User Manual

Version: 1.0
Team: INSIGHT
Sponsor: Mike Taylor
Faculty Mentor: Scott Larocca
Team Members:
Joshua VanderMeer, Michael Vertin,
Aidan Hebert, Forrest Hartley

Date: 5/9/2025

1. Introduction

This user manual aims to serve as a comprehensive guide to installing, configuring, and using the SCA Image Search Tool for the Cline Library Special Collections and Archives (SCA) at Northern Arizona University. Designed to streamline the process of identifying and researching archival images, this tool integrates modern machine learning techniques with a user-friendly web interface. We thank you for using our tool and hope it offers years of service for the SCA and Cline Library!

The manual is divided into two main usage sections: **Non-Admin Use,** and **Admin Use**. The Non-Admin section explains how everyday users, such as researchers, librarians, and staff, can access the web application, conduct image searches, and understand the tool's functionality. The Admin section provides detailed technical instructions for system setup, including server installation, dependency management, microservice maintenance, and troubleshooting.

Whether you're a casual user exploring the large collection of archival images or an administrator managing backend services, this document will walk you through every necessary step to ensure successful deployment and continued system use.

2. Non-Admin Use

2.1. Accessing the website

This section is intended for end users such as researchers, librarians, and SCA staff who will interact and access the web interface with the Reverse Image Search Tool. No coding or backend setup is required to use the tool effectively. All that is needed is a device with internet access and a modern web browser (e.g., Chrome, Firefox, or Safari).

Open a Browser

Launch any modern browser (Google Chrome recommended for best performance).

Navigate to the Web Application

In the address bar, enter the following URL:
https://insight.library.nau.edu
This will direct you to the homepage of the Reverse Image Search Tool.

Upload an Image

- Click the **"Choose File"** button to select a local file from your device.

Run the Search
Once your image is uploaded and viewable in the viewfinder, click **"Search"**. The system will process the image and return a list of similar items from the SCA archive.

View Results

- Each result will display a small thumbnail of a similar image, allowing the user to interact with each result by clicking. This will then redirect the user to the ContentDM page, where the metadata can be viewed.

- You can scroll through these results or refine your search by uploading a different image or by enabling explore mode.

2.2. Explain functionality

The SCA Image Search Tool was designed to help users identify and explore archival items from NAU's Special Collections and Archives using visual similarity. Instead of relying on keywords or titles, users can upload an image and let the system find related materials by analyzing visual features. This approach is especially useful for unlabeled, misidentified, or visually complex items where text-based search may fall short.

Key Features:

- **Image Upload & Processing:**
  Users can upload an image from their local device. The tool automatically processes the image using a machine learning model, which generates a high-dimensional vector representing the visual content.

- **Visual Similarity & Non-Similar Search:**
  The vector created from the uploaded image is compared against a database of vectors from SCA's digital image collection. The system calculates similarity using cosine similarity, then ranks and returns the closest matches. The opposite of this feature is also present, enabling users and researchers to see what may be considered the most "non-similar" to what they are seeking.

- **Search Results Page:**

    - Results are shown as image thumbnails with basic metadata (e.g., title, ID, collection).

    - Each result includes a clickable link that directs users to the item's full page in the ContentDM system, allowing access to full metadata and downloadable versions when available.

    - Results are displayed in order of relevance based on visual similarity.

- **Explore Mode:**
  Explore mode enables users to search with images they may have already found within the SCA collections. This feature can be enabled with the toggle on the main page. Once enabled, the user can click on any image returned from a prior search to now act as the new search. Preventing the user from having to download an image they found to use it in a new search manually.

- **Responsiveness & Accessibility:**
  The tool is designed to work on various screen sizes, including tablets and mobile devices. The tool can also have the number of images returned be dynamically altered via the slider on the home page.

Use Case Examples:

- A librarian finds a historic photo with no label and wants to identify similar images in the archive.

- A researcher uploads a map fragment to locate other geographic records with similar visual layouts.

- A student browsing historical photos uploads a unique image to discover more from the same era or photographer.

- A curious individual finds, photographs and uploads an image belonging to the SCA to pinpoint it in their collection and retrieve its item number and metadata.

3. Admin Use

How to install all dependencies on an EC2 instance

For the purposes of this section, the directory where Linux commands are run will be the project directory unless specified otherwise. (ie ~/myenv/INSIGHT-SCA-Search-Tool/)

3.1. Connect to Your EC2 Instance

```
ssh -i your-key.pem ubuntu@your-ec2-public-ip
```

3.2. Update Package Lists and Install Base Tools

- **Ensure apt is up to date** → *Needed to install all Linux packages*

```
sudo apt update
```

- ***python3-pip*** → *Needed to install Python packages.*

```
sudo apt install -y python3-pip
```

- ***python3-venv*** → *Needed to create virtual environments.*

```
sudo apt install -y python3-venv
```

3.3. Create a Python Virtual Environment

- Create a folder **myenv/** in your home directory.

```
python3 -m venv ~/myenv
```

3.4. Activate the Virtual Environment

```
source ~/myenv/bin/activate
```

- You should now see (myenv) in front of your terminal prompt.
- **(Important!)** From now on, always activate this before running Python code!

3.5. Upgrade pip (optional but recommended)

```
pip install --upgrade pip
```

3.6. Install Required Python Packages

Module	Why it's Needed	Install via
flask	Running the Flask apps (vectordb_app.py, service_control_app.py)	pip install flask
flask_cors	Cross-Origin Resource Sharing (CORS) in Flask apps	pip install flask-cors
pymilvus	Connecting to Milvus vector database	pip install pymilvus
transformers	Loading Vision Transformer model (embedding_model.py)	pip install transformers
torch	PyTorch (needed for inference with ViT model)	pip install torch
pillow(PIL)	Image loading and manipulation (image_reference.py)	pip install pillow
boto3	Fetching images from AWS S3 buckets (image_reference.py)	pip install boto3
requests	Fetching images via HTTP (e.g., CDM image loading)	pip install requests

Module	Why it's Needed	Install via
flask	Running the Flask apps (vectordb_app.py, service_control_app.py)	pip install flask
numpy	Array manipulation and vector normalization	pip install numpy

Now, install all source code using whatever method you have access to into the /myenv directory.

How to install all Node packages

Step 1: Check if Node.js is Already Installed

Run this in your terminal:

```
node -v
npm -v
```

If you see the version numbers, Node.js and npm have already been installed. If not, proceed to step 2.

Step 2: Download and Install Node.js

Option A: Install from Node.js Website (Recommended for Most Users)

1. Go to: https://nodejs.org
2. Download the LTS (Long Term Support) version for your OS (Windows, macOS, or Linux).
3. Run the installer and follow the steps. Make sure to check the box that says "Add to PATH".

Option B: Install via Command Line (Linux/macOS)
For Ubuntu/Debian:

```
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -

sudo apt-get install -y nodejs
```

Step 3: Verify Installation

After installation, run:

```
node -v
npm -v
```

to verify that node and npm have been installed successfully

Step 4: Install all Node Packages

Go to your project directory website/:

```
cd path/to/your/project
```

If your project has a **`package.json`**, just run:

```
npm install
```

Go to your project subdirectory website/client:

```
npm install
```

Go to your project subdirectory website/server:

```
npm install
```

Step 5: Build Frontend application (for first install or changes to the code)

Go to your project subdirectory website/client:

```
npm run build
```

Once the code base is installed on the EC2 in the myenv/ directory

Config .env file for settings

Must add a file named .env to the client folder containing these two lines of code:

```
REACT_APP_IP=IP_OF_EC2_INSTANCE
IP=IP_OF_EC2_INSTANCE
```

Important: Replace IP_OF_EC2_INSTANCE with the real IP address of the EC2 instance.

Configure Certbot settings

If you haven't already:

```
sudo apt update
sudo apt install certbot python3-certbot-nginx
```

Complete all steps in the **Config file for NGINX settings**

Run this one-liner (no need to stop Nginx):

```
sudo certbot --nginx -d insight.library.nau.edu
```

Certbot will:

- Generate and install your SSL certificate
- Automatically update your Nginx config
- Enable redirection from HTTP → HTTPS
- Set up auto-renewal

When prompted:

- Choose to redirect all traffic to HTTPS (usually option 2)

Auto-Renew Check (optional)

Certbot already adds a cron job, but to test it:

```
sudo certbot renew --dry-run
```

Config file for NGINX settings

Configure

```
sudo nano /etc/nginx/sites-available/insight
```

Replace with:

```
server {
    server_name insight.library.nau.edu;

    client_max_body_size 20M;

    root /home/ubuntu/myenv/INSIGHT-SCA-Search-Tool/client/build;
    index index.html;

    location / {
        try_files $uri /index.html;
    }

    location /api/ {
        proxy_pass http://localhost:5000/;
        proxy_http_version 1.1;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    listen 443 ssl; # managed by Certbot
    ssl_certificate
/etc/letsencrypt/live/insight.library.nau.edu/fullchain.pem; #
managed by Certbot
    ssl_certificate_key
/etc/letsencrypt/live/insight.library.nau.edu/privkey.pem; # managed
by Certbot
```

```
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by
Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by
Certbot

}
server {
    if ($host = insight.library.nau.edu) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


    listen 80;
    server_name insight.library.nau.edu;
    return 404; # managed by Certbot
    client_max_body_size 20M;

}
```

Test and restart Nginx:

```
sudo nginx -t
sudo systemctl restart nginx
```

Set up Docker:

Install docker from https://docs.docker.com/engine/install/

Installing PM2 and checking logs

```
npm install -g pm2
```

List and access logs

```
pm2 list
pm2 log <PM2_ID>
```

Managing Microservices

Starting the system:

```
cd into the base of the project
pm2 start python --name "service-control" -- service-control/service_control_app.py
```

Components:

There are four microservices in the system:

1. vectordb: Controls high-level interactions with the vector database
2. website: Hosts the UI that researchers will use to search through the SCA collections
3. milvus: Provides access to milvus' database
4. service-control: Keeps all other services running

scripts/services.py provides quick access to restarting or stopping the microservices (excluding service-control)

Restarting a microservice:

This will stop the microservice if it is already running, then start that microservice. This operation can take 1-2 minutes to fully complete.

```
python scripts/services.py reset <component_name>
```

Stopping a microservice:

To stop the microservice and prevent it from automatically restarting, run:

```
python scripts/services.py kill <component_name>
```

The following shows how to undo the previous command. This operation can take 1-2 minutes to fully complete.

```
python scripts/services.py unkill <component_name>
```

Getting the status of a microservice:

This will display if the microservice is currently running, and if it has been killed (see stopping a microservice)

```
python scripts/services.py status <component_name>
```

How to insert images into the vector database

Uploading the metadata file

Format: Metadata files are expected to be a tab-delimited file with the ID of the item in the 66th column of each row. For example, if the ID was 1384, it would be used to reference the item at https://cdm16748.contentdm.oclc.org/digital/collection/cpa/id/**1384**.

Uploading: Any uploading application can be used to transfer the metadata file from a local machine to the EC2 instance. The metadata file should be uploaded into ~/myenv/INSIGHT-SCA-Search-Tool/metadata/

```
python scripts/services.py status <component_name>
```

Inserting the metadata

```
python scripts/insert_cdm_metadata_file.py metadata/<file name>
```

Note: This inserts metadata in batches of 1024 items, and each can take 20-60 minutes. If machine resources are too low, the process may be delayed to give Milvus more time to compress its content.

Viewing the insertion progression

```
python scripts/data_check.py
```

The “num_entities” field is the amount of items currently in the database.
The “remaining batches” field is the amount of batch files waiting to be processed. When this number is 0, all batches have been inserted. Note that this does not include failed metadata.

Failed metadata

```
vim insert-service/metadata/failed-metadata.txt
```

If failed-metadata.txt does not exist or is empty, this means all items were successfully inserted.

The main causes of metadata failure are:
1. The CDM ID does not point to a valid item in the ContentDM. It must be possible to get an image using this ID. For example, 1384 would be used to get the image at https://cdm16748.contentdm.oclc.org/utils/getthumbnail/collection/cpa/id/**1384**.

2. ContentDM is not responding to HTTP requests. This is likely the reason for a large number of items present in failed-metadata.txt.

The insert-service will automatically attempt to reinsert the metadata in this file after all other batches have been inserted.

Expanding the database

Data persistence

After an item is entered into the database, that item will stay there until it is removed (through milvus_clear.py or restoring a backup).

Adding new data

To add additional items into the system, store the metadata for those new items into a new metadata file, and repeat the steps in 'How to insert images into the vector database' with that file. If one item needs to be added, the metadata file for that item should only contain one line.

Note that if any metadata in the new file matches existing items in the system, that metadata will not be inserted, although it will take time.

Resources

The first 120k items required ~675MB of disk space after automated compression. As more items are added to the database, an upgrade to available disk space will eventually be required. A backup will double the used disk space.

To view the memory used by milvus, run:

```
sudo du -h --max-depth=1
~/myenv/INSIGHT-SCA-Search-Tool/docker/volumes/minio/a-bucket/files | sort -h
```

There is a 2MB disk and ram availability check that must be passed before additional items are inserted. The intent is to reduce the chance of resource issues, but this lowers the amount of items that can be saved in the system.

The MIN_RAM_GB and MIN_DISK_GB constants can be found in insert-service/insert_app.py.

Creating a database backup

NOTE: Any uninserted metadata batches are at risk of being lost if the insertion microservice is inserting items after the backup is made. It is recommended that the insertion is done running when there is a backup to avoid missing metadata. (see “viewing the insertion progression”)

Stopping the milvus microservice

```
python scripts/services.py kill milvus
```

Wait 1-2 minutes to ensure Milvus has fully stopped.

Creating the backup

```
cd docker
sudo tar -czf volumes-backup.tar.gz volumes
cd ..
```

Restarting the Milvus microservice

```
python scripts/services.py unkill milvus
```

Using the database backup

NOTE: This will delete the current database. Any data inserted after the backup was created will not be recoverable after running these steps.

Stopping the milvus microservice

```
python scripts/services.py kill milvus
```

Wait 1-2 minutes to ensure Milvus has fully stopped.

Replace the current database with the backup

```
cd docker
sudo rm -r volumes
sudo tar -xzf volumes-backup.tar.gz
cd ..
```

Restarting the Milvus microservice

```
python scripts/services.py unkill milvus
```

Troubleshooting

Running out of memory

This might never occur due to memory checks and providing Milvus more time to compress data. Running out of disk space will often result in the EC2 instance crashing and the Docker microservice failing to start.

Stopping the milvus microservice

```
python scripts/services.py kill milvus
```

Deleting the index

```
sudo rm -r
/home/ubuntu/myenv/INSIGHT-SCA-Search-Tool/docker/volumes/minio/a-bucket/files/index_files
```

Restoring the index

```
sudo systemctl status docker
sudo systemctl start docker
cd docker
docker-compose up -d
cd ..

python scripts/data_check.py debug
collection.create_index("embedding", {"index_type": "IVF_FLAT", "params": {"nlist": 128}, "metric_type": "IP"}) # see vectordb-service/vector_database.py for the most updated index
collection.load()
exit()
```

No step in this process should take more than one minute. If it takes more than one minute to run or any error is experienced, restart this process. (In our experience this has never needed to be run more than twice.)

Rerunning the Milvus microservice

```
python scripts/services.py unkill milvus
```

4. Conclusion

We hope that this Reverse Image Search tool greatly enhances your experience and productivity at NAU's Cline Library Special Collections and Archives. Our goal was to deliver a user-friendly solution that leverages state-of-the-art machine learning technologies from HuggingFace and PyTorch to make accessing and exploring your extensive archives seamless and intuitive.

It has been our sincere pleasure and privilege to collaborate with you on this project. Our team is excited to have developed a tool that not only preserves history but also empowers researchers and supports the dedicated staff overlooking the Special Collections and Archives. We wish you many productive and insightful years using this solution and are confident that it will advance your mission of safeguarding and sharing invaluable archival resources.

While our team is moving forward into our professional careers, we remain available in the coming months for any brief questions or guidance you may need to ensure optimal deployment and smooth operation of the system within your organization.

With best wishes from your Reverse Image Search developers:

- Joshua Vandermeer – vandermeer.joshc@gmail.com
- Michael Vertin – msv68@nau.edu
- Forrest Hartley – forresth2000@gmail.com
- Aiden Herbert - arh752@nau.edu

Thank you again for the opportunity to contribute to the important work of the SCA.