FrameCheck MVP

Core Project Roadmap: FrameCheck MVP

The core project will focus on the essential features that provide a complete, single-user experience: uploading a photo, getting a visual analysis (rule of thirds, leading lines), and receiving AI-generated feedback.

Step 1: Frontend - Image Upload and Display

Goal: Create a clean, responsive UI for users to upload an image and see a preview.

- **Technology:** React with Vite, Tailwind CSS.
- **Libraries:** react-dropzone for a better drag-and-drop experience.
- **Key Tasks:**
    1. **File Input:** Use a component to handle file selection. react-dropzone simplifies this by providing a hook and handling the file validation.
    2. **State Management:** Use useState hooks to store the uploaded file and the URL for the image preview.
    3. **Preview:** Display the image using URL.createObjectURL(file) to show a local preview before it's sent to the backend.
    4. **Send to Backend:** Use axios or the native fetch API to send the image data to your FastAPI endpoint. You will need to wrap the image file in a FormData object for this.

Step 2: Backend - File Reception and Processing

Goal: Create a FastAPI endpoint that receives the image, saves it temporarily, and performs the initial image processing.

- **Technology:** Python with FastAPI.
- **Libraries:** python-multipart to handle file uploads, Pillow (PIL) for image handling, and OpenCV (cv2) for computer vision.
- **Key Tasks:**
    1. **Endpoint:** Create a /analyze-image/ endpoint that accepts a POST request with an UploadFile object.
    2. **Save Image:** Read the uploaded file's contents and save it to a temporary location on your server.

3. **Image Loading:** Use Pillow to open the image. Convert the image to a format OpenCV can work with (a NumPy array).
4. **Initial Data:** Get the image dimensions (width, height) and send a confirmation back to the frontend to signal that processing has started.

Step 3: Computer Vision - Rule of Thirds and Leading Lines

Goal: Implement the core computer vision algorithms to analyze the image composition.

- **Technology:** Python with OpenCV.
- **Libraries:** OpenCV (cv2) and NumPy.
- **Key Tasks:**
    1. **Rule of Thirds:**
        - Calculate the positions of the grid lines based on the image dimensions.
        - Draw the grid lines onto a copy of the image using cv2.line().
        - **Analysis:** This is the tricky part. You'll need to use a method to detect the "subject" or areas of interest. A simple but effective method is to use a saliency map to find the most visually prominent regions. You can then check if these regions fall on or near the grid lines or intersection points.
    2. **Leading Lines:**
        - **Edge Detection:** Apply the Canny edge detection algorithm (cv2.Canny()) to the grayscale image. This will give you a binary image with only the edges.
        - **Line Detection:** Use the Probabilistic Hough Transform (cv2.HoughLinesP()) to detect line segments from the edges. This is more efficient than the standard Hough transform.
        - **Analysis:** Filter the detected lines based on length, angle, and position to identify lines that are likely "leading lines." For example, long, diagonal lines originating from a corner are good candidates.
        - **Drawing:** Draw the detected leading lines onto the image using cv2.line().

Step 4: AI Feedback and Frontend Overlays

Goal: Generate a human-readable critique from the analysis and display it along with the visual overlays on the frontend.

- **Technology:** Python, React, and an AI API.

- **Libraries:** OpenAI Python library or Google Generative AI library for the backend.
- **Key Tasks:**
    1. **Backend Integration:**
        - Take the results from your CV analysis (e.g., "subject is centered," "found 3 diagonal leading lines").
        - Construct a detailed prompt for your chosen AI model. The more specific the prompt, the better the output. Example: "The subject is centered, not on a rule of thirds intersection. There are 2 prominent diagonal leading lines. The image has a low amount of negative space. Explain this to a beginner photographer in 3 concise, positive tips."
        - Call the AI API and receive the generated tips.
    2. **Frontend Display:**
        - The backend should return a JSON response containing:
            - The base64-encoded image with the overlays.
            - The text-based AI feedback.
        - On the frontend, display the base64 image directly in an <img> tag.
        - Render the AI tips in a clear, easy-to-read format below the image.

Integrating the "Add-On" Features

These features can be added on top of your core project. Each one represents a new, self-contained set of tasks.

1. Crop Suggestion

- **Concept:** The app analyzes the image and suggests a crop that would improve the composition based on the same principles (rule of thirds, leading lines).
- **Implementation:**
    - **Backend:** After detecting the subject's location and leading lines (from the core project), you can programmatically calculate a new bounding box.
    - For the Rule of Thirds, calculate a crop that places the subject or a key object on one of the intersection points.
    - For Leading Lines, find a crop that emphasizes the detected lines, for example, by making them appear to lead into the frame from a corner.
    - Once the new bounding box is calculated, use Pillow or OpenCV to perform the crop and return the new cropped image.
    - **Frontend:** Add a "Suggest Crop" button. When clicked, it sends a request to the new backend endpoint, which returns the cropped image and a new analysis.

2. Emotion-Based LLM Explanation

- **Concept:** Use a multimodal LLM to analyze the image content (not just composition) and provide feedback on the mood or "feel" of the photo.
- **Implementation:**
    - **Backend:** This requires a multimodal model like Google Gemini or GPT-4o that can take an image as input.
    - Create a new FastAPI endpoint. This endpoint will receive the image and send it directly to the multimodal LLM API.
    - **Prompt Engineering:** The prompt is the key here. You need to ask the model to act as a photo critic and analyze the emotional tone. Example: "Analyze the following image for its emotional tone and mood. Describe how elements like color, light, and negative space contribute to the overall feeling. Explain this to a beginner photographer."
    - **API Call:** You would send the image (as a base64 encoded string or a URL) along with the text prompt in the API request.
    - **Frontend:** Add a button like "Analyze Mood" that triggers this new API call and displays the results in a separate section of the UI.

3. "Style Match" to a Favorite Photographer

- **Concept:** This is a more advanced feature that compares a user's photo to the style of a famous photographer or a user-uploaded example. This is best done using image embeddings.
- **Implementation:**
    - **Backend:**
        - **Embedding Model:** You'll need a model that can convert images into numerical representations (embeddings). CLIP is a great choice as it links text and images. You can use a pre-trained model from Hugging Face.
        - **Database:** You will need a database of reference images (e.g., from famous photographers). For a uni project, a small, manually curated set is fine.
        - **Process:**
            1. The user uploads a photo.
            2. Your backend sends the user's photo to the CLIP model to get its embedding.
            3. You then calculate the similarity (e.g., using cosine similarity) between the user's photo embedding and the embeddings of all the reference photos in your database.
            4. Find the highest-scoring match and return the name of the photographer and the reference image.
    - **Frontend:** Create a new page or section. Allow the user to upload an image and press a "Match Style" button. Display the name of the matched photographer and their reference image.