# Requirements

**Team:** INSIGHT
**Sponsor:** Mike Taylor
**Faculty Mentor:** Scott Larocca
**Team Members:**
Joshua VanderMeer, Michael Vertin,
Aidan Hebert, Forrest Hartley

**Version**: 1
**Date:** 11/15/2024

**Accepted as baseline requirements for the project:**

| | **Signature** | **Date** |
|---|---|---|
| **Client:** | _____ | _____ |
| **Team:** | _____ | _____ |

# Table of Contents

# 1 Introduction

Northern Arizona University is known for its academic legacy and one of the driving forces that has helped students over the years is the Cline Library. The Library here on campus is a tool that offers a plethora of resources for students, researchers, faculty, and any curious individuals of the community alike. This Library hosts an important resource for these individuals to use the Special Collections and Archives (SCA). The SCA collects and organizes important pictures and documents for research purposes. The SCA is a valuable resource, housing unique collections that chronicle the stories and culture of the Colorado Plateau. This group's goal is to help elevate research and help to provide insight into important pieces of the past.

Having a reliable tool for this collection has always been a struggle for the Special Collections and Archives team. This team has collected images for years, reaching a staggering 100,000 images and documents with more to add. Without a reliable tool to search with students are left just scrolling through the pages of listed images or trying to find them with manually inputted keywords. Our Sponsor Mike Taylor and the Special Collections and Archives team aim to develop an innovative "Image Similarity Search Tool" to improve the accessibility of the digital archives. The tool requested was an image similarity searching tool to help boost the efficiency of any user trying to find images from the collections. These collections are housed on a database CONTENTdm which provides access to these images through their website page. With this new solution, users would be able to use their images as a search query. The system would then analyze the image and return the top results that are most similar to the SCA's collections displayed. For example, a user could input an image of the iconic old main building on NAU's campus, and the tool would retrieve all the similar archived images and display them for the user to access and use.

Image similarity searching involves some advanced technical strategies like "image embeddings" or "vectorization" where an image is transformed into a mathematical representation. These Vectors are based on metadata, visual features, or both and are going to be generated in multiple dimensions. Once these vectors are generated they enable similarity comparison such as Cosine Similarity. This tool is used to identify and rank images based on their resemblance to the imputed image by the users. This project will then deliver a responsive,

web-based application, presenting search results in a grid layout. Users then can interact with the results and view the images and any connected data in the database.

In addition to enriching the user experience, this project helps to move the library into a more modern and simplified search algorithm to help users discover digital resources from their collections. The end goal product for this project is to have a fully developed and tested application, a detailed set of documents and reports, and a well-documented codebase. This will ensure the tool is both functional and extendable after its creation. This project will also incorporate Web 2.0 programming, user experience design, and database management systems. Culminating is an impactful tool to bridge the gap between users and history through advanced image-searching technology.

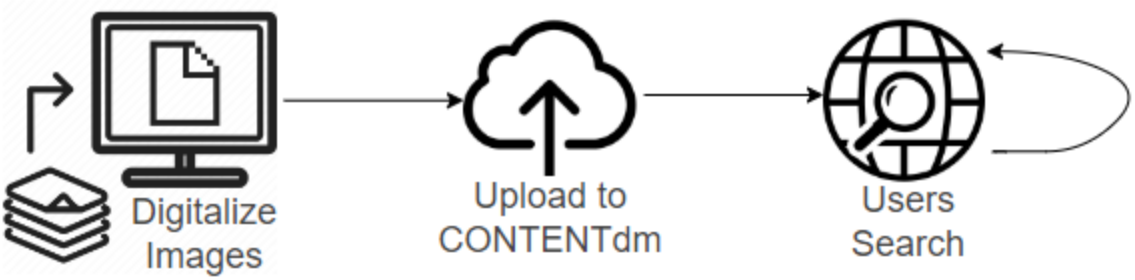
# 2 Problem Statement

The Cline Library's Special Collections and Archives (SCA) is dedicated to preserving and sharing unique historical resources related to the Colorado Plateau. They have done this through the digitalization of over 100,000 images and documents with access through CONTENTdm. Their current keyword-based searching tool falls short of user's needs for exploration of their collections. This limitation hinders users who may not know specific keywords or what they are looking for in full. Creating a new and intuitive tool could fix this problem for many users.

### 2.1.1 Key Workflow in Cline Library SCA Digital Archives

The primary workflow for the SCA's digital content management involves the following:

1. **Digitalization and Metadata Entry:** Physical items are digitized by hand and metadata (descriptions, keywords, locations, dates, etc.) is added manually for each piece added to the collection. This metadata is then used for the current text-based searching.
2. **Upload to CONTENTdm:** Digital files and metadata are uploaded to CONTENTdm, where they become part of the accessible online archive through the SCA's website.
3. **User Search and Retrieval:** Users then can access these images by either browsing different collections or keyword searching on the SCA's website. From here all images can be accessed and all associated metadata and information is located with the image.




While this process has allowed for the SCA to provide access to its digital collections and archives the reliance on text-based metadata and search restrictions gets in the way of the users trying to interact with the collection. This is preventing users from having a smooth and intuitive experience while browsing the SCA's collections.

### 2.1.2 Problem Overview

The Existing system's reliance on traditional keyword-based searching is limiting its uses and the following issues are seen in the existing system:

1. **Lacking Diverse Search Tools:** The platform is fully reliant on keyword searching and users spend time searching manually through pages of documents and images.
2. **Reliance on Accurate Metadata:** Search results are only as consistent and reliable as the quality of metadata for each piece of a collection. If the metadata isn't handled properly could lead to missed results from keyword searching.
3. **Non-Intuitive Discovery Pathways:** Users are constrained to keyword searching and the listed-based storage of the images this limits the potential of this tool and could lead to overlooking relevant images.
4. **High-Time Investment for Users:** With this existing system to find the most relevant image to a user it could take multiple inputs and considerable amounts of revised keyword searches just to locate an image. This could still lead to the potential of overlooking a relevant image.
5. **Lack of Image Grouping by Similarity:** There is no functionality to group or filter images based on visual similarity, which could allow users to explore clusters of similar images more efficiently. There are only groups based on the overall collection.

This Project proposes to develop an **Image Similarity Search Tool** to address the issues above and help optimize the user experience when working with the SCA's collection. By implementing an application capable of searching through the archives using image embeddings and similarity messages providing a richer user experience and higher quality results.

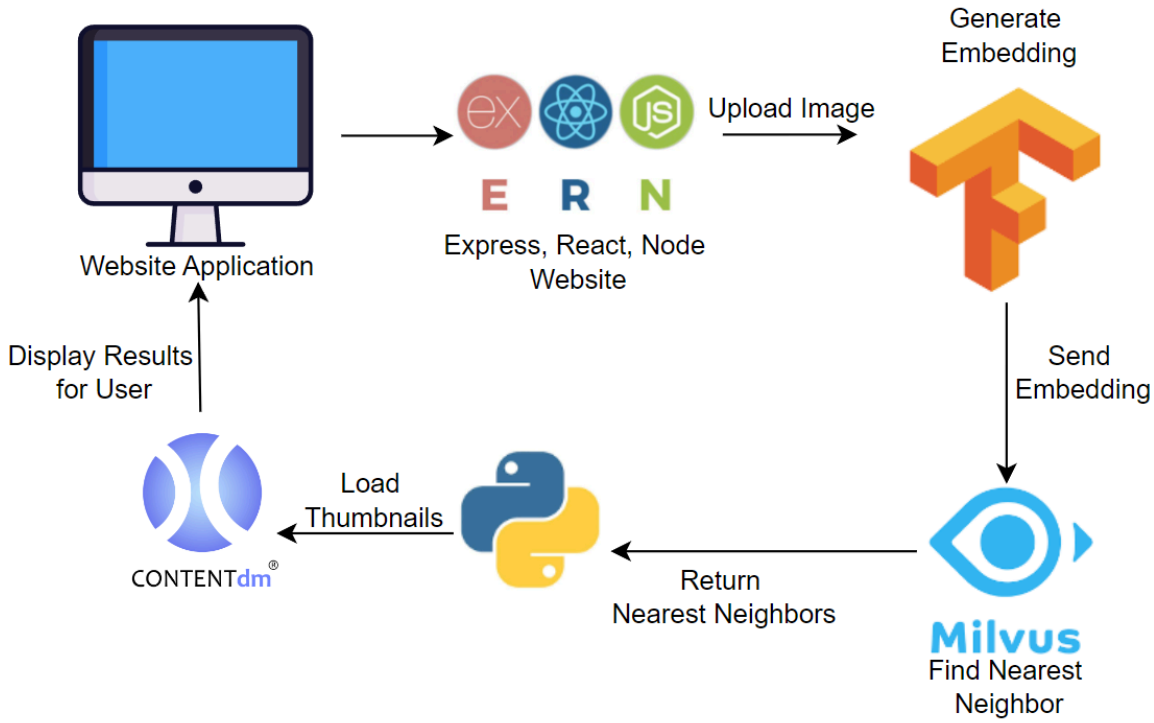
# 3 Solution Vision

We team INSIGHT plan to create an Image Similarity Searching Tool that will help to make the SCA's collection a more intuitive experience and help users find the images they need. We want to make an easy-to-use website application that users can insert an image into and quickly and reliably receive similar images from the collection. After this program finishes running and processing the user input it should be able to show the nine most similar images to the user's image-based query. This will take the time and effort out of searching for images for all users.

### 3.1.1 Features

- Simple graphical user interface that is intuitive and easy to use for all users.
    - Easy image upload process
    - Responsive results
    - Easy to access results for users' query
- Software should determine:
    - Imputed images Embeddings
    - Nine most similar images
    - Location of those Nine images in CONTENTdm
    - Return Thumbnails for users to be able to view
- Powerful and efficient Embedding generation
    - Using TensorFlow to generate embeddings
    - Generation done promptly
- Interactive Grid of images as a result
    - Similar images are displayed in a user-friendly manner
- Modern and Responsive Website
    - Using Web 2.0 frameworks like React and Express
- Extendable Codebase and Documentation

- Well-documented code base
- Documentation on how to use and work with the code base
- Documentation to support maintenance and updates




# 4 Project Requirements

The project requirements section will describe what capabilities the team and client expect from the system. The functional requirements section describes the responsibilities of each component and their interactions in the system. The performance requirements section describes specific metrics that the system must meet. The environment section describes what hardware and software tools will be used to contain the system. Together, these requirements are a high level description of the system's responsibilities, providing a concrete understanding of the team's and client's expectations.

## 4.1 Functional Requirements

The requirements in this section will describe the responsibilities and dependencies of the different parts of the system.

- ### 4.1.1 Embedding Generation
    - **Access the Model**

      A consistent machine learning model is required for both the searching and storage components. For this reason, the retrieval of the model will be handled in the embedding generation component to ensure the same model is used throughout the deployment. A pre-trained model can be attained through HuggingFace and TensorFlow and stored on the server. When the team gains access to the SCA's database, they can fine-tune the model or create a new model using the SCA's sources. This will be discussed further in 4.1.2.
    - **Calculate embeddings**

      The purpose of a machine learning model is to generate embeddings, which represent the model's interpretation of an image. This component will be responsible for returning the embedding of a given image.
    - **Supported image types**

      Currently, the SCA's database only consists of .jpg images. This image type will be required for this project. However, users may want to upload images of any type. We currently plan to support at least .png, .jpg, and .jfif types and will expand this as we learn the limitations of each tool.

- ### 4.1.2 Embedding Storage and Searching
    - **Store embeddings**

      Vector embeddings are a machine's interpretation of an image. A requirement for this system is the ability to compare images from external sources to images inside the SCA. For each image in the SCA, a reference to the image must be saved alongside its embedding. This will be accomplished with a vector database, which is used to store a variable number of embeddings.

- **Initial Storage**

  When the system is deployed, it should contain all images in the collections and archives. This process can be automatically handled by INSIGHT's code any time after the model's training has ended.

- **Future Storage**

  After the initial storage, the SCA will continue uploading thousands of images. To ensure this information is accessible through the system, all of these images must be entered into INSIGHT's vector database. If this process can be automated, the SCA will not need to interact with the system. Otherwise, if the team is unable to automate the storage of new images, the SCA will need to manually interact with the system to stay up-to-date. The team will be required to implement at least one of the following options:

  - **Insert new images**

    For this option, the SCA can enter one or more images at a time. User experience will not be impacted during this process. However, this will result in inconsistencies if one or more images are not entered. This process should be secure and straightforward to prevent unwanted users from adding images and minimize the time to learn and perform the insertions.

  - **Reload database**

    This option would automatically update INSIGHT's database to match the SCA's database. This will require access to the SCA's database during the update. It will also involve observing every image in the SCA, potentially affecting user experience. After the database is reloaded, it will be up to date with the images in the SCA.

- **Search for images**

  The purpose of this component is the main task of the project: searching for similar images. This component is required to identify several images that are similar to any given image. Any two embeddings can be compared using a

distance measurement such as Euclidean Distance or Cosine Similarity. Using the embeddings that have already been stored in the vector database, the most similar images are the ones that have the smallest distances from the uploaded image. However, comparing the input to every image in the SCA can exceed the requirements set in 4.3, especially as the SCA grows. An optimized indexing tool can be used, such as FLAT or IVF_SQ8.

An additional expectation for the system is the ability to search for images that are similar or dissimilar to a variable number of images. There is not a plan for implementing this yet. However, there are several options that could accomplish this:

- **Combine into one embedding**

  One way to implement variable input searching is to combine the embeddings before performing the search. Because only one search is performed, this is the fastest option. However, using one search will reduce the user's control over searching.

- **Implement into distance algorithm**

  Another option is to change how distances are calculated. Instead of minimizing the distance between the input and results, this would minimize the distance between similar images and the results and maximize the distance between dissimilar images and the results. This option would increase the user's control but would take significantly longer to execute.

- **Perform separate searches**

  The third option is executing searches for each image, and finding which results are shared, and not deselected. However, the total similarities and dissimilarities will be represented by the results. Because this performs a search for every input, it is not time-efficient. If no inputs share similarities, the result would be a combination of individual searches instead of a mixture of each input.

- **Retrieve image reference**

  A list of embeddings will be returned from a similarity search. The meaningful

information of these embeddings is the images they represent, so this component will be responsible for returning the image reference that each embedding represents.

- ### 4.1.3 Website
    - **Hosting**

      In order to give users access to the system, it must be available online. This will be done on the NAU Cline Library website.
    - **Upload image**

      In order to search the collections and archives, users must have the ability to upload image(s) that they have access to. Because an embedding must be generated for the user's images, the image type restrictions for uploading images will match the restrictions from 4.1.1. After an image is uploaded, the image will be temporarily saved on the client or server such that the user does not need to upload any image multiple times.
    - **Selection**

      Before searching, users may want to upload multiple images to better describe the image they are looking for. This will be split into two categories: similar and dissimilar. A similar image will look similar to what the user wants. A dissimilar image will not look similar to what the user wants. A user will have the ability to move any image from their uploads or search results into either a similar or dissimilar category. This information will be considered in the search process, allowing users to fine-tune the search results to match their preferences.
    - **Search for Similar Images**

      After images are selected, a search can be executed.
        - **Process**

          The website will be responsible for providing the backend with a list of similar and a list of dissimilar images. After processing the request, the backend will return a list of image references.
        - **Display**

          After processing, the list of image references will be displayed to the user.

Each image reference will be converted to an image and displayed in a 3x3 grid and ranked from most to least similar in left-to-right then top-to-bottom order.

- ### 4.1.4 SCA Database
    - **Retrieve image references**

      The vector database will contain references to every image with their corresponding embedding. Due to memory limitations discussed in 4.3, images will not be stored in the vector database to avoid duplicate information.
    - **Get an image from its reference**

      In order to generate embeddings and display search results, any image source must be attainable from its reference. This requirement could be met in one of two ways. The first is the reference having direct access to its image source in the SCA. The second requires access to the SCA database. The reference would be provided to the SCA database, which will return the image.
    - **Get an image's data from its reference (optional)**

      The current search implementation brings users to a webpage description of the image. If an image reference could get this data, or a link to the webpage, a similar format can be followed when a user selects an image.

## 4.2 Performance requirements

Now that the main functional requirements for this project have been established and fleshed out, we can direct our attention towards the non-functional requirements concerning performance. These requirements will detail the performance expectations that we have for our functional requirements. Minimum computational times, data storage expectations, UX expectations, and function maintainability and reliability will all be covered in this section.

### 4.2.1 Computation Times

In order to provide a seamless user experience, it's crucial that the technologies that we employ can respond within a reasonable time frame. The computational times for this project include the time taken to upload images,  the time taken to generate vector embeddings, the time taken to carry out similarity searches, and the time taken to retrieve and display results. Our web app will need to carry out processing and displaying operations responsively to allow for real time image traversal while still returning a grid of images that are relevant to the given query. Our aim here is to provide the user with accurate results within 3 seconds of image upload.

#### 4.2.1.1 Image upload time

Our expectation here is that the system will allow users to upload images quickly to reduce waiting times before processing begins.

Performance Expectations:

- The image upload process should handle images of up to 10 MB in size within 5 seconds.
- The system should provide a responsive interface during uploads, possibly showing progress indicators to reassure users that their upload is in progress.
- Efficient compression or streaming methods may be implemented to optimize upload times.

#### 4.2.1.2 Embedding generation time

Once one image has been uploaded to the web app, its vector embedding must then be generated.

Performance Expectations:

- The calculation process should utilize efficient machine learning, and image processing techniques to allow for timely results. Our aim here is to return a result back to the user within 3 to 5 seconds of them querying their initial image.
- Depending on the model that we use for embedding generation, optimization techniques such as quantization or the use of faster inference frameworks should be taken into consideration.
- Vector embeddings will be generated ahead of time and stored for all pre-existing images local to the database.

#### 4.2.1.3 Similarity Search Time

Image similarity search methods should be performed promptly to accurately retrieve relevant images. Cosine similarity searching will be used here.

Performance Expectations:

- Our similarity search must return the top-K most similar images from the database within 1 second.
- An optimized indexing tool such as FLAT or IVF_SQ8 can be used here to improve efficiency as mentioned in 4.1.2.
- We may utilize caching here to maintain smooth performance.

#### 4.2.1.4 Concurrency Handling

Our system should be able to handle multiple concurrent users querying for images without noticeable drops in performance.

Performance Expectations:

- Our web app must support up to 50 users performing searches simultaneously while maintaining smooth and consistent performance.
- Unexpectedly large volumes of requests outside of the comfortable range should be handled with grace, to allow for sustained functionality, even if only partial.

#### 4.2.1.5 Performance Monitoring

Continuous metrics for system performance should be collected to gauge performance regularly. This will provide insightful data that can be used to assess where inefficiencies might lie.

### 4.2.2 Data Storage

Efficient and reliable data storage is an essential part of our web app. It's important that we ensure quick access to both the image embeddings and the images themselves.

#### 4.2.2.1 Efficient Data Storage

The system must store all images and their vector embeddings efficiently to facilitate fast and efficient retrieval and processing.

Performance Expectations:

- Embeddings should be stored in a high-performance database or vector search engine designed for rapid similarity queries.

- All database retrieval operations for any given query should be performed within 500 milliseconds.

#### 4.2.2.2 Data Integrity

All stored images and embeddings must be maintained reliably with correct linkage and minimal possibility for corruption risk.

Performance Expectations:

- When an image is stored, we must ensure that its embedding and metadata are linked correctly, and consistent with our expectations.
- Regular backups and redundancy measures should be conducted to prevent data loss.
- Checks should be implemented to look for and identify any corrupted data, or mismatched images and embeddings.

### 4.2.3 UX design

The frontend of the website is the aspect that the user interacts with when using the tool, so it should be presentable, intuitive, and responsive.

#### 4.2.3.1 Responsiveness

Performance Expectations:

- Actions such as image uploads and image search queries should give immediate user feedback such as loading spinners or progress bars to inform the user that the system is working, and inform the user of the loading times.
- Pages should load within 2 seconds, and interactive elements should respond quickly to user input.

#### 4.2.3.2 Clarity and Simplicity

The user interface should be clear, simple, and intuitive.

Performance Expectations:

- When necessary, clear instructions and tooltips should be provided to help any users that may be struggling. Any constraints that the user should be aware of such as compatible image types, or size will need to be provided.
- All elements that are of the same type on the webpage, should appear and function the same, to avoid any confusion.

#### 4.2.3.3 Accessibility

Our web-app should be accessible to all users, and make accommodations accordingly.

Performance Expectations:

- Our app should be accessible on a large range of screen sizes to account for mobile and wide screen environments.
- Users with connectivity issues should be accounted for. This means that less intensive loading strategies should be enforced when a slower connection is detected.
- Our website should follow accessibility standards to account for users with disabilities.

### 4.2.4 Reliability

Our web app should be consistent and dependable. All image search features should perform as expected under varying conditions.

#### 4.2.4.1 System Uptime

It goes without saying that our web-app should have high availability, with very little downtime.

Performance Expectations:

- Our system should maintain an uptime of 99.9%.
- Intentional redundancy, and failsafes should be included to avoid single point failures.

#### 4.2.4.2 Error Handling and System Recovery

Errors and exceptions should be handled gracefully. Additionally, our system should be easily recoverable in the event of a system failure.

Performance Expectations:

- Informative messages should be provided for all exceptions
- A log should be kept of all previous errors to help with future diagnostics.
- Retry logic should be implemented for certain systems such as data transmission.

### 4.2.5 Maintainability

Maintainability requirements will focus on the ease in which updates, modifications, and extensions can be performed on the system in the future.

#### 4.2.5.1 Documentation and code Readability

Our codebase should be well structured and readable with provided documentation in place. Best coding practices should be utilized and new code and technologies should be implemented with future-proofing in mind.

Performance Expectations:

- Code should be modular, and developers should follow agreed upon standards for code layout and function/variable naming.
- Relevant documentation should be provided for code, APIs, dependencies and other technologies.

#### 4.2.5.2 Scalability

The web-app's architecture should be designed to allow for new features, and increased traffic and data volumes.

Performance Expectations:

- Chosen technologies should be widely supported, and easily upgradable if possible.
- New systems such as similarity searches or vector embeddings should be easily swappable with little modification.

## 4.3 Environmental Requirements

### 4.3.1 Introduction

Throughout the planning and development of any system, it is always important to consider the environmental requirements that will be at play. These requirements help better understand the underlying issue at hand. Additionally, our project requires unique accommodations as it heavily relies on machine learning and large data sets. For the scope of this document and our project, we will be discussing some of the inherent platform limitations as well as the limitations or potential limitations regarding the hardware and software systems behind the system.

### 4.3.2 AWS platform limitations & requirements

1. **Image Storage & management**

   This system is heavily reliant on the ability to efficiently traverse and process large quantities of data to serve accurate results to the end user. To accomplish this, **AWS S3** will be crucial in dealing with the storage requirement needed to hold the large size of photos. Additionally, To effectively retrieve results, some image meta-data may need to be stored in one of AWS's provided database solutions including **AWS RDS (Relational Database Service).** Given these solutions, it's important to note that large scale storage costs can add up quickly in AWS.

2. **Image feature extraction**

   The core process involves the extraction of image features which are then needed to form comparisons. This process of extraction can be very resource intensive, specifically in regards to the GPU. AWS provides numerous resources that allow for high powered GPUs to be leveraged in projects. However, it's important to note that this will come at an additional expense as GPU instances can be particularly expensive.

3. **Image feature comparison**

   Another core function to our system is the ability to rapidly compare results extracted from each of the images to form groupings and similarities. This involves some computationally intensive algorithms and search methods including NNS (Nearest Neighbor Search). While not nearly as costly as high power GPU instances, the number of comparisons can also quickly add up in terms of cost. It's important to optimize the number of calls necessary to assist with cost.

# 5 Potential Risks

### 5.1 Introduction

To successfully ensure that system requirements are met, it's important to note and address potential issues that may arise within development and implementation of said project. While some of these issues possess no perfect solution, having a better understanding of the issue can help mitigate its negative effects on the system.

### 5.2 System Scalability

When dealing with large data sets that aim to be accessed frequently, developing a scalable solution remains a crucial issue to be tackled. Given this system, ensuring that the underlying infrastructure is capable of growing and expanding as the image set expands is crucial to ensuring the system serves its intended purpose.

### 5.2 System Access & Security

Security and Access remains an integral pillar within any system. For our system, we will specifically be dealing with important collections of images as well as user submitted content. ensuring that both the user and the data the user is accessing is secured properly remains a crucial issue to handle and ensure throughout development.

### 5.3 Biases within training data

Biases present within trained machine learning data has long been a known issue with the community. Within the specific scope of image similarity, these biases will most likely manifest at two different points within the system. The first being bias within the image feature extraction and the second being in image selection & comparison. Both of these issues can be mitigated through the use of a more diverse dataset that accounts for greater variation and possibilities.

### 5.4 The Black Box Issue

The black box issue refers to the inherent lack of transparency when dealing with machine learning models and more specifically, deep learning networks. Within the context of our system, these deep learning networks aim to be extremely effective; however, tracing and understanding how certain decisions are made can be very difficult. This may manifest in the form of not understanding why certain images are considered similar to the system and not to us. While there is no guaranteed solution to this issue, it can be mitigated in a handful of ways including the use of model explainability techniques such as Grad-CAM.

# 6 Project Plan

### 6.1 General Plan of execution

In regards to this system, we aim to keep a clear plan of execution that will assist and aid with the development and implementation of said system. This plan of execution is a general set of objectives that should be met in order to ensure the system is executed correctly. These objectives include a set of **General Milestones** to be completed, **clear and consistent communication** with our client to ensure the system stays faithful to original conception and doesn't stray off the path, and finally a feedback driven approach to assist with any implemented systems to ensure usability is not lost.

### 6.2 General milestones

**Initial Tech Demo**

The initial Tech Demo aims to serve as nothing more than a brief proof of concept regarding a piece of what will eventually be the entire system.
December, 2024

**Successful Image Retrieval & Processing**

This milestone hinges on our ability to successfully retrieve any image(s) from the existing AWS bucket and add image(s) to the image set that the model utilizes and processes.

February, 2025

**Successful feature comparison between user submitted images**

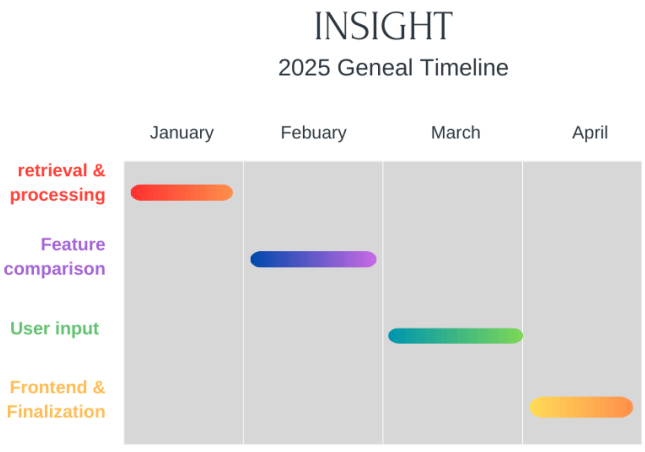
The successful feature comparison between user submitted images hinges on not only the successful implementation of the prior milestone but also on the being able to process a user submitted image. Additionally, this user image must be able to be successfully compared to any and all images within the trained data set to allow for proper comparison and similarity searching.

March, 2025

**Finalized Frontend**

The finalized frontend milestone aims to be reached in the final stages of this development process as it will most likely require the most user-feedback and iteration. The final frontend implementation should be a refined interface to support any described functionality while remaining lightweight and malleable for future expansion and updating.

April, 2025



# 7 Conclusion

The **Image Similarity Search Tool** project addresses a significant challenge the NAU Cline Library is facing with the SCA's collection. The use of traditional keyword-based searching and how it limits users' ability to fully explore the rich visual history contained in the archives must be addressed. By providing the image-based searching solution we hope to offer users a more intuitive and effective way to discover similar items and enrich research opportunities in the future.

The core problem stems from the limitation of the current SCA's website and how its fully reliant on metadata and text searching. This restricts what users will be able to interact with and slows down the process of searching for information and images for research purposes. Our solution, a web-based application that allows users to search the archives using image-based queries will help enhance the user's experience. This application will make use of modern machine learning techniques to generate and compare images using embedding and present the images that are most similar responsively and simply.

In this document, we outline the project requirements, detailing the functional and performance expectations necessary for the successful completion of this project. We also identified potential risks that come with this project and how we will make this product reliable and expandable in the future.

This project will progress steadily and with the foundation now set efficient image similarity searching within the SCA will be possible. We intend to bridge the gap between the SCA's vast historical archive and the students, faculty, and researchers here at NAU.