

Intelligent Photo Retrieval System Using Text and Image Queries

Group 51

Group Members Alibek Aitzhan
Nurlan Nurlybay
Aidyn Kazhakhmet
Elnaz Kospanova

Adviser Fatih Demirci

Course CSCI 409 Senior Project II

Semester Spring 2026

Submission Date April 20, 2026

Contents

1	Executive Summary	2
2	Introduction	2
3	Identification of Constraints	3
4	Background and Related Work	5
5	Project Approach	7
5.1	High-Level Architecture	7
5.2	Core Workflows	8
5.4	Third-Party Components and Integration	9
5.5	Team Functioning and Roles	9
6	Project Execution	10
6.1	Fall 2025 Baseline	10
6.2	Spring 2026 Continuation and Major Changes	10
6.3	What Went Right	11
6.4	Problems Encountered and Mitigations	12
6.5	Teamwork, Leadership, and Collaboration	13
7	Evaluation	13
7.1	Stage 1: Evaluating Fine-Tuning of CLIP	13
7.2	Stage 2: A Comparison Between CLIP and SigLIP	14
7.3	Stage 3: RAG-Type Reranking	15
7.4	Stage 4: Final Weighted Retrieval Design	16
7.5	Interpretation	17
8	Conclusion and Future Potential Research	17
	References	19

1. Executive Summary

This project presents a smart photo retrieval system application which provides users with an opportunity to search through their pictures using both natural language prompts (dog, cat, mountain, nighttime party etc.) and example images. The system uses a multimodal vector representation to search through the images using natural language, therefore removing the issue of relying on users manual tagging process and filename based searches.

We developed a working prototype in Fall 2025 using CLIP embeddings, a Go server-side application programming interface (API), Python applications utilizing machine-learning/vector-services, Milvus vector searching, and a Flutter mobile client application. In Spring 2026, we further expanded upon this base to create a more robust production-oriented architecture including strong multi-tenancy, cloud-based processing/storage, asynchronous job scheduling/orchestration, support for additional model-stack architectures, and more comprehensive data-lifecycle control.

Our completed architecture consists of a Go API-gateway, Python-based ML-inference worker applications (that utilize both fast and deep semantic pathways), a Milvus vector-service adaptor, PostgreSQL-based metadata storage, Redis-based queues/publish-subscribe messaging service, and AWS S3 object storage. Our evaluations were centered around solution-level effectiveness rather than simply testing our prototype during development time: We assessed retrieval-quality metrics across multiple different model variants, manually evaluated all aspects of end-to-end functionality including tenant-isolation and deletion cascades, and performed distributed-environment deployment-readiness assessments. The final hybrid retrieval design preserved fast initial search while substantially improving top-result accuracy on our real-gallery evaluation set.

Our ultimate product represents a solution that is both modular and scalable, designed for deployment in production environments with evidence supporting that our Spring 2026 design provides improved operational resiliency as well as increased user-control over their respective data sets, without degrading semantic-search performance.

2. Introduction

It is often difficult for users to efficiently manage and find specific images in large collections of digital photos, especially when relying on metadata or text queries. One of the major issues with how users tag their photos is inconsistent tagging behavior, which makes it difficult to perform consistent metadata based

searches. Additionally, performing manual organization of large collections of digital photographs is extremely time consuming and scales poorly. As such, this project was driven by the need to develop an intuitive image retrieval method whereby users can enter natural language query strings (e.g., “white flowers on a tree”, “people on a hike”) and/or provide example images to achieve similar results quickly.

While developing a single image-processing model represents one of the largest challenges of the project, there are several other components that require consideration in order to develop a complex image retrieval system. These include ingestion, indexing, retrieval, user isolation, lifecycle management and frontend integration. Therefore, the primary focus of this project is developing and evaluating an end-to-end system that uses machine learning together with reliable software engineering and cloud deployment practices. More specifically, the project investigates how a personal photo retrieval system can remain both accurate and responsive by separating fast initial retrieval from deeper semantic enrichment.

As compared to Fall 2025, in Spring 2026 we extended the project beyond the capabilities of a prototype to represent a more robust distributed platform that includes:

- multi-tenancy request semantics (i.e. the requirement for user_id across all API's)
- two-speed asynchronous processing (i.e. fast path and deep semantic path-way)
- stronger operational workflows (i.e., queueing, state-transitions, CI/CD workflow, and cloud registration workflow)
- verified deletion cascades spanning PostgreSQL, S3, and Milvus

3. Identification of Constraints

The project has some constraints which directly influence design decisions and implementation trade-offs.

Technical constraints: Inference costs of GPUs are quite expensive for multi-modal models. To control the costs, the architecture separates API/backend hosting from GPU-hosted ML workers and introduces a fast path (SigLIP) plus a slower deep reasoning path (Qwen-3-VL), so high-cost processing is used only when needed. When we were in a process of evaluation and data preparation we discovered another significant technological limitation. Public image reposi-

tories like Unsplash provide a convenient way to acquire large volumes of data easily. Open image repositories, like Unsplash, provided an easy way to acquire large amounts of images. However, their associated metadata and tags were often noisy, incomplete, or only loosely related to the actual visual content of the images. Thus, there existed limitations in the feasibility of performing evaluations on this basis since it was often assumed that the metadata represented the “ground truth” for evaluating how relevant retrieved results are. Consequently, later parts of our evaluation relied on acquiring manually tagged real gallery images personally, so as to assess the relevance of queries more closely to reality.

Environmental/Sustainability constraints: High-resource utilization models consume significant amounts of power. Therefore we have implemented various techniques to minimize unnecessary computation and storage utilization in our system including asynchronous processing, model-quantization options, and duplicate-detection mechanisms to help mitigate wasteful resource consumption.

Social constraints: Trust of the users can be significantly impacted by both timely and transparent retrieval performance. Systematically delayed indexing or poor ranking can negatively impact usability. We’ve designed our system to provide status states (pending/fast-encoded/etc.) and show real-time update information to improve visibility.

Political/Legal constraints: Photos are frequently classified as Personally Identifiable Information (PII) because they can reveal a person’s identity, location, or sensitive traits. Jurisdictional expectations for handling this data are modeled after GDPR-like regulations, which emphasize data minimization, strict deletion policies, and limited user access scoping. To comply with these standards, the system must ensure that data is strictly partitioned between users. Also, include a dedicated per-account delete endpoint to facilitate the permanent removal of user data upon request.

Ethical constraints: Vision Language models inherently capture potential biases in their training datasets. Potentially biased and skewed associations may also be returned by the system because of that. We’ve documented model limitations and maintained user-delete control to mitigate this issue and avoid exposing unnecessary sensitive metadata.

Health & Safety constraints: There are no direct physical risks associated with the system. However psychological/privacy related issues can occur if personal data is leaked. We’ve therefore placed strong emphasis on providing reliable tenant boundaries as well as safe deletion processes.

Deployability constraints: Since this is a student-led effort operating under

limited financial means within existing cloud computing resources we had to make sure that each component could be easily maintained by ourselves. We achieved this by designing modular services with clearly defined interfaces, using containers for deployment and implementing CI/CD automated workflows for reproducible builds. The inference delay also became another very significant constraint for the entire system. Reranking with a deeper VLM significantly improved the semantic accuracy of search results. However, this process was too computationally expensive to execute within the query phase at production scale. Consequently, we were forced into a design tradeoff between search accuracy and system response time. This led to our decision to replace real-time reranking with an asynchronous pre-indexing strategy for every user query.

These constraints necessitated a shift from a monolithic architecture toward modular cloud services. This modular approach ensures that each team member can independently manage and scale their respective microservices without the complexities of a single, unified codebase.

4. Background and Related Work

Looking for your own images using file name, timestamp, folder name or manual tagging can be a difficult task when the user did not memorize those patterns, but remembers only the image description. There have been many studies that show that searching for images via metadata can be inconsistent or not complete. Such inconsistencies create an issue where a gap between how people remember and can describe their images and how the image search engine can find them is of size. Personal photographs are particularly problematic with regard to the lack of detail in manually assigning tags. In addition to the fact that users usually do not apply descriptive tags to each photograph, many times there are multiple photos of the same event. Additionally, there may be other photos that contain the subjects of interest, but not necessarily within the context of the original event. These factors contribute significantly to why users will generally prefer to browse through their personal photo albums to locate specific photos rather than attempting to search for them. Before deep learning, content-based image retrieval (CBIR) systems utilized computer-extracted visual characteristics to perform better than traditional metadata-based searches. Although, as noted previously, some of these early attempts allowed for image-based retrieval of similar images, they did not effectively capture the overall semantics contained in images, such as what happened in the image, the scene depicted in the image, etc. (Rui et al., 1999; Datta et al., 2008). More recently, researchers have begun applying deep-learning techniques to develop new forms of image representation known as “learned image embeddings.”

Learned embeddings provide much stronger semantic representations than computer-extracted features (Dubey, 2021). This is especially relevant in the case of personal photo galleries, where users usually remember scenes, people, and events semantically rather than by filename, folder structure, or manually assigned tags.

Visual-Language Models represent one of the most significant advancements in this area. They learn a common embedding space for both text and images. CLIP was the first model to enable direct text-based image retrieval. CLIP enabled users to query images using natural language when no manual labels existed. Therefore, CLIP represents a solid base for developing personal photo retrieval systems. Subsequent models such as SigLIP built on this concept. SigLIP uses a sigmoid-based training objective, enabling more accurate and scalable retrieval relative to previous models (Zhai et al., 2023). Other subsequent models, such as Qwen2-VL extended these concepts into areas involving more complex visual reasoning and deeper semantic understanding. While improving performance in this manner increases computational expense (Wang et al., 2024). The tradeoff between retrieval accuracy and computational expense justifies the methodology employed by this study. One can select a simple/fast model capable of handling virtually all retrieval tasks at a lower cost. Alternatively, one could employ a more complex model to analyze images at a deeper level and with more advanced visual reasoning. This concept is represented by the two-stage architecture employed in this study. Stage one employs a fast semantic pathway to quickly facilitate retrieval. In contrast, stage two utilizes a slow pathway with more advanced reasoning abilities. In particular, this architecture was described in the Spring 2026 report as a means of increasing retrieval accuracy while maintaining control of GPU-related expenses. Practical System Design has generated three primary alternatives for addressing these two issues with respect to retrieval. These alternatives include either (1) a single stage retriever that uses embeddings as a basis for the retrieval model, (2) an end-to-end reranking pipeline for every query, or (3) a pre-indexed enrichment pipeline. We have chosen to follow alternative number 3 as our final architecture because it retains much of the semantic benefit from deepening the VLM's reasoning process without incurring the high query time latencies associated with reranking.

Furthermore, performing efficient retrieval over dense vectors relies heavily upon appropriate indexing strategies. Due to the inefficiencies associated with exact nearest neighbor searches on very large datasets, most contemporary systems utilize approximate nearest neighbor (ANN) methods. Among the two most commonly used methods for scalable high dimensional similarity search

are FAISS and HNSW (Johnson et al., 2017; Malkov & Yashunin, 2018). Consequently, vector database technology, specifically Milvus, has grown increasingly popular as it enables developers to easily store and retrieve image and text embeddings without having to implement indexing logic from scratch. Given this functionality, Milvus is an ideal candidate for use in multimodal retrieval systems similar to the ones presented in this study. Both the prototype and final system architectures indicate the usage of Milvus for storing and retrieving vectors.

To summarize, metadata-only image search is ineffective due to many modern personal photo collections containing too little metadata. CBIR methods created using pre-defined features are less effective than methods employing deep learned features for representing semantic meaning. Vision-Linguistic embeddings represent a good starting point for both text-to-image and image-to-image retrieval. However, ANNs employed in vector databases provide a viable method of scaling these types of applications. Therefore, the chosen architecture is valid based on combining recent advancements in multimodal retrieval research with well-established software engineering practices to scale these applications. More specifically, the literature supports the use of fast embedding based retrieval for scalable search, while our project extends this idea by expanding on it with asynchronously generated semantic descriptions and tag-aware scoring.

5. Project Approach

5.1 High-Level Architecture

The final system contains the following components:

- **Go Backend (API Gateway):** Handles uploads/search requests, metadata workflows, dedup logic, and orchestration.
- **ML Service (Python):** Performs multimodal inference through two pathways: a fast SigLIP-based pathway for immediate embedding and retrieval, and a slower deep-semantic pathway using Qwen-based vision-language processing to generate richer descriptions and tags during asynchronous background processing.
- **Vector Service (Python + Milvus adapter):** Manages vector insertion and nearest-neighbor search.
- **Hybrid Retrieval Layer:** Combines visual embeddings, generated textual descriptions, and exact tag-word overlap through a weighted scoring function

optimized using Optuna to maximize ranking quality.

- **PostgreSQL:** Stores media metadata and processing state.
- **Redis:** Supports queue-based background execution and status streaming via pub/sub.
- **AWS S3:** Stores original media and serves retrieval URLs.
- **Flutter Frontend:** Upload, search, and gallery interactions for end users.

5.2 Core Workflows

Ingestion workflow: upload → deduplication/metadata extraction → S3 storage → DB insert (pending) → queued ML processing → Milvus indexing → DB state update (active) → status notification.

The spring 2026 design was finished as described. At the end of uploading an image (or at least shortly thereafter) the image will go immediately down a fast semantic path. This allows the user to search on the images without having to wait for a full or longer analysis of each uploaded image. Simultaneously with the fast semantic path there is also a background process that is analyzing the same image(s). After this second pass has completed, the system then updates the searchable representation of the image and switches to a weighted retrieval method based on greater accuracy.

Retrieval workflow: query input (text or image) → query embedding → Milvus top-k search → metadata enrichment from PostgreSQL → ranked response to client.

In one iteration of testing the retrieval method was similar in behavior to that of a RAG-type pipeline. A candidate top k-set of images was returned by SigLip and were subsequently reranked by Qwen using both the text query and the candidate images themselves. While this did improve the quality of the top rank returns, this style of retrieval proved to be much too slow for production use if multiple images needed to be sent into the VLM at once via a single request.

As such, the final retrieval workflow placed deep semantic reasoning in pre-indexing rather than in request time. Rather than have Qwen rerank results for every query, the system utilized off-line semantic enrichment to create descriptions and tags during the upload processing of each image. For the purposes of ranking at query time, the system then combined three types of signals: Visual similarity, Textual description similarity, Exact Tag/Query Word Matches.

Data lifecycle workflow: granular delete, gallery clear, and full-account deletion across DB, storage, and vector layers.

5.4 Third-Party Components and Integration

The initial iteration of the project required the use of many additional tools that would enable scalable deployment and experimentation as well, such as AWS ECR and DockerHub for container registry workflows, Vast.ai for GPU-hosted ML inference, and Optuna for tuning hybrid retrieval weights.

The project integrates external technologies not implemented from scratch:

- Milvus for vector indexing and ANN search,
- Redis for queueing and pub/sub,
- PostgreSQL for metadata persistence,
- AWS S3 for media object storage,
- Flutter, Gin, and FastAPI for client/server layers.

All of these integrations were done through clearly defined API bounds and containerized services.

5.5 Team Functioning and Roles

The team employed an iterative collaboration model (sprint-style development, issue tracking, code reviews and advisor feedback). The team had responsibilities distributed across backend orchestration, ML/vector pipeline design, front-end integration, and infrastructure/deployment. This distribution allowed the team to develop parallel to one another and have shorter cycles for integrating their parts of the system while still maintaining overall architectural integrity.

Responsibilities were generally grouped around three primary areas: back-end/orchestration (API workflow, queues, Redis task routing, S3 object handling, tenant safe lifecycle operations), ml/retrieval (model experimentation/pipeline design for SigLip & Qwen; Hybrid Weighted Retrieval; Evaluation Scripting; Optuna Tuning) and infrastructure (containerization; CI/CD; Cloud Registry; GPU Service Deployment). The front-end work included integrating Flutter into our application so that users can upload images, perform searches, view galleries and visually monitor how the processing states of objects evolve. While all team members worked together across these domains when troubleshooting/integrating, each member primarily owned responsibility for one of these large work domains.

6. Project Execution

6.1 Fall 2025 Baseline

The first semester established a working prototype:

- *CLIP*-based semantic retrieval pipeline,
- *Go* backend with core upload/search media endpoints,
- *Python* ML and vector microservices,
- *Milvus* vector indexing and *SeaweedFS*/*PostgreSQL* storage stack,
- basic *Flutter* upload and search interfaces,
- initial integration and demo readiness.

This baseline validated technical feasibility but also exposed limitations in scalability, user isolation, lifecycle management, and production deployment readiness. For example, uploaded images were not yet isolated per user, deleted media could remain partially referenced across services, and everything ran only locally.

6.2 Spring 2026 Continuation and Major Changes

In Spring 2026, we executed a significant architectural improvement:

- shifted storage/deployment patterns toward AWS S3 and cloud-oriented service orchestration,
- introduced asynchronous queued processing for ingestion workloads,
- enforced strict multi-tenant semantics across frontend/backend contracts,
- added real-time processing status reporting via stream events,
- expanded deletion flows into a three-tier cascade (single-item deletion, gallery clear, full account deletion).

In addition to the retrieval model experimentation conducted in Fall 2025, significant additional retrieval-model experimentation took place during Spring 2026. We compared several different retrieval backends, which included both *CLIP* and *SigLIP*-family variants. It was revealed that the retrieval quality of *SigLIP*-based models exceeded the results obtained from the *CLIP* configurations within the experimental setup. While not all of the retrieval models evaluated represented the most recent development at the state-of-the-art; however, these experiments allowed us to identify an appropriate balance between retrieval accuracy and latency, i.e., a “sweet spot.”

A mid level version of the design incorporated a type of RAG-style retrieval strategy. The first stage utilized SigLIP to select a small pool of potential images as candidates. In the second stage, Qwen reranked those candidate images by jointly evaluating both the query-image and each candidate-image. The use of this retrieval strategy significantly improved the quality of the ranked lists, particularly near the top of the ranked lists. Unfortunately, this added substantial overhead to the system's overall processing time. Therefore, the system was too slow for a production oriented user experience.

As such, the final design of the system moved the Qwen-type of semantic reasoning from being executed at request time to being executed prior to index building. Rather than reranking numerous images for every query, the system generates semantic descriptions and tags for each image to be indexed into a database. This indexing process is handled as an asynchronous operation that runs efficiently in the background. To perform retrievals, the system combined visual embedding similarities, description similarity scores, and overlap scores via a weighted retrieval strategy.

The weights assigned to each component of the retrieval strategies were optimized using Optuna with respect to the ranking quality metric (NDCG) to produce a much faster query time system while retaining the advantages of deeper semantic understanding.

In terms of dataset related execution changes, one of the key changes involved the way data sets are constructed. As previously mentioned, early experiments relied very heavily upon external sources of images and metadata. These include large scale Unsplash crawling by categories with API rate limit management; date filtering client side; retry logic; and duplicate avoidance. To manage the scaling of external datasets and address the unreliability of public metadata, we transitioned to using our own gallery of 1,400 real images. This shift provides a more dependable source of truth and ensures a significantly more realistic environment for evaluation.

6.3 What Went Right

- The modular service boundaries enabled incremental upgrades without full rewrites.
- Queue based processing reduced API blocking and improved user-perceived responsiveness.
- Multi-tenant isolation logic was validated through scenario based testing.
- CI/CD automation improved reproducibility and reduced deployment fric-

tion.

- The hybrid retrieval redesign improved both usability and retrieval quality in the realistic gallery setting.

6.4 Problems Encountered and Mitigations

A primary difficulty was that the evaluation of retrieval quality did not reflect true user relevance. Even though public image collections are convenient for doing large scale evaluations (for example, Amazon’s ImageNet Collection), they often have weak/semantically poor tagging/metadata. Therefore it is hard to determine what constitutes “accuracy” when assessing how well a system can rank reranked candidates. This issue was somewhat mitigated by including additional evaluations using our gallery style image collection.

One of the other issues was that deep-VL-semantic-re-ranking systems are computationally expensive and therefore much too slow for use within an interactive consumer-facing environment. The reason for the slowness was that many potential image candidates needed to be fed into Qwen for each query (the number of candidates typically being 50+). The system architecture was redesigned to decouple deep semantic processing from the search phase by handling it asynchronously during the initial image upload. Consequently, while the indexing happens in the background, the retrieval process remains synchronous to provide immediate results when a user issues a query.

Fine tuning models may be another way to potentially improve retrieval performance. In our particular application however, fine tuning wasn’t feasible because of limited categories, limited computational capabilities and lack of sufficient labeled training data. As such, we chose to pursue changes based on either architectural changes or fusion of retrieval methods as opposed to pursuing model fine tuning.

In terms of operational feasibility, there were two issues related to performing batch image uploads. First, there were fairness and throughput issues like allowing one user’s large batch of images to monopolize all available processing capability. Second, providing some form of feedback to the user as quickly as possible after receiving a query was important to maintaining usability. The backend utilized round-robin scheduling of batches prior to queuing them up and sent both fast path and slow path queries for each batch in order to provide quick responses while continuing asynchronous processing in the background.

The last engineering related issue was how to efficiently move large amounts of images across different services. Instead of simply passing raw image binary payloads to workers or directly into queues, the backend would store all

images in AWS S3 immediately upon uploading and pass secure object links through the rest of the system. Then, the ML service would retrieve these images asynchronously and thereby reduce queue size and increase processing efficiency.

6.5 Teamwork, Leadership, and Collaboration

Team members coordinated through shared issue tracking, pull-request review, and role specialization. Leadership rotated by workstream (backend reliability, ML quality, frontend integration, and deployment), and cross functional debugging sessions were used to resolve integration issues quickly.

The collaboration process was especially important during integration of distributed components. Because the system involved backend APIs, queues, vector services, storage services, cloud GPU inference, and mobile frontend behavior, many issues only emerged end-to-end. These were resolved through repeated integration cycles, shared debugging sessions, and coordinated iteration across service boundaries rather than isolated component development.

7. Evaluation

Our Retrieval System's logic went through several iterations throughout the project. At each stage we evaluated the accuracy and assessed the feasibility of upgrades considering our computational and latency limitations.

7.1 Stage 1: Evaluating Fine-Tuning of CLIP

First of all, we had to evaluate the baseline model's performance on images that people are most likely to search in their galleries. Therefore, we chose the USED dataset that includes eight different categories; i.e., concert, graduation, mountain trip, meeting, picnic, sea holiday, ski holiday and wedding. After evaluating the model, the next logical step was to tune the model to perform better on these types of images. We LoRA fine-tuned openai/clip-vit-base-patch32 on USED's train split and compared it to its original version using a dedicated test split.

Stage	Accuracy	Correct
Baseline (Zero-Shot)	83.21%	46,593 / 55,994
Post-Tuning (LoRA)	91.56%	51,266 / 55,994
Improvement	+8.35%	+4,673

Table 1: CLIP fine-tuning results on the USED dataset.

The results of this study confirmed that a general multimodal dual-encoder can be fine-tuned for a specific use case and for specific queries. The original CLIP model is trained to map images of objects and textual descriptions into the same vector space while preserving semantic similarity. In our case, a person is unlikely to search for an image of an apple or a nasal spray or any other random objects. Popular search queries are mainly social events, pets, food, aesthetic images like sunset, nature, sea, beach, and more. In theory, it is possible to fine-tune CLIP to perform better on these specific categories. However, the research also demonstrated high costs of this method: tuning the model on just eight categories required fifteen hours with our limited equipment. To cover most of the search queries that different people might make we would have to gather data and tune for a far greater number of categories. Due to this resource barrier, we abandoned this method as a theoretical option and pivoted towards improving general accuracy with cost effective methods.

7.2 Stage 2: A Comparison Between CLIP and SigLIP

While CLIP is the foundational model for aligning multimodal data, we noticed that SigLIP, a model developed by Google, often outperforms its predecessor. This is mainly because SigLIP uses sigmoid activation function instead of CLIP's softmax. To test their performance for our use case, we obtained images from a large set of relevant categories using the Unsplash API. In addition, for each category, we selected one hundred images that were uploaded in 2026 to ensure that we are evaluating the models on images that they have not seen during training.

Model	Size (parameters)	Recall@1
CLIP_ViT_L_14	428M	0.843
ViT_L_16_SigLIP2_384	303M	0.883

Table 2: CLIP vs. SigLIP comparison on Unsplash images uploaded in 2026.

This comparative analysis showcased a clear superiority of the SigLIP model: CLIP scored 0.843 for Recall@1 metric with 428M parameters, while SigLIP achieved a score of 0.883 with 70% of CLIP’s size. Hence, SigLIP was chosen as the primary retrieval engine instead of CLIP for our system.

7.3 Stage 3: RAG-Type Reranking

Because fine-tuning was effective but unfeasible at scale, we decided to test if the retrieval quality can be improved using a RAG-type design. The newly proposed approach was to use SigLIP to fetch (filter) ten image vectors that are closest to the text query vector, and use Qwen3-VL-8B-Instruct, a multimodal LLM model, to rerank these ten images based on their relevance to the text query.

To test this new approach, we decided to gather our own dataset of 1,468 images taken from our team’s personal galleries. We chose to switch from Unsplash because public tags are often inaccurate and many photos appear too polished or professional for most users’ personal photo collections. Rather than labeling each image individually, we used Gemma (google/gemma-3-27b-it) to create text descriptions of images as a source of truth. We chose this model because it is designed to generate human-like descriptions. Also, we chose a different LLM model than the one employed in our system (Qwen) to avoid potentially biased results. Lastly, the size of the model (27B) minimizes hallucination risk. Nevertheless, we manually verified that all of the descriptions are accurate and edited them whenever necessary.

Metric	Baseline SigLIP	RAG (Qwen reranker)
Recall@1	0.8011	1
Recall@5	0.9748	1
Recall@10	0.9946	1

Table 3: Retrieval results on the personal gallery dataset (1,468 images).

When evaluated on these realistic images from our galleries, the standalone SigLIP model scored 0.8011 on Recall@1. This is a significant 8% drop from our previous results when evaluating images from Unsplash. Nonetheless, it was expected because Unsplash images were very professional while our images better represent an average person’s gallery. Nonetheless, when we used Qwen to rerank the results, it perfectly matches all of the images to their descriptions. This was an outstanding result. Unfortunately however, due to the complexity of additional reasoning applied at the user query time, retrieval could take

anywhere between 20 seconds to 50 seconds even using flagship GPUs. On our limited hardware, moreover, a single query took at least 2 minutes, which is even slower than manually searching for the images in a gallery.

7.4 Stage 4: Final Weighted Retrieval Design

To solve this problem, we decided to move the slow LLM processing from the retrieval stage to the upload / pre indexing stage. Thus, when users upload their images, they can start using the application almost immediately with our fast standalone SigLIP retrieval logic while Qwen is generating descriptions and a set of tags for the images. As soon as it is finished, the system will automatically switch to a more accurate weighted retrieval strategy that incorporates three different signals in its final ranking. These include:

1. Cosine similarity between the vectorized image and vectorized search query (sim_1)
2. Cosine similarity between the vectorized text description of an image and vectorized search query (sim_2)
3. Number of exact lexical overlaps between the set of tags and the words that are in the search query (N)

Therefore, the final scoring function was:

$$\text{Score} = W_1 \cdot \text{sim}_1 + W_2 \cdot \text{sim}_2 + W_3 \cdot N \quad (1)$$

The weights were optimized using Optuna by maximizing NDCG. The final values were:

$$W_1 = 0.6936, \quad W_2 = 0.6681, \quad W_3 = 0.0064$$

Metric	Baseline SigLIP	RAG (Qwen reranker)	Weighted Retrieval
Recall@1	0.8011	1	0.998
Recall@5	0.9748	1	1
Recall@10	0.9946	1	1

Table 4: Full comparison of retrieval strategies on the 1,468-image dataset.

On the same 1,468-image dataset, the final weighted retrieval system achieved Recall@1 = 0.998, Recall@5 = 1.0, and Recall@10 = 1.0. Thus, our final weighted retrieval approach reached nearly the same performance as the RAG system while completely removing the additional LLM reasoning latency from the user end.

7.5 Interpretation

These studies resulted in several key findings. The first major finding is that fine-tuning a standalone model was possible to boost accuracy for a specific use case; however, it was far too resource demanding to cover most of the search categories that people could make. The second major finding is that SigLIP's multimodal encoding algorithm proved to be superior to that of CLIP's, especially for our specifics of dealing with small batches. Therefore, SigLIP became the primary model utilized by the final version of our system. The third major takeaway is that using a multimodal LLM (or vision language model) for greater semantic depth, we can achieve near perfect accuracy in image retrieval. However, it had to be implemented on an appropriate stage of the pipeline to not burden the user with additional latency.

8. Conclusion and Future Potential Research

The initial CLIP-based retrieval prototype made in Fall 2025 evolved over time into a much fuller distributed framework for finding images within personal photo galleries. One of our primary lessons from the first six months of the project was how important it is for the model and its components to both contribute equally toward better retrieval performance. We tested using VLM-based rerankers at query time to potentially increase the accuracy of those images ranked highest by the original model. However, due to performance issues, reranking at query-time was unacceptably slow to be practically useful. Due to these limitations, we moved most of the deeper semantic processing into the earlier stages (pre-indexing) while keeping fast image representations for real-time searchable applications. Ultimately, we created a working system that was both highly accurate and practical. Asynchronous enrichment may represent a more viable long-term approach to achieving comparable semantic understanding in conjunction with faster response times when compared to reranking at query time for personal photos.

We believe there are three major contributions made by this project.

1. A fully functional end-to-end multimodal retrieval architecture.

2. An experimentally supported analysis of the impact of multitenant delete cascades and all associated operations.
3. Quantitative comparisons of different retrieval stacks.
4. Patterns and implementations of deployment and automated workflows that will serve as a foundation for possible future improvements.

There are still some directions for future work regarding the system developed this semester. Some examples are completing account management and hardening, adding full authentication, conducting larger user studies, relevance tuning based on feedback from users, and developing formal bias and safety evaluation protocols. We could also keep improving UI/UX for better accessibility and work on the system to meet production ready status including support for autoscaling and observability dashboards.

Another potential area for future research includes development of larger, more diversified evaluation datasets based on human annotated relevance judgments instead of metadata-derived labels. Another example is exploring alternative fine tuning methodologies including supervised or preference-based fine tuning, however this would require more of labeled data along with additional computational resources than were accessible throughout this project. Lastly, adaptive weight strategies for different types of user queries could also be a direction we could take this project further in.

References

Datta, R., Joshi, D., Li, J., & Wang, J. Z. (2008). Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys*, 40(2), Article 5.

<https://wang.ist.psu.edu/survey/analysis/datta07.pdf>

Dubey, S. R. (2021). A decade survey of content-based image retrieval using deep learning. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(5), 2687–2704.

<https://doi.org/10.1109/TCSVT.2021.3080920>

Johnson, J., Douze, M., & Jégou, H. (2017). Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3), 535–547.

<https://doi.ieeecomputersociety.org/10.1109/TBDATA.2019.2921572>

Liu, Y., Zhang, D., Lu, G., & Ma, W.-Y. (2007). A survey of content-based image retrieval with high-level semantics. *Pattern Recognition*, 40(1), 262–282.

<https://doi.org/10.1016/j.patcog.2006.04.045>

Malkov, Y. A., & Yashunin, D. A. (2018). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4), 824–836.

<https://doi.org/10.1109/TPAMI.2018.2889473>

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., & Sutskever, I. (2021). Learning transferable visual models from natural language supervision. *Proceedings of the 38th International Conference on Machine Learning*, 8748–8763.

<https://proceedings.mlr.press/v139/radford21a/radford21a.pdf>

Rui, Y., Huang, T. S., & Chang, S.-F. (1999). Image retrieval: Current techniques, promising directions, and open issues. *Journal of Visual Communication and Image Representation*, 10(1), 39–62.

<https://doi.org/10.1006/jvci.1999.0413>

Wang, P., Bai, S., Tan, S., Wang, S., Fan, Z., Bai, J., Chen, K., Liu, X., Wang, J., Ge, W., Fan, Y., Dang, K., Du, M., Ren, X., Men, R., Liu, D., Zhou, C., Zhou, J., & Lin, J. (2024). Qwen2-VL: Enhancing vision-language models' perception of the world at any resolution. *arXiv*.

<https://doi.org/10.48550/arXiv.2409.12191>

Zhai, X., Mustafa, B., Kolesnikov, A., & Beyer, L. (2023). Sigmoid loss for language image pre-training. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 11975–11986.

https://openaccess.thecvf.com/content/ICCV2023/papers/Zhai_Sigmoid_Loss_for_Language_Image_Pre-Training_ICCV_2023_paper.pdf