PROJECT TITLE

ACNE CLASIFICATION

(BY SEMESTER – **VII,** M.SC. (CA & IT) 2024)

SUBMITTED BY:

STUDENT NAME: -	**ROLL NO: -**
JAIVIK SANDIPBHAI RATHOD	4243
MO SUFYAN UMARFARUK SHAIKH	4257
JAY MANOJBHAI UPADHYAY	4270

GROUP ID: - 25

SUBMITTED TO: -

K. S. SCHOOL OF BUSINESS MANAGEMENTAND INFORMATION TECHNOLOGY

Table of Contents

Introduction

The Acne Classification Web App is an advanced tool designed to analyze skin images for acne detection and classification. Built using Python Flask for the backend and React.js for the frontend, the app enables users to upload skin images and receive real-time analysis.

Objectives

- **Acne Detection**: Identifies whether the uploaded image contains signs of acne.
- **Severity Assessment**: If acne is detected, the app evaluates its severity on a scale.
- **Personalized Suggestions**: Provides suggested medications for severity levels below 4.

Problem Statement

- Acne is a common skin condition affecting individuals worldwide, often leading to concerns about appearance and skin health. Identifying acne and determining its severity can be challenging without professional guidance, causing delays in treatment or inappropriate remedies. There is a need for a user-friendly tool that can assist individuals in identifying acne, evaluating its severity, and providing actionable advice, including medication suggestions or a recommendation to consult a dermatologist for severe cases.

- This web app addresses the gap by offering an automated acne detection and classification solution accessible to all users.

Basic Concepts

Key AI Terminologies

- **Computer Vision**: A field of artificial intelligence that enables machines to interpret and process visual data, such as images or videos.
- **Image Processing**: Techniques used to enhance or extract meaningful information from images, such as applying filters to highlight specific features.

- **Color Filtering**: Identifying specific color ranges in an image to isolate features of interest, such as acne in this project.
- **Pixel Intensity**: The numerical representation of colour and brightness at each pixel in an image, crucial for detecting patterns like redness in acne.

Introduction to AI Techniques Used in the Project

- **OpenCV (cv2):** Used for image processing tasks, including filtering and manipulating the uploaded images.

- **NumPy**: Provides efficient numerical operations, such as managing pixel arrays and performing calculations on image data.

Requirement Analysis

Tools and Technologies Needed

➢ **Frontend:**

- **React.js:** For building a responsive and interactive user interface.
- **CSS/Tailwind CSS**: For styling the app's layout and components.

➢ **Backend:**

- **Python Flask:** To handle image processing requests and manage API communication.
- **CSS/Tailwind CSS:** For styling the app's layout and components.

Basic System Requirements

➢ Hardware:

- **Processor**: Minimum dual-core processor (i5 or equivalent recommended).

- **RAM**: 4GB minimum (8GB recommended for smooth operations).

- **Storage**: At least 10GB free space for development tools and libraries.

➢ **Software**:

- Python (Version 3.7 or higher).
- Package Managers: pip (for Python) and npm/yarn (for React.js).
- Node.js (For React.js development).

➢ **Dependencies**:

- **Python Libraries**: OpenCV, NumPy, Flask.
- **React.js Libraries:** React Router, Axios (for API calls).

Dataset

- We haven't used any dataset for acne classification. We used random image from the internet for developing & testing purpose.

Proposed Solution

Simple Explanation of the Approach

- The proposed solution is a web-based application that allows users to upload an image of their skin. The system processes the image using image filtering techniques to enhance features and detect acne. By identifying specific color ranges commonly associated with acne, the app determines:

- **Presence of Acne**: Whether acne is detected in the uploaded image.

- **Severity Level**: If acne is present, it classifies the severity level based on the affected area or intensity of detected colors.

- **Recommendations**: Depending on the severity level, the app suggests medicines or advises consulting a dermatologist for severe cases.

- This streamlined approach uses computational techniques to analyse images without requiring complex machine learning models, ensuring fast and accurate results.

Algorithm or Model Chosen

- In this acne classification web app, two primary models and algorithms are used Face Detection and Color Filtering.

❖ Face Detection Model (Using OpenCV):

- The app uses a Haar Cascade Classifier for face detection, provided by the OpenCV library. This is a machine learning-based object detection method that can detect faces in images or video frames.

- The classifier is pre-trained on a large dataset of face images, allowing it to quickly detect the presence of faces in uploaded images. If a face is detected, the system proceeds with acne detection. If no face is detected, it may either alert the user or stop further processing.

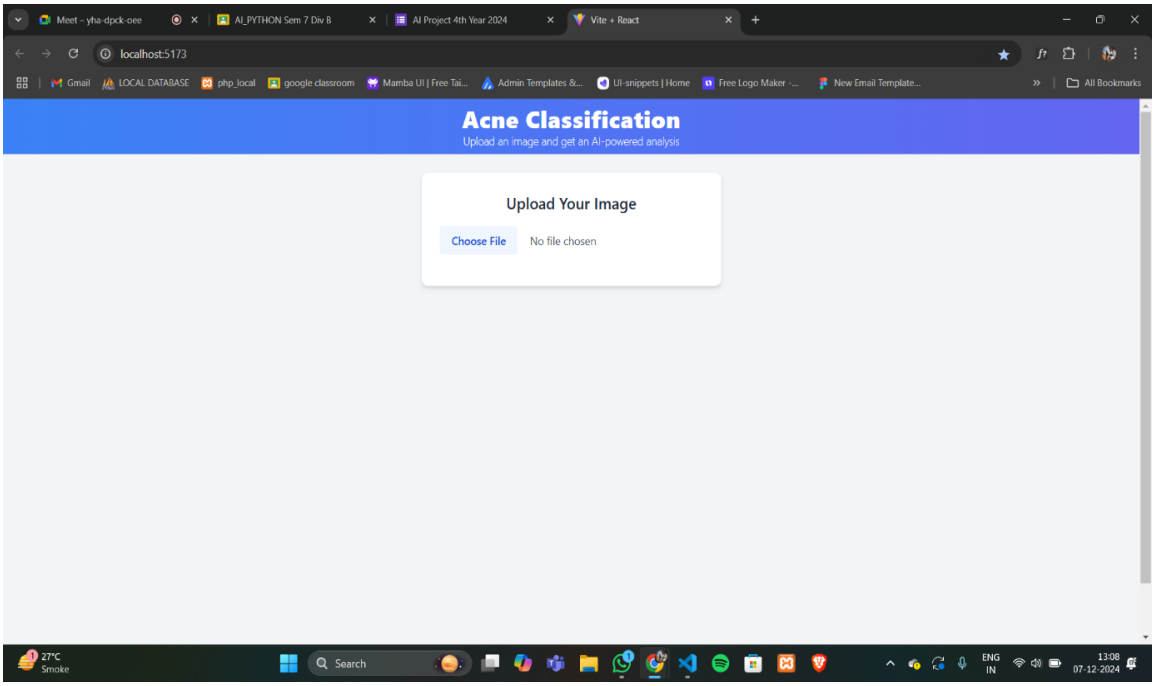
❖ Acne Detection (Using Color Filtering):

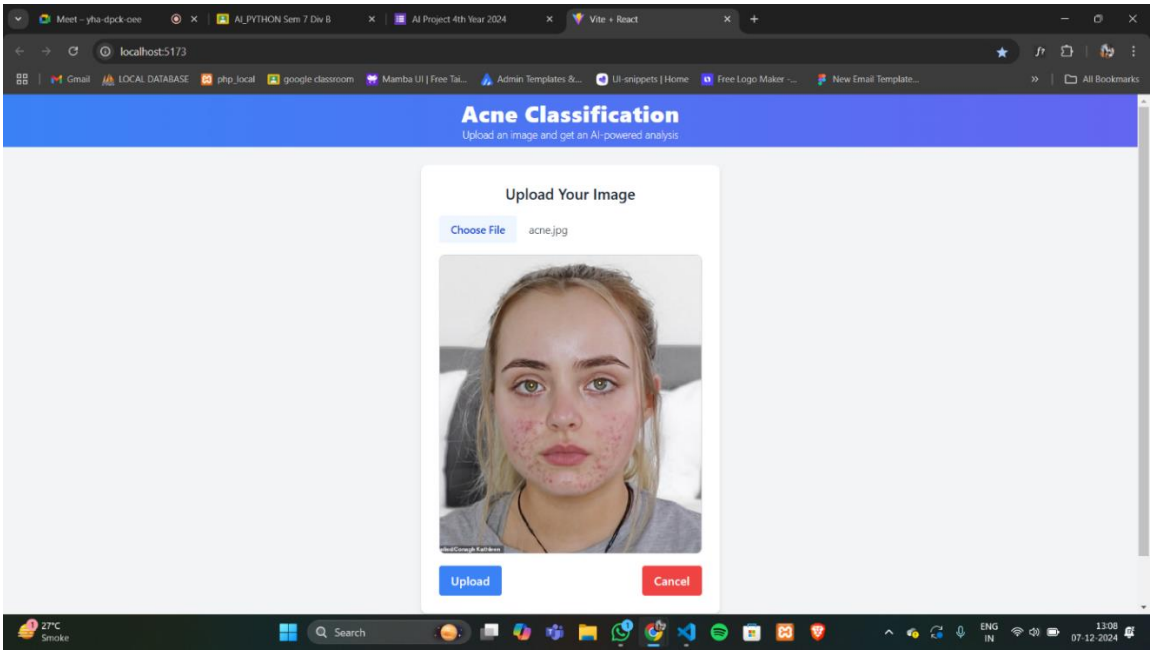
- Once a face is detected, the system applies an image filtering technique to isolate the colors typically associated with acne (e.g., red or pink tones). This is done by applying thresholding to the filtered image, focusing on specific color ranges that correspond to acne.

Implementation

Step by step process

1. **Image Upload**: User uploads the image to the app.





2. **Image Preprocessing:** Convert the image to a suitable color space (such as HSV). And then Apply filters to enhance relevant features (e.g., Gaussian blur to reduce noise).

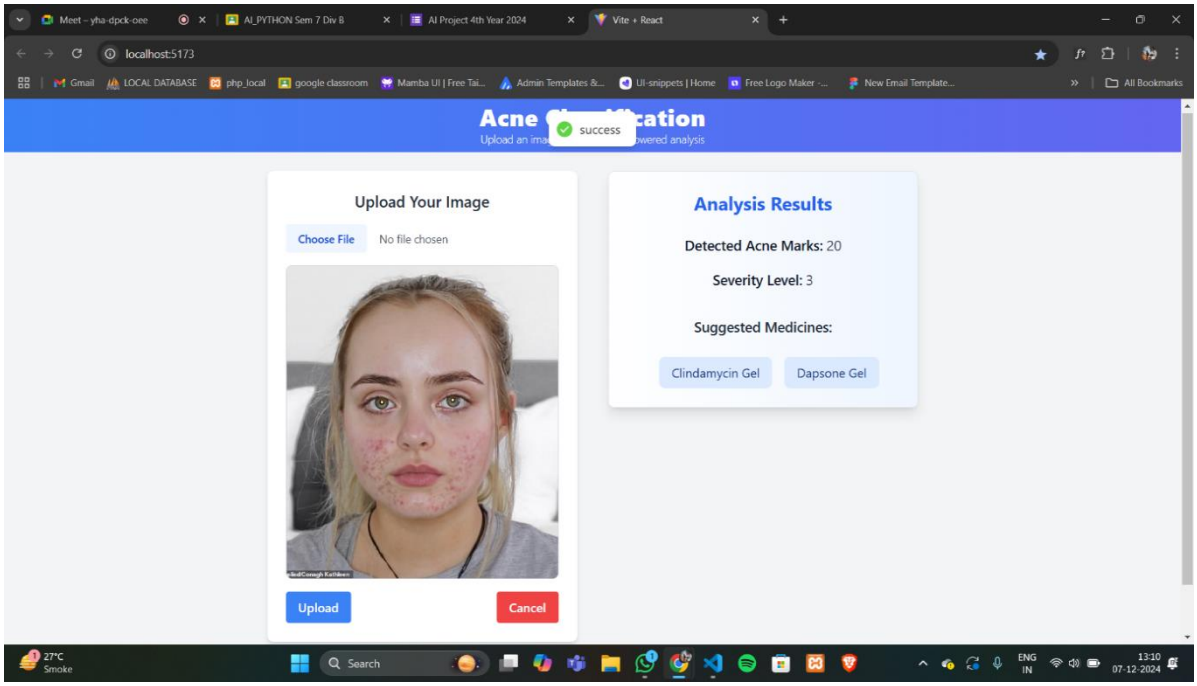
```python
# Function to apply filters to enhance the image
def apply_filters(image, brightness=-100, contrast=250, saturation=80, sharpness=100):
    from PIL import Image, ImageEnhance
    pil_image = Image.fromarray(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))

    # Brightness adjustment
    enhancer = ImageEnhance.Brightness(pil_image)
    pil_image = enhancer.enhance(1 + brightness / 255.0)

    # Contrast adjustment
    enhancer = ImageEnhance.Contrast(pil_image)
    pil_image = enhancer.enhance(1 + contrast / 255.0)

    # Convert back to OpenCV format
    image = cv2.cvtColor(np.array(pil_image), cv2.COLOR_RGB2BGR)

    # Saturation adjustment
    hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    hsv_image[..., 1] = cv2.add(hsv_image[..., 1], saturation)
    image = cv2.cvtColor(hsv_image, cv2.COLOR_HSV2BGR)

    # Sharpness adjustment using a kernel
    kernel = np.array([[-1, -1, -1], [-1, 9 + sharpness / 25, -1], [-1, -1, -1]])
    image = cv2.filter2D(image, -1, kernel)

    return image
```

3. **Color Range Detection**: Define a range of colors typically associated with acne (e.g., red or pink tones). And Use thresholding to isolate these colors in the filtered image. .

```python
def detect_acne_marks(image):
    # Convert the image to HSV color space for color-based thresholding
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

    # Define multiple HSV ranges for red and pinkish tones
    hsv_ranges = [
        (np.array([0, 50, 50]), np.array([20, 255, 255])),    # Red tones
        (np.array([140, 50, 50]), np.array([180, 255, 255]))  # Pink tones
    ]

    # Initialize an empty mask
    final_mask = np.zeros(hsv.shape[:2], dtype=np.uint8)

    # Apply each HSV range and combine masks
    for lower_range, upper_range in hsv_ranges:
        mask = cv2.inRange(hsv, lower_range, upper_range)
        final_mask = cv2.bitwise_or(final_mask, mask)

    # Find contours of potential acne marks in the combined mask
    contours, _ = cv2.findContours(final_mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    # Draw rectangles around detected acne areas on the image
    for contour in contours:
        x, y, w, h = cv2.boundingRect(contour)

        if((x+w) > (x+3) and(x+w) < (x+20) and (y+h) < (y+20) ):
            cv2.rectangle(image, (x, y), (x + w, y + h), (0, 0, 255), 2)  # Red rectangles

    return image
```

4. **Severity Classification**: Calculate the proportion of detected acne-like region and map the proportion to severity levels (e.g., 1–5).

5. **Result Display**: Show the detection result along with recommendations based on severity and If severity > 4, suggest consulting a dermatologist.



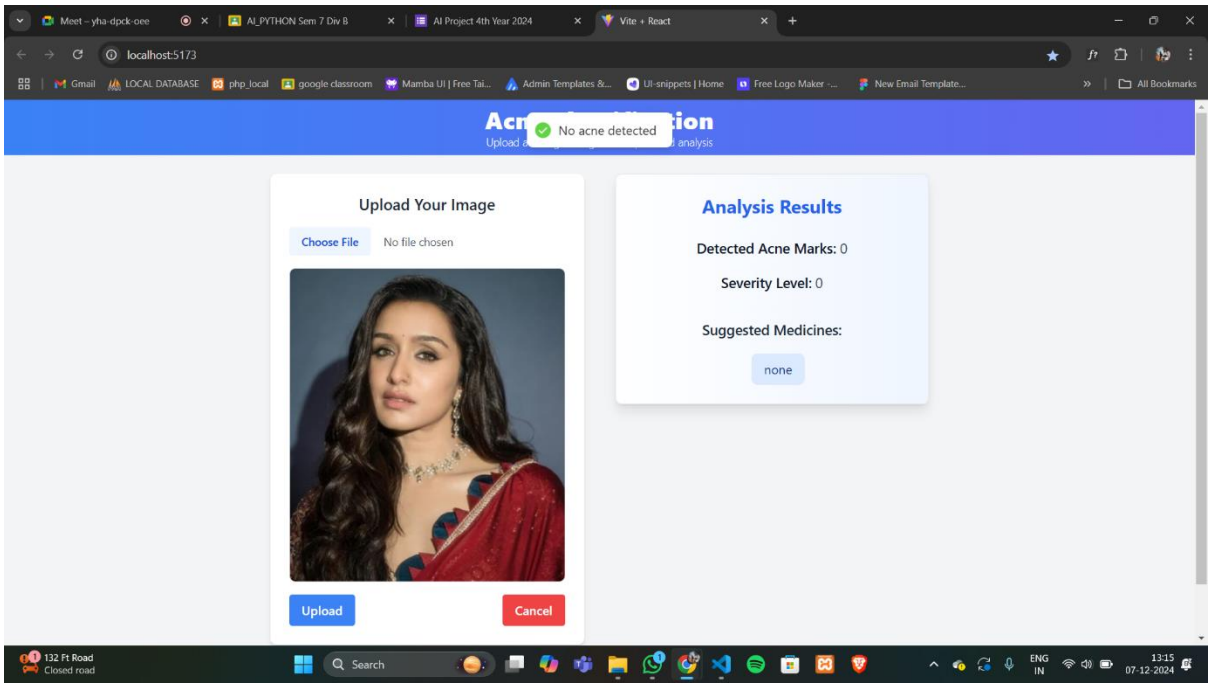
This approach combines efficiency with simplicity, making it accessible to end users while delivering reliable outcomes.

Testing

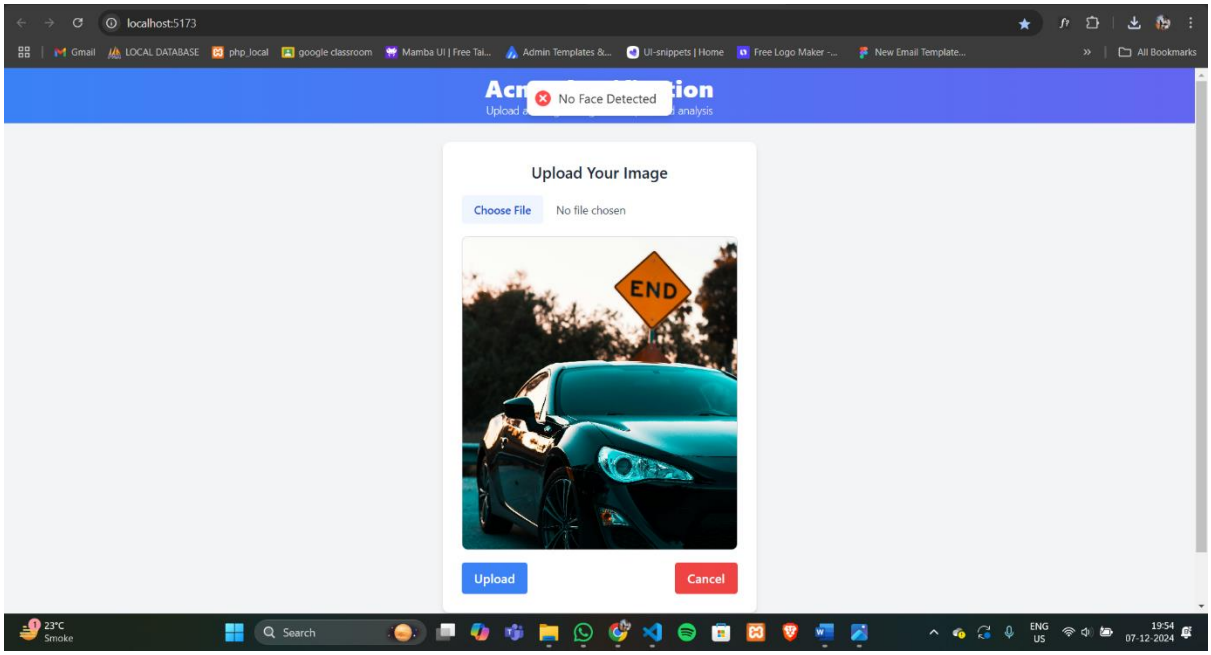
Simple Test Cases

Test Case	Input	Expected Output	Actual Output	Status
Image with Face & Acne	Image containing a face with visible acne	- Face detected - Acne detected - Severity level calculated - Medicine or dermatologist suggested	Matches Expected Output	Pass
Image with Face & No Acne	Image containing a face without acne	- Face detected - No acne detected	Matches Expected Output	Pass
Image Without Face	Image without a face	- No face detected - Processing stopped	Matches Expected Output	Pass
Non-Image File	Invalid input (e.g., text or PDF)	- Error message: “Invalid file format. Please upload an image.”	Matches Expected Output	Pass
Low-Resolution Image	Blurry or pixelated image of a face	- Face detection accuracy may decrease - Possible incorrect results	Matches Expected Output	Pass
High-Resolution Image	Clear image with acne	- Face detected - Accurate acne detection and severity calculation	Matches Expected Output	Pass
Multiple Faces	Image containing multiple faces	- Error message	Matches Expected Output	Pass

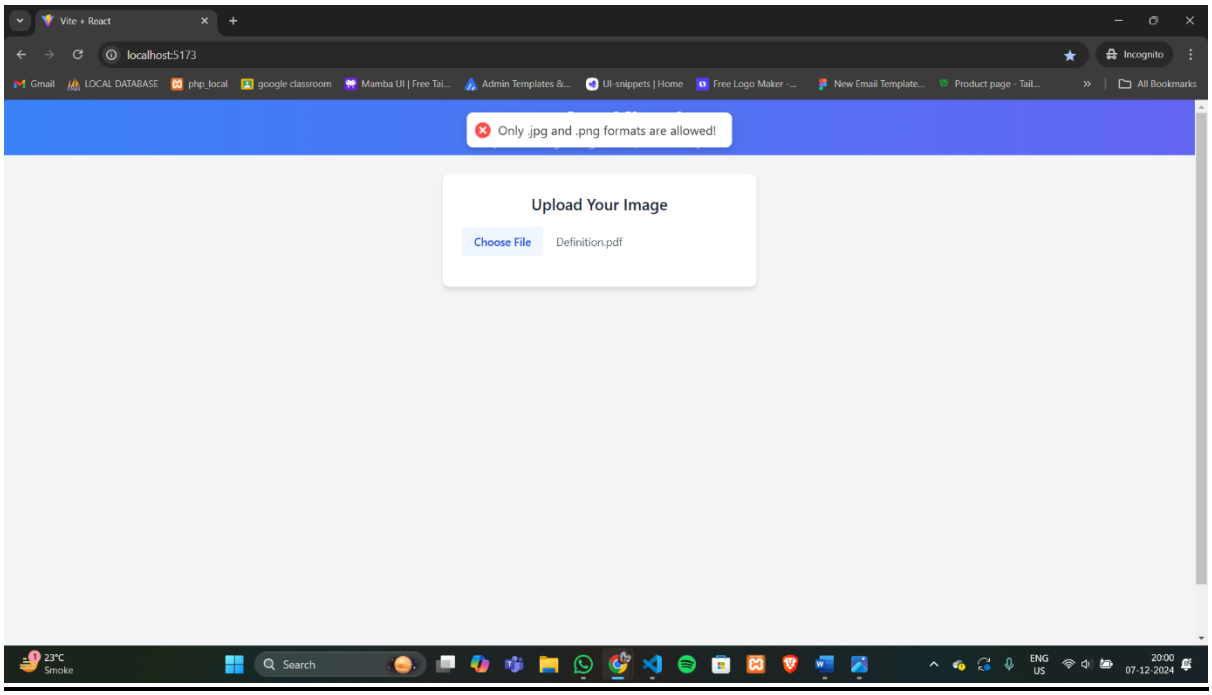
- **Image with Face & No acne: -**



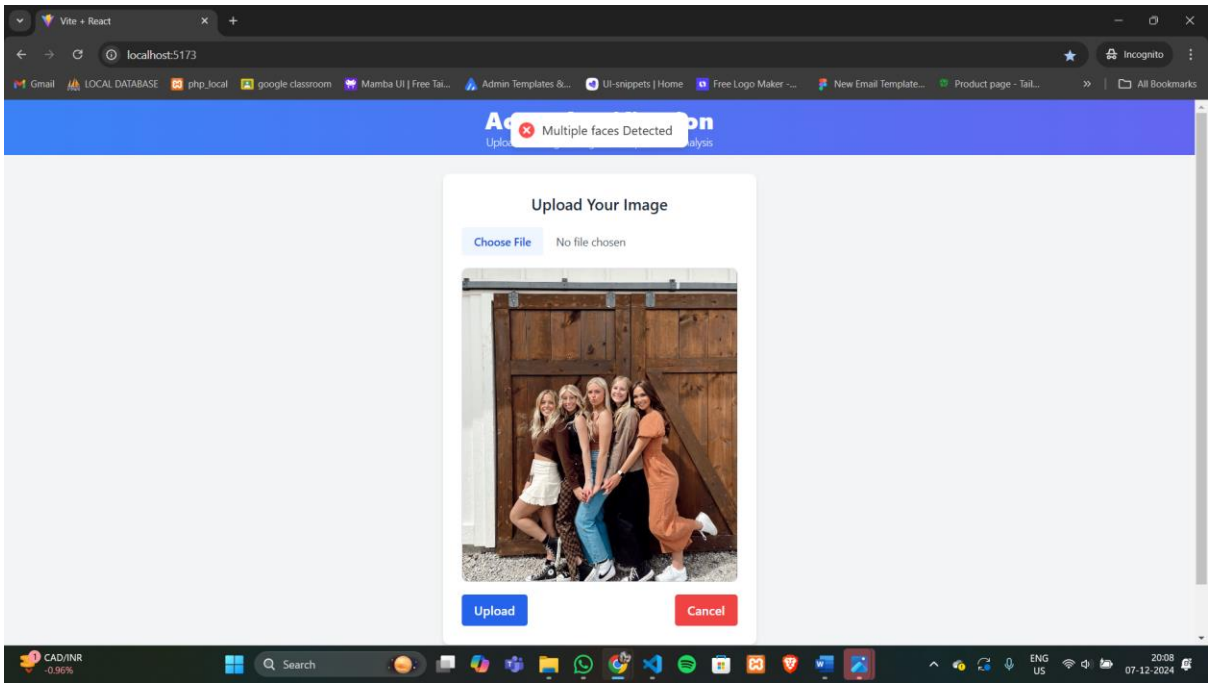
- **Image without Face: -**



- **Non-Image File : -**



- **Multiple Images : -**



Challenges

Basic Problems Encountered

1. **Face Detection Accuracy**: Difficulty detecting faces in poor lighting, unusual angles, or partial images led to false negatives.
2. **Colour Filtering Issues**: Acne detection based solely on colour filtering struggled with noise, uneven lighting, and misidentifying unrelated red areas like rashes or makeup.
3. **Image Quality**: Low-resolution or blurry images affected face detection and acne analysis. Non-face images disrupted the process.
4. **User Feedback**: Users were unclear on why images failed validation or found recommendations too generic.

How They Were Resolved

1. **Improved Face Detection**: Adjusted Haar Cascade parameters, enhanced preprocessing (e.g., contrast adjustments), and added user-friendly error messages for invalid images.
2. **Enhanced Colour Filtering**: Used HSV colour space for better lighting adaptation, applied skin segmentation to avoid false positives, and refined severity classification.
3. **Image Quality Handling**: Validated image resolution and sharpness, prompting users to upload better-quality images when necessary. Added cropping for better focus on facial areas.
4. **User Guidance and Recommendations**: Provided clearer error messages and personalized severity-based recommendations for improved user understanding and satisfaction.

This streamlined approach resolved technical and user experience issues, enhancing accuracy and usability.

Conclusion

What Was Learned?

1. **Image Processing Efficiency**: Using a combination of face detection (Haar Cascade) and colour filtering proved to be a lightweight and effective approach for acne detection without requiring complex machine learning models.

2. **User Behaviour Insights**: User-friendly features like clear error messages and guidance significantly improved the overall experience and usability.

3. **Importance of Preprocessing**: Preprocessing steps such as image enhancement, resolution validation, and skin segmentation played a critical role in ensuring accuracy.

4. **Adaptability**: Challenges like varying lighting, skin tones, and image quality highlighted the need for flexible and robust algorithms.

Key Outcomes of the Project

1. **Functional Acne Classification Tool**: Successfully developed a web app capable of detecting acne, classifying its severity, and providing actionable advice to users.

2. **Scalability Potential**: The system can be expanded to include additional features like personalized skincare tips or integration with dermatologist consultations.

3. **Foundation for Future AI Integration**: While this project used image processing techniques, it sets a foundation for potential integration of AI models for even more precise detection and analysis in future iterations.

References

- OpenCV Documentation - https://docs.opencv.org
- NumPy Documentation - https://numpy.org/doc/
- Flask Documentation - https://flask.palletsprojects.com
- React.js Official Documentation - https://react.dev/

Appendix

- **GitHub Link : -** https://github.com/jaivikrathod/Acne_classification