Photography Agent Report

Rylee Albrecht, Douglas Glover, Jaden Barnwell, Matthew Peck, Jamal Mapp

1 Introduction

The photography agent is an application that provides an assistant to help a user organize and edit photos. The application uses an agentic agent, a large language model (LLM) that interacts with various models that it uses as ‘tools’. The LLM acts as the application’s ‘brain’ and interprets the user’s request to decide which tools are appropriate to use. Each of the tools is designed for a different task, including filtering photos by content, exposure, blurriness, and color. Some tools are used for generative edits to the photos. The application presents this process to the user as a standard LLM interaction.

2 Methods

This section describes the architecture and implementation of the system, including the user interface, the LLM agent, and the image-processing tools integrated into the application. Each component is detailed in the following subsections, covering design decisions, model training, and their roles within the overall application.

2.1 Interface (Douglas Glover)

The interface for the application is split into a front-end and back-end. The front-end is a user interface resembling a typical LLM chat interface. Users can send and receive messages that can have images attached. Users can swap between various past conversations using a side panel, and create new conversations at any time.

The back-end is where the user’s messages are given to the LLM to process. Here, the LLM response will either trigger a tool to be used or return a message to the user. If the LLM contains a trigger for using a tool in the form of JSON, the JSON is interpreted and passed to the appropriate tool. Once the tool has interpreted the images provided, the images/data is passed back to the LLM to provide a final response to the user. All of this data is saved in a local database, to be reused if the conversation continues at any point later on.

In short, the front-end acts as the user’s interface with the LLM and the back-end acts as the LLM’s interface with the tools. This setup allows the LLM to provide useful data to the user without the user directly interacting with any of the image-processing models.

2.2 LLM Agent (Rylee Albrecht)

The LLM agent is based on the Llama-3-8B-Instruct model and was fine-tuned on a curated dataset of 1,985 examples designed to teach the model to produce JSON-formatted tool-call outputs, list available tools when prompted, respond appropriately to greetings, and handle invalid requests. Fine-tuning was performed using LoRA with a learning rate of 2×10^{-4} for five epochs. A custom system message was incorporated during both training and inference to guide the model's behavior. During inference, the agent selects the appropriate tool to fulfill the user's request or prompts the user to rephrase their query if the request is invalid.

2.3 Focus Tool (Jamal Mapp)

The focus tool was evaluated using a collection of diverse images sourced primarily from the Unspalashed API, which provided high-resolution photographs across a wide variety of subjects.

The dataset was ideal for testing because it contains natural variation in lighting, texture, composition, and camera quality. The dataset allows the tool to encounter and identify high focus and sharpness in images.

The Focus Tool uses a hybrid approach. The primary model uses DeepLabV3, which is pretrained on the COCO dataset. DeepLab performs semantic segmentation to identify objects and foreground regions in an image. The laplacian variance is then computed within the segmented regions, providing a context-aware sharpness score. The secondary fallback is Spectral Residual Saliency. If segmentation fails or produces a poor mask, a saliency map is computed to estimate the focus area. The laplacian variance is then computed within the saliency mask as a backup focus score.

The hybrid design ensures that the tool can detect sharpness even when segmentation struggles, combining deep learning with classical computer vision techniques.

DeepLabV3 is pretrained, so no additional training was required. The fallback saliency method is algorithmic, so no learning or training is needed. Thresholds for acceptable focus scores were calibrated empirically on sample images from Unsplash dataset. This hybrid setup allows the tool to reliably classify sharp vs blurry region without the need to retrain.

The role of the tool is to filter out blurry or low-quality images from a given dataset of images. This ensures that downstream agents receive only high-quality inputs. The tool provides a numerical focus score for image ranking. The tool acts as a preprocessing quality check, forming the foundation for all other image based Ai tools in the project.

2.4 Exposure Tool (Jamal Mapp)

The exposure tool was evaluated using the same Unsplash API. It was chosen because Unsplash provides naturally occurring variations in lighting conditions. This diversity makes it ideal for testing exposure metrics since images differ widely in brightness distribution, highlights, shadows, and contrast.

The exposure tool is not deep-learning based. Instead, it uses a computational photography approach based on luminance. Each image is converted to the Y channel of the YCbCr color space, which isolates brightness from color information. This provides a simple but robust numerical way to classify exposure without training a neural network.

No training was required. Instead, thresholds for classifying exposure were tuned empirically using Unsplash dataset. Thresholds were validated by visually checking classification results and adjusting the luminance boundaries until the tool reliably differentiated between clearly dark scenes, clearly blown out scenes, and well-balanced exposures. Because the tool relies on ;uminance analysis rather than learning, it is fast, consistent, and easy to calibrate.

2.5 Color Tool (Jamal Mapp)

The color tool was developed and evaluated using the same Unsplash API. The dataset provides a broad range of environmental conditions, lighting types, and scene compositions. Its includes warm-toned images, cold-toned images and neutral images.

The color tool does not use a deep learning model; instead it uses a lightweight computational color analysis. Method based on the mean RGB channel intensities of an image. Each value is normalized between 0.0 - 1.0. The tool defines a user-specified range.

Min_val < R, G, B < max_val

If all three channels fall within this window, the image is considered well-balanced. This approach focuses on avoiding images dominated by color cast, capturing images with neutral white balance, and filtering out extremely warm, cold, or tinted photos.

The training does not use machine learning or training. However, the threshold (min_val, max_val) were tuned empirically. First a sample set of Unsplash images was collected. Their RGB mean scores were measured, Human visual examination was used to identify which images appeared balanced. Thresholds were \adjusted until the classification aligned with human judgment.

The tool's role in the pipeline is to identify and return only images with well-balanced color. The tool's contributions are filtering noisy datasets where some images are heavily tinted. Improving downstream model input quality, ensuring consistency across curated dataset.

2.6 Album Filtering Tool (Methodology Matthew Peck)

The album filter tool was evaluated using the Unsplash Dataset which was loaded in via the huggingface dataset. The dataset was chosen due to providing good lighting conditions depicting many different areas. This will prove to be helpful as the model needs to know how to differentiate many different backgrounds.

The album filtering tool utilized the Open CLIP model which uses a dual-encoder architecture to embed images and text into a common vector space. During the training, it matches image-text pairs which helps it understand the images that match with the text. This is used during the testing phase as users will input a text and the array of images and the model needs to understand what it is looking for and output the correct set of images based on the input.

The model utilized a threshold to determine what were related images. There is a static component and a dynamic component to the threshold. The model computes the similarity scores between each image and text. Afterwards, it normalizes the values to be between 0 and 1. It then first checks if the max score is above the static threshold to determine if any of the images are related to the input text. It then utilizes the dynamic threshold to determine which similarity scores are close to the max score. This indicates that the chosen images are similar to the best scoring image. It also determines that this group of images are related to the input text and returns the images as the filtered album.

2.7 Background Blur Tool

The background blur tool utilizes a DETR Resnet 50 panoptic segmentation model by facebook in order to identify all regions labeled as the main person in the image. These segments are merged into a single mask which will identify as the subject mask.

The tool then constructs a background mask in relation to the subject mask and applies a Gaussian blur to this background mask using a depth of field approach so the blur intensity varies based on how far the portion of the image is from the subject.

2.8 Removal Tool

The tool will first parse the prompt using a rule-based parser. Here the parser extracts targets such as person, car, dog and will map these to COCO style categories and open vocabulary terms. It also will use spatial hints like left, right, and center. These cues help with detection and selection logic. Following that the tool also utilizes the DETR Resnet 50 panoptic segmentation model to produce masks for these semantic targets we established. For objects less covered by COCO the tool uses OWL-ViT for text-conditioned object detection where detected boxes are filtered based on confidence which helps with thin objects and colored prompt matching. Once masks are identified the tool uses GrabCut to refine the mask better and these masks are passed into the Stable Diffusion XL 1.0 inpainting model. Here the masked people are removed and the fine tuned model will work to realistically fill the removed areas to recreate the image with the people successfully removed.

3 Results

This section presents the performance, behavior, and observed limitations of each component of the system. Results are reported for the interface, the LLM agent, and all image-processing tools, including quantitative evaluations, runtime measurements, and qualitative observations gathered during testing.

3.1 Interface (Douglas Glover)

At the beginning of the project we were unsure if the application was going to be a web application or a desktop application. So we built the project to be split into two parts: front-end (javascript) and back-end (python). This setup allowed for both options to be viable. In the end we created a web application that runs the frontend with github pages and the backend on a cloud server provider with high GPU and RAM capabilities that is suitable for running several models at once. This significantly improved the response time of several of the more resource intensive tools that were previously running on lesser GPUs and low VRAM.

3.2 LLM Agent (Rylee Albrecht)

Fine-tuning the LLM required several iterations to reach stable performance. Balancing the model's ability to produce correctly formatted JSON tool calls while still responding naturally when no tool was needed proved challenging. Incorporating a more detailed system message significantly improved training stability; the model required fewer epochs and avoided overfitting on the fine-tuning dataset. A separate test set of 225 shuffled examples was used to evaluate the model's ability to handle varied request types and switch between tool-calling and natural responses. The final model achieved an overall response accuracy of 88%. In cases where the model cannot interpret the user's request, it does a good job of prompting the user to restate or modify their query.

3.3 Focus Tool (Jamal Mapp)

The focus tool demonstrates efficient runtime performance, averaging 0.08 - 0.12 seconds per image on CPU-only execution. When GPU is available for DeepLabV3 based semantic masking, the average time per image decreases by approximately 30-35%. This is due to accelerated mask generation.

The system produces four quantitative focus metrics derived from Laplacian variance and mask-guided feature isolation.

- Global Focus Score: Laplacian variance over the entire image.
- Saliency-Guided Focus Score:Focus score computed only on visually salient regions.
- Canny-Edge Focus Score: Laplacian score measured only along detected edges.
- DeepLab Masked Score: Focus score on semantically segmented foreground regions.

The tool selects the highest-confidence metric from among these masked evaluations to determine if an image is sufficiently sharp to be returned.

Several observations were noted during experimental testing. The mask-guided focus measurements produced more accurate results than global-only approaches, especially for images with sharp subjects and blurred backgrounds.

The DeepLab-guided score is particularly effective for object-centered or portrait imagery, where foreground sharpness is most relevant.

Images containing broad smooth regions often exhibit artificially low laplacian variance despite being visually acceptable.

High- frequency noise can inflate focus scores, although the masked approach reduces this effect, relative to global scoring alone.

Despite the strong performance, several limitations still remain. Linear horizontal or panning motion blur is not always detected accurately due to the isotropic nature of the laplacian operator. The effectiveness of masked scoring depends on the quality of the DeepLab segmentation. When segmentation fails or produces overly coarse masks, score reliably is reduced. Scores are normalized but not mapped to perceptual sharpness thresholds. As a result, numerical scores cannot directly be interpreted as “human-perceived sharpness levels”. The last limitation of the tools is images with inherently low texture tend to receive lower scores regardless of the actual optical focus quality.

3.4 Exposure Tool (Jamal Mapp)

The exposure tool demonstrated fast inference performance, with an average response time of 0.09-0.12 seconds per image during batch evaluation on the Unsplash test dataset. This efficiency is largely attributed to the tool’s reliance on lightweight statistical operations rather than deep convolutional processing, allowing it to scale well across large image sets.

Quantitatively, the tool produced highly consistent overexposure and underexposure classifications, with decision thresholds rooted in established luminance ranges. When evaluated on Unsplash dataset, the tool correctly flagged 92% of severely underexposed and 89% of severely overexposed images. Moderate exposure deviations were detected with lower sensitivity, reflecting an expected trade-off between computational efficiency and perceptual nuance.

Notable limitations include decreased performance on images with mixed lighting, where global luminance alone is insufficient to infer exposure quality. In cases where only small image regions are improperly exposed, the global mean intensity metric can underrepresent the severity of local exposure issues. Additionally, because the tool does not incorporate perceptually calibrated color-space modeling, it may misclassify certain scenes with intentionally stylized lighting or high-key compositions.

Overall, the exposure tool provides fast and reliable exposure assessment for the majority of natural images, with predictable failure modes primarily tied to local lighting complexity.

3.5 Color Tool (Jamal Mapp)

The color tool exhibited high computational efficiency, with an average response time of 0.06-0.10 seconds per image on the Unsplash dataset. Because the tool relies exclusively on lightweight statistical operations, it introduces minimal processing overhead compared to more complex color-space conversions or perceptual modeling techniques.

Quantitatively, the tool produced consistent separation between well-balanced and color-skewed images, based on the predefined threshold for inter-channel deviation. During evaluation on a 1000 -image subset, the tool correctly identified 87% of images with strong color dominance and filtered out 78% of naturally balanced images. This reflects an expected trade-off between sensitivity and false positives. Images with extreme monochromatic scenes were consistently flagged as "Imbalanced", aligning with the tool's intended behavior.

Notable limitations emerged during testing. The tool's reliance on global RGB statistics make it less effective for images with localized color balance. Artistic photography with intentionally exaggerated color palettes was often classified as imbalanced, despite being aesthetically appropriate . the RGB based approach also does not account for human perceptual nonlinearities, such as varying sensitivity to changes in different color ranges, which can cause the tool to overpenalize minor channel difference in darker scenes or under-penalize difference in highly saturated images.

Overall, while the original color tool offers fast deterministic, and interpretable color-balance detection, its performance is constrained by the limitations of simple RGB statistics. This limit motivated the team's efforts to explore perceptually informed alternatives.

3.6 Album Filtering Tool (Results Matthew Peck)

Two models were tested of the SigLIP and the Open CLIP model and based on the results, the open CLIP model was selected. THe SigLIP model took 4.2 seconds for each iteration while the Open CLIP took 3.84 seconds per iteration. These models were evaluated using Recall which measures sensitivity and finds all the relevant instances of a class. It was evaluated on 100 samples and Open CLIP greatly outperformed the SigLIP model in terms of related images. The SigLIP model was very ineffective at 25% recall while the Open CLIP model had a 60% recall. The SigLIP model was very ineffective at distinguishing related images and random images and would group all the images around a similar score. However, the Open CLIP model would give a score on average of 5-10% higher for related images compared to unrelated ones which helped determine the threshold for the dynamic filter.

Limitations of the model included that the model dynamic threshold is based on the best input image in relation to the text. If the best image is only slightly related, the model will not be able to completely distinguish it from the unrelated images and would return unrelated images alongside related ones. Furthermore, the model was more consistent when not including the static filter at all. However, this would become an issue if all of the user's images were unrelated. Instead of outputting none of the images, it would output all of the images. It was determined this would be better as from a user perspective, it would be better if all the images were kept and they realized they needed to give a better prompt compared to all images disappearing which may appear as if the model deleted the images.

3.7 Background Blur Tool (Jaden Barnwell)

The background blur tool provides a non-generative, segmentation driven effect. It targets the main subject and segments it out while applying a blur to the surrounding background utilizing a depth of field approach where it will change blur intensity according to the depth in relation to the main subject.

The tool will be used by the user if the user enters into the LLM a photo and a prompt like "blur background." The LLM will then return the photo edited with the applied blur.

The result with this depth of field approach is a soft mask surrounding the main subject with a smooth transition between blur and in focus. The depth of field approach was much more realistic and effective than the regular gaussian blur or the other approach of bokeh blurring.

The background blur tool was pretty consistent and effective with blurring everything but the subject on most occasions successfully.

3.8 Removal Tool (Jaden Barnwell)

The removal tool implements a text driven generative inpainting pipeline built on top of modern vision and diffusion models. The goal is to remove people or specified objects like vehicles from an input photo from the user while realistically reconstructing the underlying background.

The user will input a photo and a prompt like “remove the people in this image” or “remove the person in the center of the image” to the LLM and it will trigger the removal pipeline. The tool works best on images larger than 1000 by 1000 pixels and higher quality images.

Once run, the LLM will return an image that has been edited in response to what the user prompted it to remove. After the first run the removal tool usually takes around three minutes to edit and return an image.

The model performs the best when it tries to remove people or a person from a high quality image. It also is pretty good with identifying and removing vehicles. However, it still needs further training with buildings, smaller objects in an image that are harder to find like a rag, and colored items. It also sometimes will struggle when trying to identify a person in a very busy, crowded image where the person or background might not be clear and obvious.