying hosting
i hosting[projectsaadhna-blogger-2]: beginning deploy...
i hosting[projectsaadhna-blogger-2]: found 111 files in output
✓ hosting[projectsaadhna-blogger-2]: file upload complete
i hosting[projectsaadhna-blogger-2]: finalizing version...
✓ hosting[projectsaadhna-blogger-2]: version finalized
i hosting[projectsaadhna-blogger-2]: releasing new version...
✓ hosting[projectsaadhna-blogger-2]: release complete

✓ Deploy complete!

Project Console: https://console.firebase.google.com/project/projectsaadhna-blogger-2/overview
Hosting URL: https://projectsaadhna-blogger-2.web.app
Blog post should be generated and deployed successfully!
-----
Application Run Attempt Ended at 2024-07-28 00:50:52.679465
Total Time Taken: 0:00:23.095713
-----
```

Method 3 : Run without Parameters for working with Text User Interface

python main.py

Output from Feature 3

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
(bash - GeminiBlog + v [ ] [ ] ... v x)

(GPEnv) ~/GeminiBlog (projectsaadhna-blogger-2)$ python main.py

Application Run Attempt Started at 2024-07-28 00:33:27.928328
Setting Up Directory
-----
Setting User Home Directory: /home/
Setting Pelican Content Directory: /home/
Setting Pelican Attached Media Directory: /home/
Setting Pelican Deployer Directory: /home/
Adding ./local/bin to PATH if not already present
No Parameters Detected. Running the blog generation process in TUI mode.

Welcome to Gemini Blog Generator!

Set Quote, Poem, Image Generation Sources.
Select your Source :
Select From Available Quote Sources:
[1] QuotablesIO [ Working ][ No API_KEY Needed]
[2] LocalQuoteIO [ Working ][ Use from 'Backup_Repository/local_quotes.json']
[3] ZenQuotesIO [ Working ][ No API_KEY Needed]
Enter your choice: 3
-- you selected ZenQuotesIO
Select From Available Poem Sources:
[1] Gemini [ Working ][Set 'GEMINI_API_KEY' in .env ]
[2] OpenAI [ Not Fully Tested ][Will Require 'OPENAI_API_KEY' in .env]
[3] Perplexity [ Not Fully Tested ][Will Require 'PERPLEXITY_API_KEY' in .env]
Enter your choice: 1
-- you selected Gemini
Select From Available Image Sources:
[1] LimewireImage [ Working ][Set 'Limewire_API_KEY' in .env ]
[2] StabilityAIImage [ Not Ready ][Will Require 'StabilityAI_API_KEY' in .env ]
[3] ImagenImage [ Working ][Set 'GOOGLE_API_KEY' in .env ][Requires VertexAI Enabled with Imagen Allowlist]
Enter your choice: 2
-- you selected StabilityAIImage
Proceed to Fetch from Sources?
[1] Yes
[2] No. Exit Program
[3]. No. Start Selection Again.
Enter your choice: 1
Fetching Daily Quote
-----
Quote: Life is 10% what happens to you and 90% how you react to it.
Author: Charles Swindoll
Generating Poem
-----
In life's uncertain tide, where storms may brew,
Our choices shape the course, it's up to you.
Ten percent is fate, beyond our control,
But ninety percent lies within our soul.

How we respond to life's twists and turns,
Defines the path our destiny earns.
```



```
How we respond to life's twists and turns,
Defines the path our destiny earns.
With every trial, a chance to grow,
To rise above and let our spirits glow.

When darkness looms and shadows creep,
We have the power to light up the deep.
Embrace resilience, let it be your guide,
And turn life's challenges into a stride. ('quote': 'Life is 10% what happens to you and 90% how you react to it. ', 'author': 'Charles Swindoll', 'source': 'ZenQuotesIO')
Combining Blog Post
-----
Markdown file created at: /home/ /GeminiBlog/pelican-CMS/_site/content/20240728003435.md
Running Pelican to generate HTML output.
-----
Do you want to generate the HTML output using Pelican?
[1] Yes
[2] No
Enter your choice: 1
Pelican Deployer Directory: /home /GeminiBlog/Modules/.../pelican-CMS/_site
Running Pelican MakeFile
"pelican" "/home/ /GeminiBlog/pelican-CMS/_site/content" -o "/home/ /GeminiBlog/pelican-CMS/_site/output" -s "/home/ /GeminiBlog/pelican-CMS/_site/pelicanconf.py"
Done: Processed 4 articles, 0 drafts, 0 hidden articles, 0 pages, 0 hidden pages and 0 draft pages in 0.15 seconds.
Blog Post Generation Complete!
Static HTML Files for hosting will be present in /home/ /GeminiBlog/pelican-CMS/_site/output
-----
Media Files copied to: /home/ /GeminiBlog/pelican-CMS/_site/output/media
Do you want to deploy the Pelican output to Firebase?
[1] Yes
[2] No
Enter your choice: 1
Deploying Pelican output to Firebase.

=== Deploying to 'projectsaadhna-bloggergenerator-2'...

i deploying hosting
i hosting[projectsaadhna-bloggergenerator-2]: beginning deploy...
i hosting[projectsaadhna-bloggergenerator-2]: found 108 files in output
✓ hosting[projectsaadhna-bloggergenerator-2]: file upload complete
i hosting[projectsaadhna-bloggergenerator-2]: finalizing version...
✓ hosting[projectsaadhna-bloggergenerator-2]: version finalized
i hosting[projectsaadhna-bloggergenerator-2]: releasing new version...
✓ hosting[projectsaadhna-bloggergenerator-2]: release complete

✓ Deploy complete!

Project Console: https://console.firebase.google.com/project/projectsaadhna-bloggergenerator-2/overview
Hosting URL: https://projectsaadhna-bloggergenerator-2.web.app
Blog post should be generated and deployed successfully!
-----
Application Run Attempt Ended at 2024-07-28 00:46:14.062937
Total Time Taken: 0:12:46.134609
```

I have tried to anticipate the majority of exceptions or errors. Most of them will be logged. Some just still keep getting through. If you face any errors, check 'Logs/gemini-main.log'.

Review Logs

GeminiBlog > Logs > = gemini_main.log		
665	2024-07-28 00:48:49,110	INFO - main.py - 316 - Setting User Home Directory: /home/ /GeminiBlog/pelican-CMS/_site/content
666	2024-07-28 00:48:49,110	INFO - main.py - 317 - Setting Pelican Content Directory: /home/ /GeminiBlog/pelican-CMS/_site/content
667	2024-07-28 00:48:49,110	INFO - main.py - 318 - Setting Pelican Attached Media Directory: /home/ /GeminiBlog/pelican-CMS/_site/content/media
668	2024-07-28 00:48:49,110	INFO - main.py - 319 - Setting Pelican Deployer Directory: /home/ /GeminiBlog/pelican-CMS/_site
669	2024-07-28 00:48:49,117	INFO - main.py - 326 - PATH before adding local/bin /home/ /GeminiBlog/GPBenv/bin:/opt/gradle/bin:/opt/maven/bin:/usr/local/sbin:/usr/local/bin
670	2024-07-28 00:48:49,117	INFO - main.py - 331 - Local Bin was not present, appending /home/ /local/bin
671	2024-07-28 00:48:49,117	INFO - main.py - 333 - PATH after adding local/bin/home/ /GeminiBlog/GPBenv/bin:/opt/gradle/bin:/opt/maven/bin:/usr/local/sbin:/usr/local/bin
672	2024-07-28 00:48:49,117	INFO - main.py - 32 - Running the blog generation process in script mode.
673	2024-07-28 00:48:49,117	INFO - main.py - 39 - Fetching Daily Quote
674	2024-07-28 00:48:49,118	INFO - ModuleGetDailyQuotes.py - 33 - Entering Inside QuotablesIO Function
675	2024-07-28 00:48:49,874	INFO - ModuleGetDailyQuotes.py - 41 - Quote fetched from QuotablesIO: {'_id': '-CzNrWmGlg8V', 'content': 'Take things as they are. Punch when you have to punch
676	2024-07-28 00:48:49,875	INFO - ModuleGetDailyQuotes.py - 116 - Quote data returned to main: {'date': '2024-07-28 00:48:49', 'content': 'Take things as they are. Punch when you have to
677	2024-07-28 00:48:49,875	INFO - main.py - 47 - Generating Poem
678	2024-07-28 00:48:53,566	INFO - ModulePoemGenerator.py - 35 - Poem generated from Gemini: In life's ring, where challenges arise,
679	2024-07-28 00:48:53,566	INFO - main.py - 53 - Generating Image
680	2024-07-28 00:50:29,583	INFO - main.py - 292 - Application Run Attempt Started at 2024-07-28 00:50:29.583752
681	2024-07-28 00:50:29,587	INFO - main.py - 301 - Setting Up Directory
682	2024-07-28 00:50:29,590	INFO - main.py - 316 - Setting User Home Directory: /home/ /GeminiBlog/pelican-CMS/_site/content
683	2024-07-28 00:50:29,590	INFO - main.py - 317 - Setting Pelican Content Directory: /home/ /GeminiBlog/pelican-CMS/_site/content
684	2024-07-28 00:50:29,590	INFO - main.py - 318 - Setting Pelican Attached Media Directory: /home/ /GeminiBlog/pelican-CMS/_site/content/media
685	2024-07-28 00:50:29,590	INFO - main.py - 319 - Setting Pelican Deployer Directory: /home/ /GeminiBlog/pelican-CMS/_site
686	2024-07-28 00:50:29,597	INFO - main.py - 326 - PATH before adding local/bin /home/ /GeminiBlog/GPBenv/bin:/opt/gradle/bin:/opt/maven/bin:/usr/local/sbin:/usr/local/bin
687	2024-07-28 00:50:29,598	INFO - main.py - 331 - Local Bin was not present, appending /home/ /local/bin
688	2024-07-28 00:50:29,598	INFO - main.py - 333 - PATH after adding local/bin/home/ /GeminiBlog/GPBenv/bin:/opt/gradle/bin:/opt/maven/bin:/usr/local/sbin:/usr/local/bin
689	2024-07-28 00:50:29,598	INFO - main.py - 32 - Running the blog generation process in script mode.
690	2024-07-28 00:50:29,598	INFO - main.py - 39 - Fetching Daily Quote
691	2024-07-28 00:50:29,598	INFO - ModuleGetDailyQuotes.py - 33 - Entering Inside QuotablesIO Function
692	2024-07-28 00:50:30,282	INFO - ModuleGetDailyQuotes.py - 41 - Quote fetched from QuotablesIO: {'_id': 'p3WmUYECz33S', 'content': 'The meaning I picked, the one that changed my life: 0
693	2024-07-28 00:50:30,283	INFO - ModuleGetDailyQuotes.py - 116 - Quote data returned to main: {'date': '2024-07-28 00:50:30', 'content': 'The meaning I picked, the one that changed my l
694	2024-07-28 00:50:30,283	INFO - main.py - 47 - Generating Poem
695	2024-07-28 00:50:34,061	INFO - ModulePoemGenerator.py - 35 - Poem generated from Gemini: Fear's grip unyielding, a suffocating embrace,
696	2024-07-28 00:50:34,062	INFO - main.py - 53 - Generating Image
697	2024-07-28 00:50:34,063	INFO - ModuleImageGenerator.py - 78 - Sending image generation request to Limewire API with payload: {'prompt': "Generate a captivating, visually striking imag
698	2024-07-28 00:50:39,851	INFO - ModuleImageGenerator.py - 82 - Image Generation - Success!
699	2024-07-28 00:50:39,851	INFO - ModuleImageGenerator.py - 83 - You have 4.0 credits remaining.
700	2024-07-28 00:50:42,194	INFO - ModuleImageGenerator.py - 97 - Image downloaded and saved to: /home/ /GeminiBlog/pelican-CMS/_site/content/media/20240728005040.png
701	2024-07-28 00:50:42,194	INFO - main.py - 57 - Image file name: 20240728005040.png
702	2024-07-28 00:50:42,195	INFO - main.py - 60 - All 3 requirements met : Quote, Image, Poem. Generating MD File
703	2024-07-28 00:50:42,195	INFO - ModulePelicanController.py - 66 - Markdown content generated: ---
704	2024-07-28 00:50:42,195	INFO - main.py - 69 - Markdown file created at: /home/ /GeminiBlog/pelican-CMS/_site/content/20240728005042.md
705	2024-07-28 00:50:42,195	INFO - main.py - 71 - Running Pelican to generate HTML output.
706	2024-07-28 00:50:42,198	INFO - ModulePelicanController.py - 91 - Running Firebase command: make html in directory: /home/ /GeminiBlog/Modules/.../pelican-CMS/_site
707	2024-07-28 00:50:42,198	INFO - ModulePelicanController.py - 93 - Pelican command executed successfully.
708	2024-07-28 00:50:42,796	INFO - main.py - 76 - Static HTML Files for hosting will be present in /home/ /GeminiBlog/pelican-CMS/_site/output
709	2024-07-28 00:50:42,797	INFO - main.py - 77 - -----
710	2024-07-28 00:50:42,797	INFO - main.py - 79 - Copying generated images from content/media to output/media
711	2024-07-28 00:50:42,813	INFO - main.py - 82 - Media files copied to: /home/ /GeminiBlog/pelican-CMS/_site/output/media
712	2024-07-28 00:50:42,814	INFO - main.py - 86 - Deploy Pelican output to Firebase.
713	2024-07-28 00:50:42,814	INFO - ModuleFirebaseController.py - 18 - Running Firebase command: firebase deploy --project projectsaadhna-bloggergenerator-2 in directory: /home/
714	2024-07-28 00:50:52,679	INFO - ModuleFirebaseController.py - 22 - Firebase command executed successfully.
715	2024-07-28 00:50:52,679	INFO - main.py - 88 - Blog post should be generated and deployed successfully!
716	2024-07-28 00:50:52,679	INFO - main.py - 345 - Application Run Attempt Ended at 2024-07-28 00:50:52.679465
717	2024-07-28 00:50:52,679	INFO - main.py - 346 - Total Time Taken: 0:00:23.095713

Understand the Capacity and Limits of Package

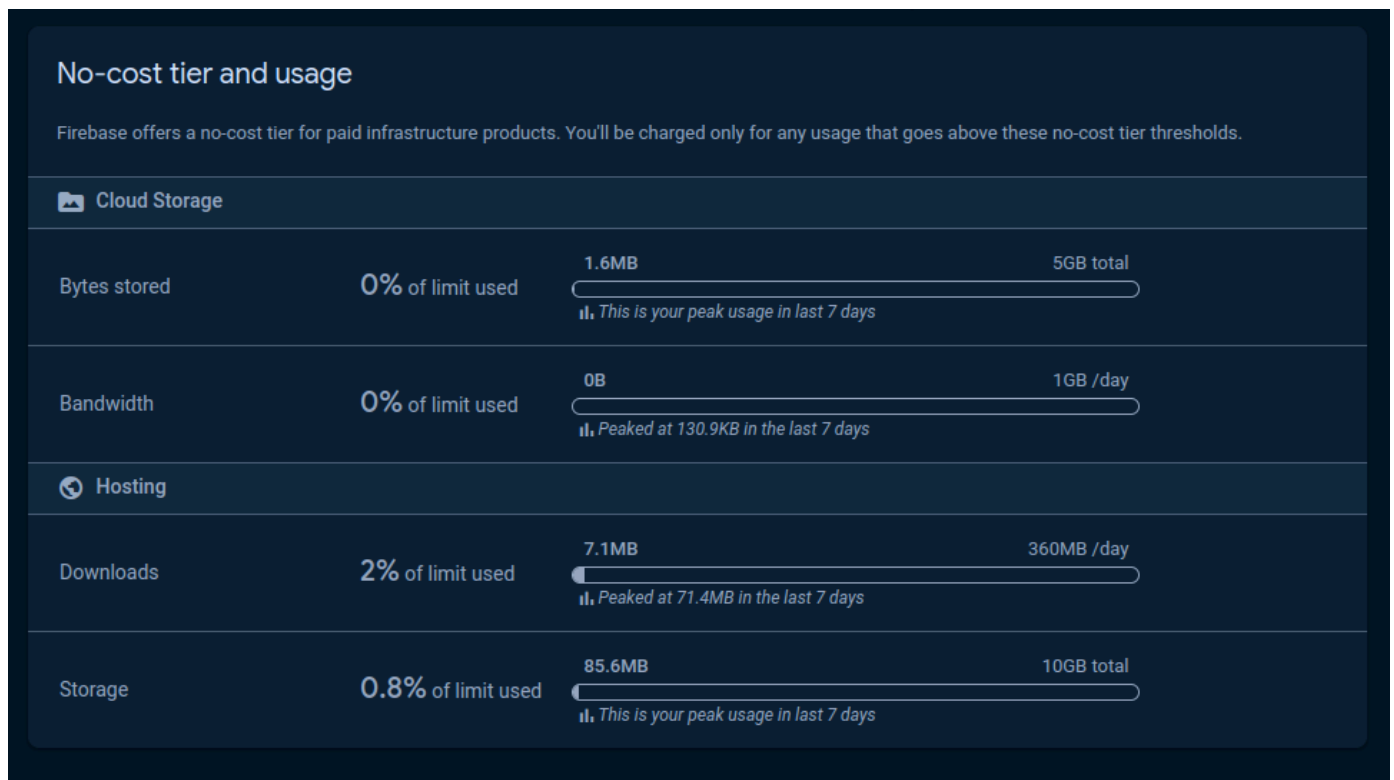
Increase in size over time.

It is understood that hosting this project on Firebase is only viable for demonstrating the functionality and for small scale applications. Since Firebase Hosting does not support incremental partial upload, each time you run firebase deploy, the complete fill is being pushed again and again.

This means it is fine for hosting a blog/application that you update on a fairly low number of postings.

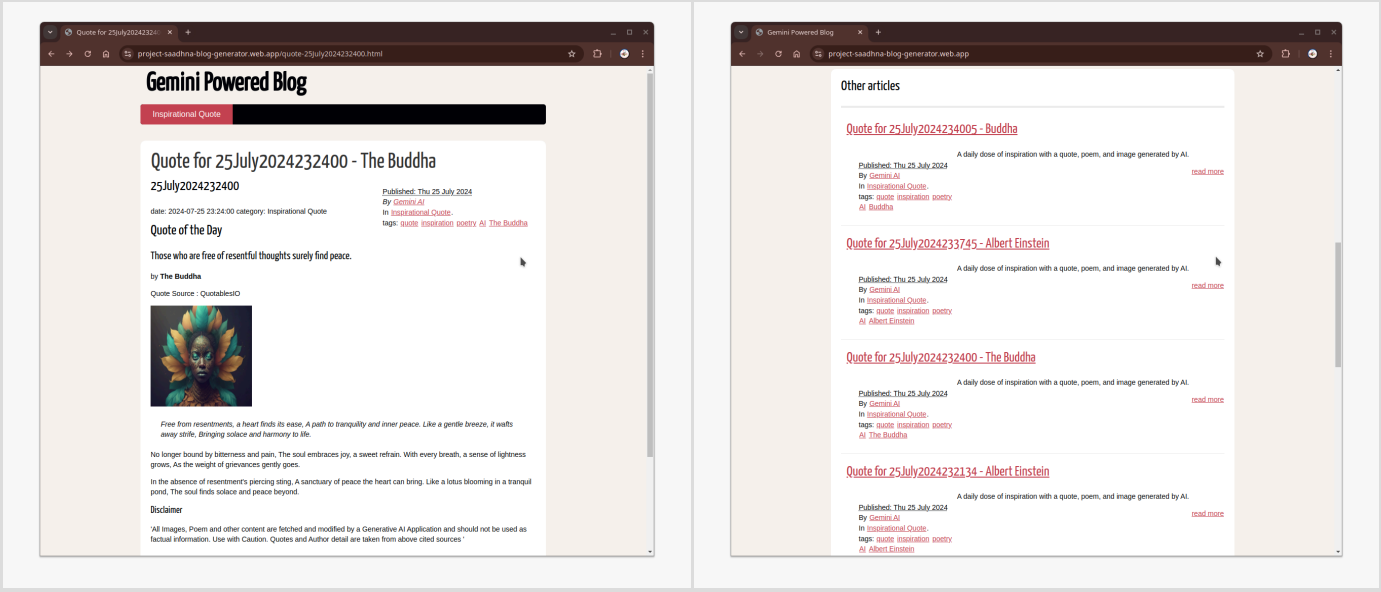
Potential Solutions :

1. Google recommends setting budget alerts for your project in GCP.
2. Visit [Billing and Usage Dashboard](#) to see how much storage and bandwidth you are using.
3. Move the project to a solution that does not require full reupload and can handle serving content with active incremental regeneration on change.

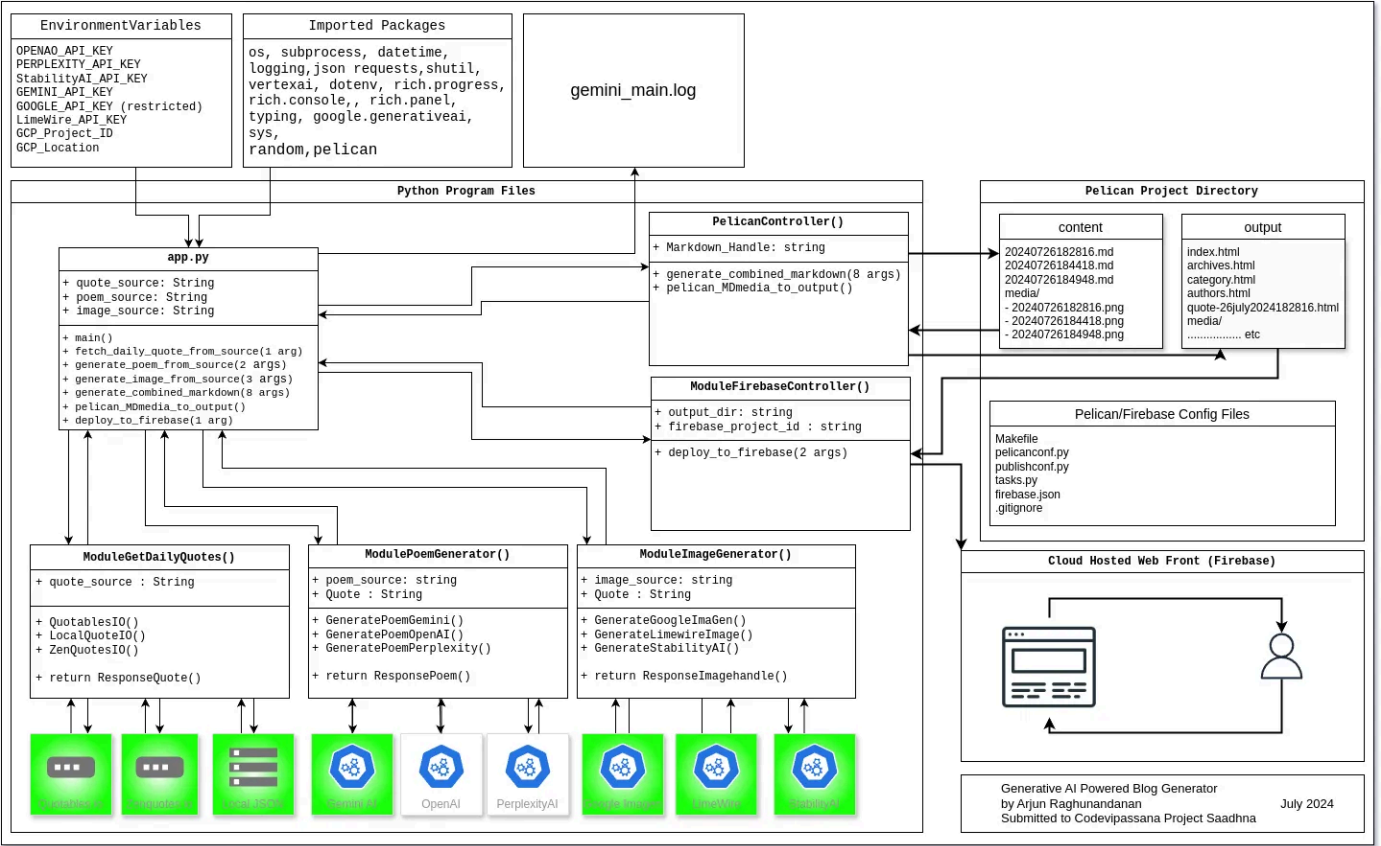


Final Product

This is how the final website looks.



Final Project Diagram



Task	Service	Status	Requirements
Quote Generation	ZenQuotes.io	Code Fully Functional	No API Needed

Task	Service	Status	Requirements
Quote Generation	Quotables.io	Code Fully Functional	No API Needed
Quote Generation	Local Json File	Code Fully Functional	Locally Available
Poem Generation	Google AI Studio : Gemini	Code Fully Functional	AI Studio API Needed : Free
Poem Generation	OpenAI	Pending : Not Fully Tested : Disabled in Code	API Needed : Paid
Poem Generation	PerplexityAI	Pending : Not Fully Tested : Disabled in Code	API Needed : Paid
Image Generation	Google Imagen2	Code Fully Functional	API Needed : Restricted GA
Image Generation	LimeWire	Code Fully Functional	API Needed : Limited Free
Image Generation	StabilityAI Core	Code Fully Functional	API Needed : Limited Free

Potential Project Improvements

Based on both additional experimentation done before review after reaching MVP and receiving feedback.

- Port out and modify projects to work as an App Engine Code or Containerize with Cloud Run . **[DONE]**
- Automate with Cron in Cloud shell/Linux and Cloud Scheduler for Cloud Native Solutions.
- Improve Prompting Coherency and provide Added Services on Blog other than just an AI Blog Generation. **[Out of Scope for Project Idea]**

Updates : Additional Changes Made

Section for updated changes outside of the main project idea if any.

Date	Changes
28/07/2024	Started Working on Cloud Native Solution after Cloud Shell Quota Reset
29/07/2024	Started Private Github Repo to Host Project Files
29/07/2024	Dockerized the solution to run on Google Cloud Run
29/07/2024	Solution 1 : Successfully Serving Static Blog output on Firebase : {redacted}.web.app
29/07/2024	Solution 2 : Successfully Serving Static Files on Unicorn at Cloud Run : {redacted}.a.run.app
29/07/2024	Introduced Basic Auth API POST Request function serving at {redacted}.a.run.app/generate
30/07/2024	Stripped down the project Files for Sharing on Github

Information on Updates

Version 3.0 is not intended to be part of the project submission and is just nice to have got done.

This is a containerized solution that can be hosted on Google Cloud Run.

You can trigger blog generation by sending an authenticated POST method request to the hosted service at the /generate entry point (**{redacted}.a.run.app/generate**). This allows you to schedule and automate periodical triggers to generate new posts on defined frequency. After processing the request, it will send a successfully done response to the POST method and the new post generated will be available at **{redacted}.a.run.app/generate**

You can send the request using any of the following methods.

Using Curl

```
curl -X POST -u admin:password -H "Content-Type: application/json" -d '{
  "quote_source": "QuotablesIO",
  "poem_source": "Gemini",
  "image_source": "LimewireImage"
}' http://{redacted}.a.run.app/generate
```

Using Python Requests

```
import requests
url = 'http://{redacted}.a.run.app/generate'
headers = {'Content-Type': 'application/json'}
data = {
```

```

"quote_source": "QuotablesIO",
"poem_source": "Gemini",
"image_source": "LimewireImage"
}
response = requests.post(url, headers=headers, json=data, auth=('admin', 'password'))
print(response.text)

```

Screenshot of POST request update in Action

The screenshot shows a code editor with a file named `testcurl.py`. The script imports `requests` and `json`, sets a URL to `https://...a.run.app/generate`, and defines a `basic_auth` function. It then makes a POST request with a JSON body containing source information and prints the response. The terminal output shows the command `python testcurl.py` being executed, resulting in a 200 status code and a detailed JSON response containing a poem and a quote.

```

testcurl.py x
testcurl.py > data
1 import requests
2 import json
3 url = 'https://...a.run.app/generate'
4 #url = 'http://localhost:8080/generate'
5 headers = {'Content-Type': 'application/json'}
6 data = {
7     "quote_source": "ZenQuotesIO",
8     "poem_source": "Gemini",
9     "image_source": "LimewireImage"
10 }
11
12 def basic_auth(username, password):
13     """Simple authentication function."""
14     return (username, password)
15 #auth = basic_auth("admin", "password123")
16 auth = basic_auth("...", "A")
17
18 response = requests.post(url, headers=headers, json=data, auth=auth)
19
20 print(response.json())
21
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
cloudshell:~ (project-saadhna-bg-gae)$ python testcurl.py
{'Response Code': 200, 'image_file_name': '20240730164626.png', 'message': 'Content generated successfully!', 'poem': "***Song of the Unsung**\n\nMelodies dance in silent sway,\nA symphony of dreams yet to play.\n\nThe strongest notes, the sweetest strain,\nAwait their time to burst and gain.\n\nIn hidden depths, where echos dwell,\nA chorus waits, its story to tell.\n\nThe harmonies that time conceals,\nWill someday pierce the air like peals.\n\nLike morning dew on petals soft,\nMusic's essence will waft aloft.\n\nThe strongest and sweetest songs entwined,\nA symphony that's utterly divine.", 'quote_data': {'author': 'Walt Whitman', 'quote': 'The strongest and sweetest songs yet remain to be sung.', 'source': 'ZenQuotesIO'}}
@cloudshell:~ (project-saadhna-bg-gae)$

```

Based on the response, here is the context. *This output is manually modified for easy reading*

Server Response: Content generated successfully!

Poem: ****Song of the Unsung***
 Melodies dance in silent sway,
 A symphony of dreams yet to play.
 The strongest notes, the sweetest strain,
 Await their time to burst and gain.
 In hidden depths, where echos dwell,
 A chorus waits, its story to tell.
 The harmonies that time conceals,
 Will someday pierce the air like peals.
 Like morning dew on petals soft,
 Music's essence will waft aloft.
 The strongest and sweetest songs entwined,
 A symphony that's utterly divine."

Image:20240730164626.png

Quote:

"author": "Walt Whitman"

"quote": "The strongest and sweetest songs yet remain to be sung."

"source": "ZenQuotesIO"

You can review the performance and uptime of Cloud Run applications in Cloud Trace.

[Visit Github Project Repo](#)

[Visit Arjun's Website](#)

Disclaimers from Developer.

- There are non-google alternatives included to make the project more generalized. (eg OpenAI,Perplexity for Poem Generation, LimeWire, StabilityAI for Image generation) This was not an initial plan but was decided during development, This was done because my initial idea of Imagen was reliant on Google Imagen Team accepting my request to grant access to **Imagen on VertexAI** . Since this could be revoked at any time, It's good to have other options. If one breaks, you should still be able to use an alternative option.
- This program is only developed to show learner project level application of generative AI to explore its capabilities and it does not necessarily encourage pushing bulk AI Generated content to spam the web.
- Live Version of project will not be maintained after formal project period to avoid incurring run-time costs and service call quota limits.
- Due to the Non-Deterministic Nature of Generative AI based content creation, There is always a high risk of posts generated with offensive message, visual, quotes. The developer is not reviewing each iteration of the generated post upon periodic deployment.
- Regardless of whether you are using Limewire's Image Generator, Google's Imagen, OpenAI's Dall-E, Always review your generated content before publishing them in a formal use-case. Never use this code 'as is' for production. I have skipped the Per Post Review step for this project implementation. **This is an example of 'Do as I say, not as I do'.**
- I made this in under 1 week with a lot of help from Documentations (see references at the end) and with Google Cloud Code Assist. There may still be errors or future dependency breakdown and this application is not fully error proof when all service options are unavailable.
- Some parts of the program are partially developed and may not be developed further. They are marked as such. Review program modules to verify before use to avoid breaking. Some services may stop providing their API that this code depends on.
- Can't guarantee the upkeep and availability of the project live view in a future date.

Project by

Arjun Raghunandanan

www.arjunraghunandanan.com

linkedin.com/in/arjunraghunandanan

July 2024

API & Package Documentations

Quote Generation

ZenQuotes

<https://docs.zenquotes.io/zenquotes-documentation>

Quotable

<https://docs.quotable.io/docs/api/ZG9jOjQ2NDA2-introduction>

Poem Generation

Gemini AI Studio models/gemini-1.0-pro

<https://ai.google.dev/gemini-api/docs/quickstart?lang=python>

<https://github.com/google-gemini/generative-ai-python/blob/main/docs/api/google/generativeai>

OpenAI

<https://platform.openai.com/docs/guides/text-generation/quickstart>

Perplexity AI

https://docs.perplexity.ai/reference/post_chat_completions

Image Generation Reference Content URLs

LimeWire API (Version 1.1.2) for Image Generation

<https://docs.limewire.com/#operation/generateImage>

Google Imagen 'imagegeneration@006' Model

<https://cloud.google.com/vertex-ai/generative-ai/docs/model-reference/imagen-api>

https://github.com/GoogleCloudPlatform/python-docs-samples/tree/main/generative_ai/imagen

StabilityAI REST API (v2beta)

<https://platform.stability.ai/docs/api-reference#tag/Generate/paths>

Project Saadhna

Code Vipassana

<https://rsvp.withgoogle.com/events/cv>

Project Saadhna Cycle 2

<https://rsvp.withgoogle.com/events/ps-2>

Pelican Documentation

Pelican Static Site Generator Version 4.9.1

<https://pypi.org/project/pelican/4.9.1/>

<https://github.com/getpelican/pelican>

<https://docs.getpelican.com/en/latest/quickstart.html>

<https://github.com/getpelican/pelican-themes>

Hosting

Firebase

<https://firebase.google.com/docs/cli>

[https://firebase.google.com/s/results?q=Google Cloud&product=googlecloud](https://firebase.google.com/s/results?q=Google+Cloud&product=googlecloud)

<https://adityacodes.medium.com/build-a-website-with-google-firebase-from-scratch-dd95a99c8d38>

Others

Cloud Shell Environment Variables

https://cloud.google.com/shell/docs/configuring-cloud-shell#environment_customization_script

<https://cloud.google.com/shell/docs/configuring-cloud-shell>