Image Generation and Similarity Search Service

Ashwin Baluja

March 21, 2025

1 Project Overview

This project is an image generation and similarity search service. Users can generate images from text prompts, upload images, and find semantically similar images. Specifically, they can find the most similar images that were generated with a specific prompt (e.g., finding a generated fork that is a similar style to a real image of a fork, without comparing to generated images of other objects). Similarity is determined using CLIP embeddings and cosine similarity.

2 System Architecture

The service uses a primarily serverless architecture:

- **API Gateway:** Serverless RESTful API interface.
- **Lambda Functions:** Serverless business logic for each endpoint.
- **AWS Bedrock:** Serverless, managed text-to-image generation.
- **Amazon SageMaker:** CLIP model for embeddings. *Not serverless due to long startup times*
- **Amazon S3:** Image storage.
- **Amazon DynamoDB:** Image metadata and embedding database.

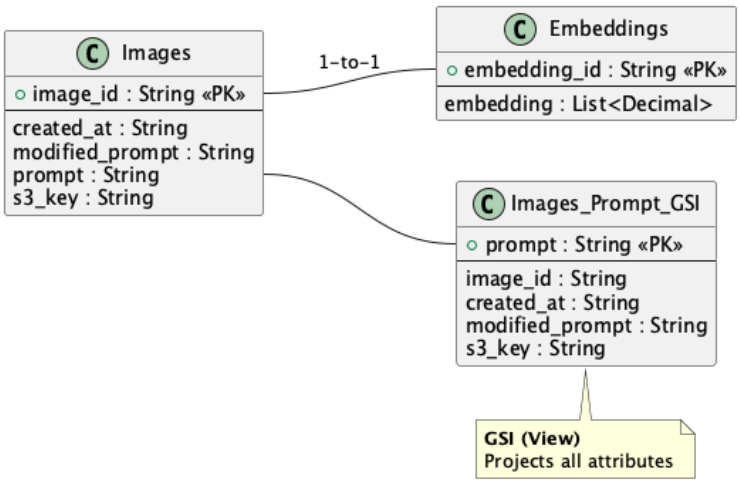

Figure 1: DynamoDB Schema and Relationships

3 API Design

Detailed endpoint descriptions and sequence diagrams are provided below.

Endpoint	Method	Parameters	Description
`/images`	GET	`prompt` (query, optional)	Generates a new image.
`/images/{image_id}`	GET	`image_id` (path)	Retrieves image and metadata.
`/images`	POST	`image_data` (body, base64)	Uploads an image.
`/embeddings/{embedding_id}`	GET	`embedding_id` (path)	Retrieves/generates embedding.
`/similarity`	GET	`image_id, prompt` (query)	Finds similar images.

Table 1: API Endpoint Summary

3.1 GET /images

Generates a new image from a text prompt using AWS Bedrock. Saves the image to S3 and writes metadata to DynamoDB. Appends random camera angle and style modifiers to the prompt.

- **Success (200 OK):**

```
{
  "base_prompt": "...",
  "modified_prompt": "...",
  "image_id": "..."
}
```

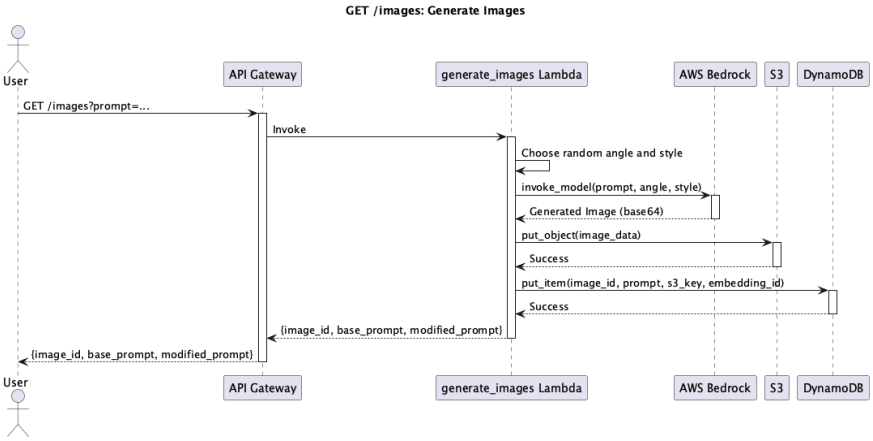

Figure 2: GET /images Sequence Diagram

3.2 GET /images/{image_id}

Retrieves an image by ID. Returns metadata from DynamoDB and a presigned URL to S3.

- **Success (200 OK):**

```
{
  "image_id": "...",
  "prompt": "...",
  "s3_key": "...",
  "url": "{presigned url to S3}"
}
```

3.3 POST /images

Uploads an image. Puts it to S3 and adds its metadata to DynamoDB.

- **Success (200 OK):**

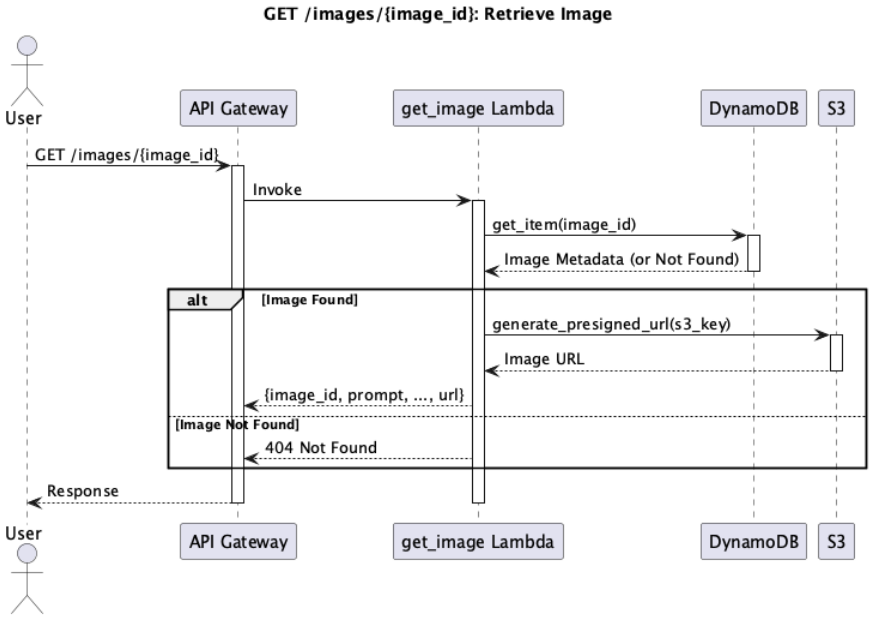

Figure 3: GET /images/{image_id} Sequence Diagram

```
{
  "image_id": "...",
}
```

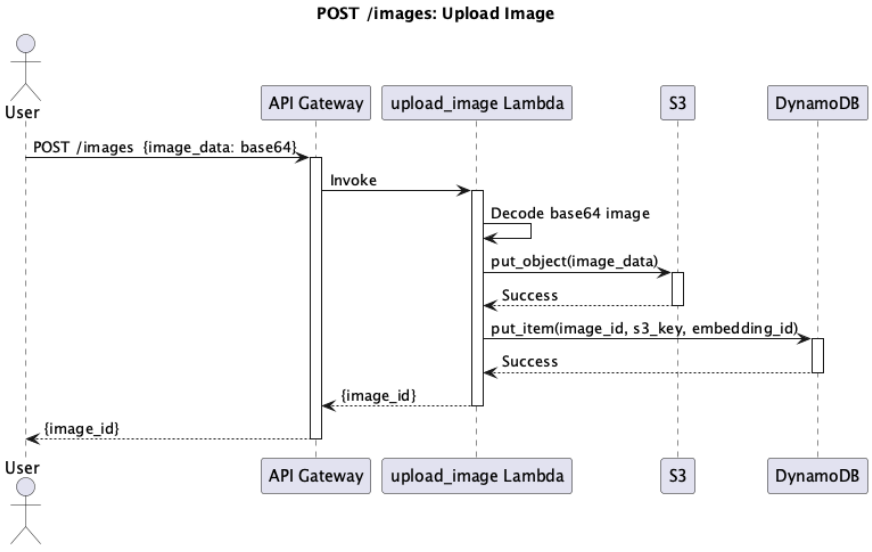

Figure 4: POST /images Sequence Diagram

3.4 GET /embeddings/{embedding_id}

Retrieves an image embedding from DynamoDB. If it does not exist, fetches image from S3, generates embedding using CLIP model hosted on SageMaker, and writes the embedding to DynamoDB.

- **Success (200 OK):**

```
{
  "embedding_id": "...",
  "embedding": [
    ...
  ]
}
```

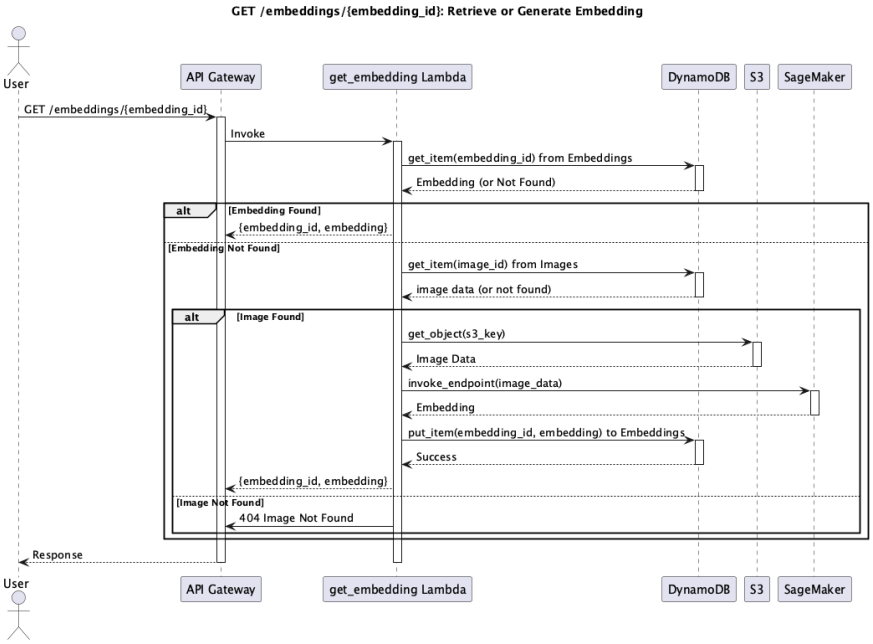

Figure 5: GET /embedding/{embedding_id} Sequence Diagram

3.5 GET /similarity

Finds similar images to a given image. Filters images compared to only images generated with a certain prompt, so as to compare between style and camera angle within one class of objects. Only searches through images that have embeddings already generated. Returns top 10 most similar matches, as determined by maximum cosine similarity.

- **Success (200 OK):**

```
    {
      "results": [
            {
                "image_id": "...",
                "similarity": ...
            },
            ...
        ]
    }
```

4 Database Schema

The service uses two DynamoDB tables: 'Images' and 'Embeddings'. A Global Secondary Index 'Images_Prompt_GSI', allows filtering by prompt. See Fig. 1, Table 2, and Table 3.

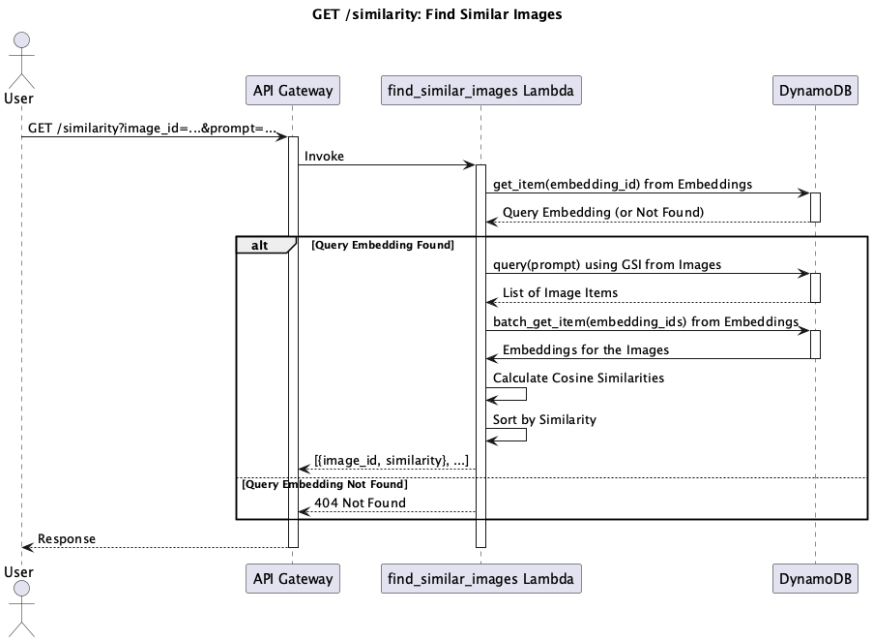

Figure 6: GET /similarity Sequence Diagram

Attribute	Type	Description
`image_id`	String	Unique image ID (Primary Key)
`created_at`	String	Timestamp
`modified_prompt`	String	Full prompt
`prompt`	String	Base prompt
`s3_key`	String	S3 path

Table 2: Images Table Schema

Attribute	Type	Description
`embedding_id`	String	Embedding ID (Primary Key)
`embedding`	List⟨Decimal⟩	Embedding vector

Table 3: Embeddings Table Schema