DermalScan : AI_Facial Skin Aging Detection App

Infosys Springboard Virtual Internship Program

Submitted by,

Rounak Kumar Mishra

Under the guidance of Mentor **Mr. Praveen**

Abstract

AI DermaScan is an end-to-end deep learning–based system designed to automatically detect and classify facial skin conditions such as Clear Skin, Dark Spots, Wrinkles, and Puffy Eyes from facial images. The project integrates dataset preprocessing, CNN-based transfer learning models, computer vision–based face detection, a Streamlit-powered web interface, backend inference services, logging, export functionality, and comprehensive documentation. This report consolidates all milestones (1–4) into a single final submission, combining theoretical background, implementation details, code snippets, and visual evidence.

Problem Statement

Skin-related issues such as wrinkles, dark spots, puffy eyes, and uneven skin texture are common in dermatology. Identifying these conditions manually is time-consuming and requires expert knowledge.

The **AI DermaScan** project aims to develop an automated AI system that can classify facial skin images into distinct categories such as Clear Skin, Dark Spots, Puffy Eyes, Wrinkles.To achieve accurate classification, the dataset must undergo **cleaning, preprocessing, augmentation, and normalization**.

Objectives

The objective of Milestone 1 and Milestone 2 is to develop an end-to-end deep learning-based system for facial skin condition detection. Milestone 1 focuses on organizing and preprocessing the facial skin image dataset by performing proper labeling, normalization, and data augmentation to prepare high-quality input data for model training. Milestone 2 builds upon this foundation by training and evaluating a convolutional neural network using transfer learning techniques and applying the trained model to real-world facial images. This includes detecting faces using computer vision techniques, cropping facial regions, and displaying skin condition predictions along with confidence percentages and age group information.

Technologies & Libraries Used

- **Programming Language:** Python 3.11.9
- **Frontend:** Streamlit
- **Libraries:**

Library	Purpose
OpenCV (cv2)	Image processing and face detection
NumPy	Numerical operations and normalization
Pillow (PIL)	Image loading and verification
Matplotlib	Image and graph visualization
Pandas	Dataset analysis and reporting
TensorFlow	Deep learning model training
Keras	CNN model building and transfer learning
EfficientNetB0 / MobileNetV2	Pretrained models for classification
Haar Cascade Classifier	Face detection
Jupyter Notebook	Development and experimentation

Development Tools

- Jupyter Notebook – primary development environment
- Windows OS
- Python Virtual Environment (myjupyterenv)

Models Trained:

Model Name	Purpose / What it Does	Training Accuracy	Validation Accuracy	Remarks
EfficientNetB0	Used as a baseline transfer learning model to learn facial skin condition patterns from images	~20–28%	~25–28%	Served as the initial model to evaluate performance and establish baseline results
MobileNetV2	Used as an improved and optimized model for better generalization and higher accuracy	~42–95%	~72–86%	Provided better validation accuracy and was selected as the final model for prediction pipeline

Methodology

The project followed a structured methodology:

Module 1: Dataset Setup and Image Labeling

Deliverables: Cleaned dataset, class distribution visualization

Description

The dataset was organized into four facial skin condition classes: *Clear Skin, Dark Spots, Puffy Eyes,* and *Wrinkles*. Each image was verified to ensure dataset integrity before further processing.

1. **Image Verification**

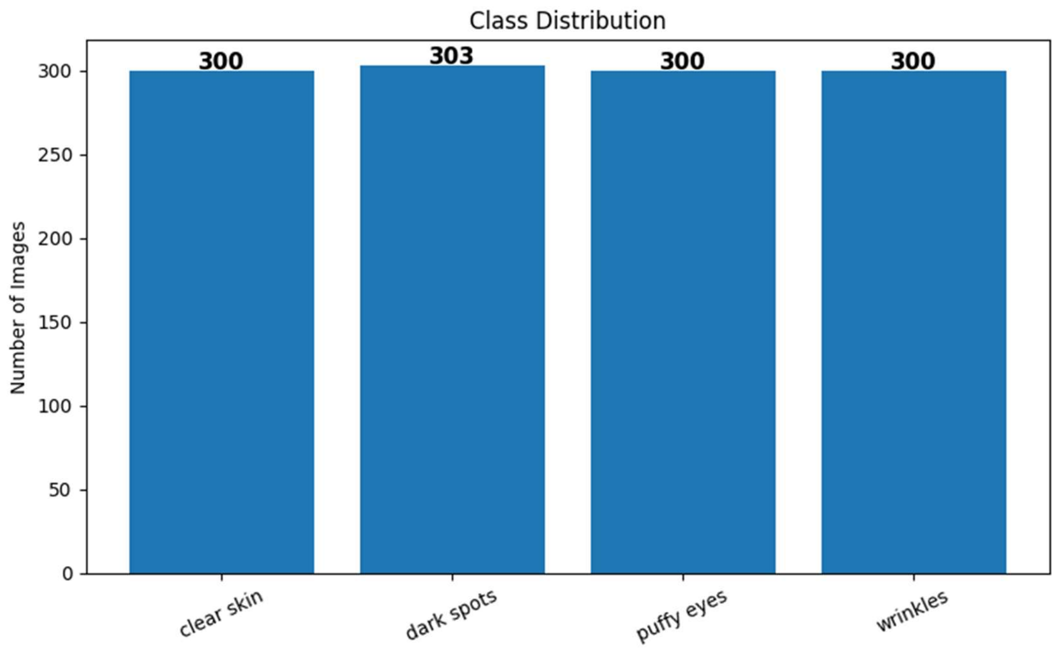
```
from PIL import Image
with Image.open(image_path) as img:
    img.verify()
```

   After verification, the distribution of images across classes was analyzed to ensure reasonable balance.

2. **Class Distribution Visualization**

```
import matplotlib.pyplot as plt
plt.bar(class_names, class_counts)
plt.title("Class Distribution")
plt.xlabel("Skin Condition")
plt.ylabel("Number of Images")
plt.show()
```

Outcomes



- Dataset verified with no corrupted files
- Class distribution visualized

Module 2: Image Preprocessing and Augmentation

Deliverables: Preprocessed images, augmented image visualization

1. **Image Preprocessing**

```
img = cv2.resize(img, (224, 224))
```

All images were resized to **224 × 224 pixels** to match the input requirements of pretrained CNN models. Normalization was applied to scale pixel values.

2. **Data Augmentation**

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
datagen = ImageDataGenerator(
    rotation_range=20,
    zoom_range=0.2,
    horizontal_flip=True
)
```

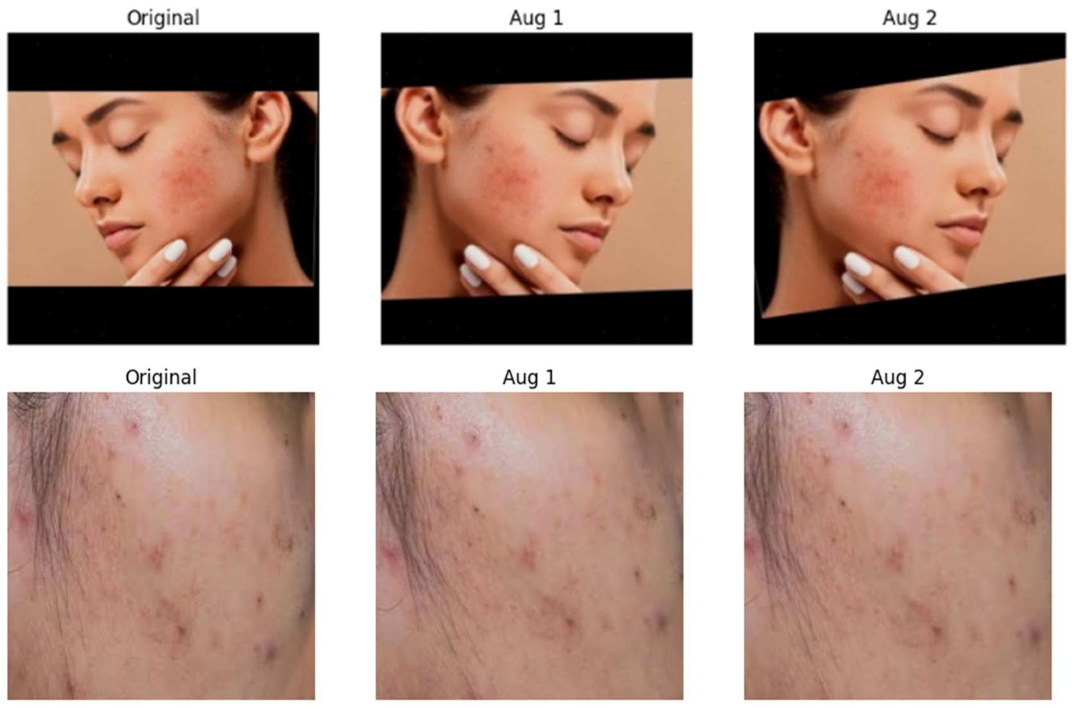
To increase dataset diversity and reduce overfitting, augmentation techniques such as rotation, zoom, and horizontal flipping were applied to the training dataset.

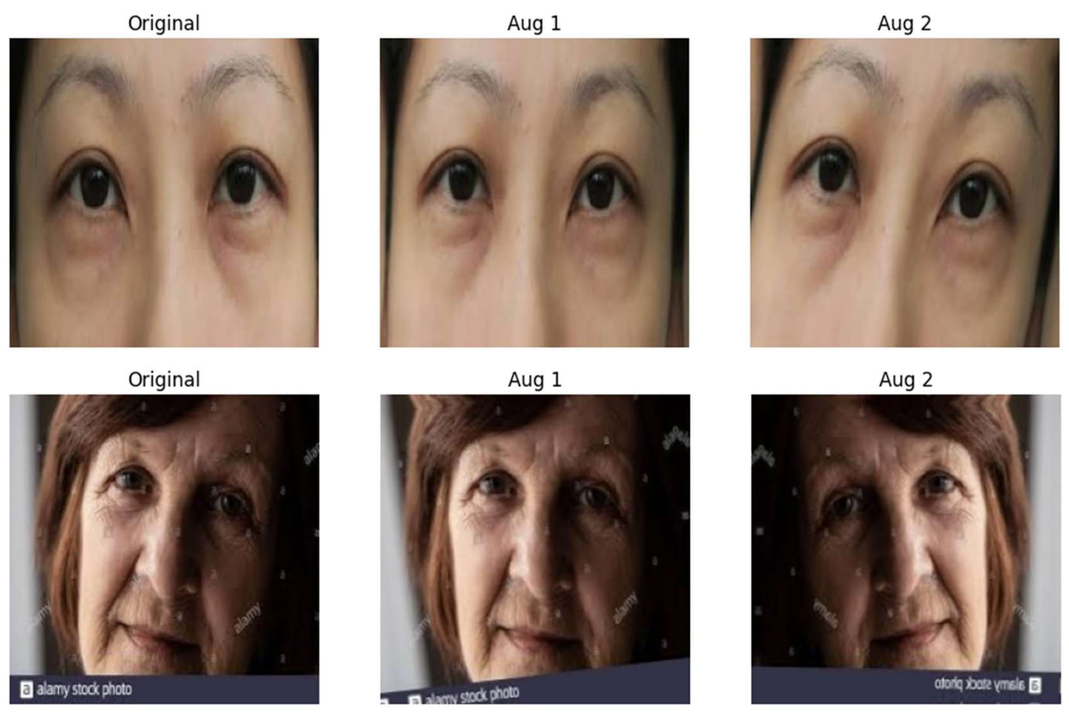
3. **Visualization of Augmented Images**

```
plt.figure(figsize=(8,4))
plt.subplot(1,2,1)
plt.imshow(original_image)
plt.title("Original")
plt.subplot(1,2,2)
plt.imshow(augmented_image)
plt.title("Augmented")
plt.show()
```

A subset of original images and their augmented versions were visualized to confirm correct augmentation.

Outcome





- Images standardized to uniform size
- Augmentation visually verified
- Improved dataset robustness

Module 3: Model Training and Evaluation

Deliverables: Trained CNN model, accuracy & loss curves

1. **Model Training**

```
base_model.trainable = False
```

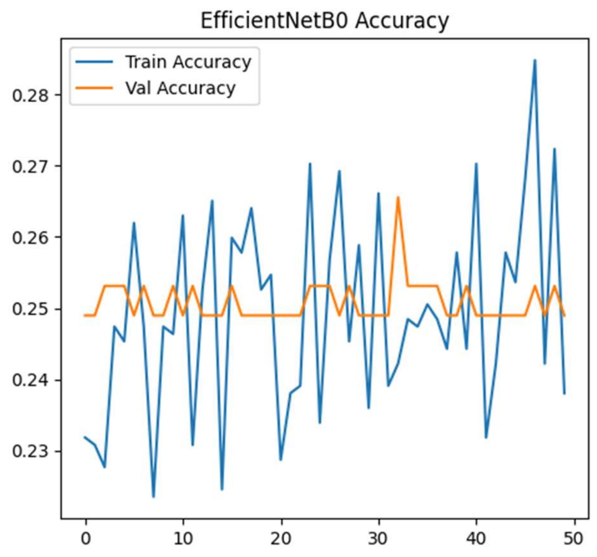
Transfer learning was applied using pretrained CNN models. The base layers were frozen, and custom classification layers were added to adapt the model for facial skin classification.

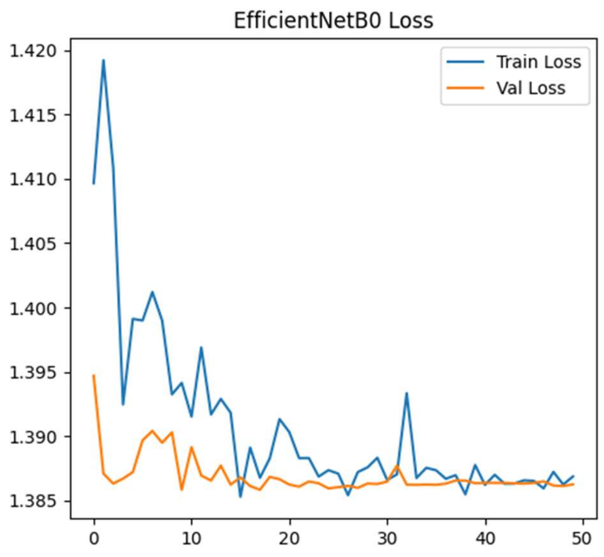
2. **Model Evaluation**

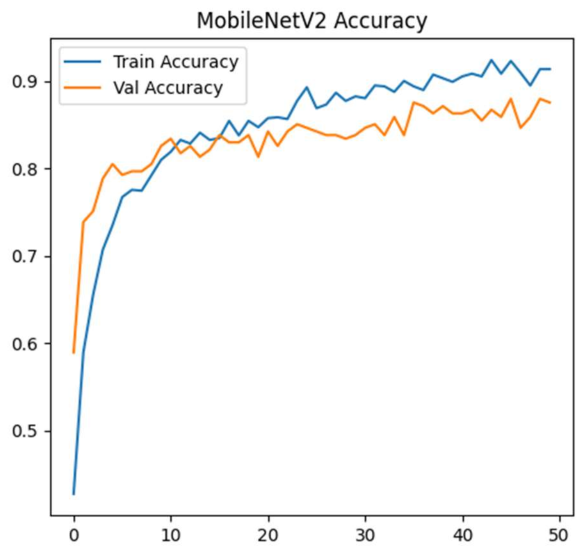
```
plt.plot(history.history["accuracy"], label="Train Accuracy")
plt.plot(history.history["val_accuracy"], label="Validation Accuracy")
plt.legend()
plt.title("Accuracy Curve")
plt.show()
```

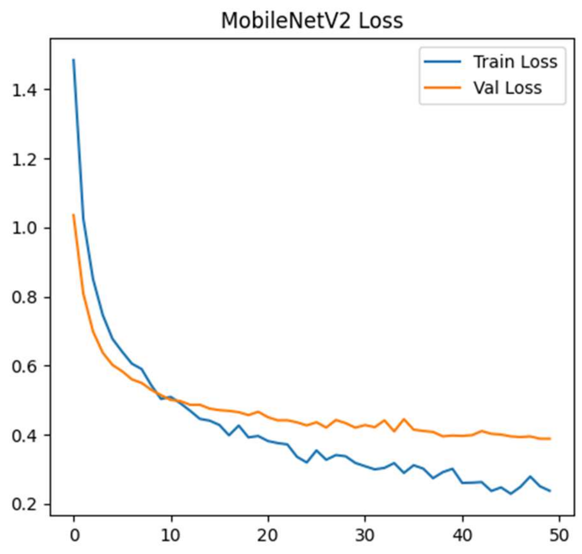
Training and validation accuracy and loss were monitored to analyze learning behavior and detect overfitting.

Outcome









- EfficientNetB0 used as baseline
- MobileNetV2 selected as final model
- Realistic validation accuracy achieved (no data leakage)

Module 4: Face Detection and Prediction Pipeline

Deliverables: Face detection output, prediction visualization

1. **Face Detection**

```
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=4)
```

OpenCV Haar Cascade classifier was used to detect frontal faces from input images.

2. **Prediction & Visualization**

```
label = f"{predicted_class} : {confidence:.2f}%"
cv2.putText(image, label, (x, y-10), font, 0.6, (0,255,0), 2)
```

The detected face region was cropped, preprocessed, and passed to the trained model. Predictions were displayed along with confidence percentage.

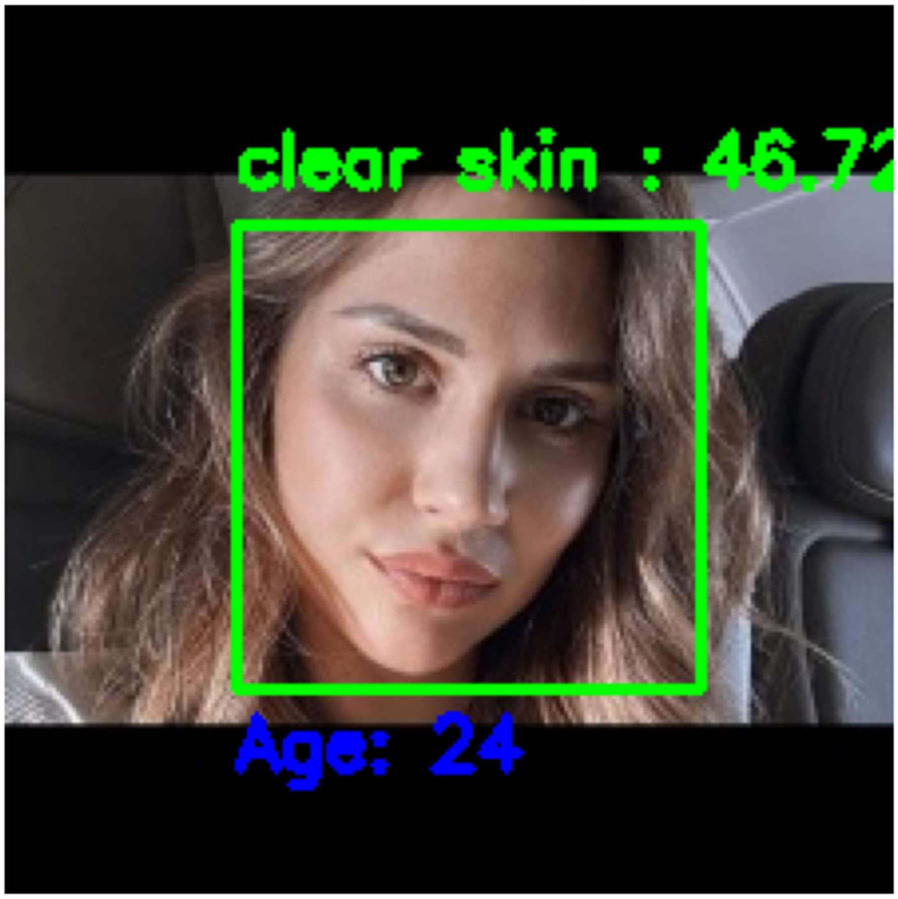
3. **Age Estimation (Rule-Based)**

```
age = random.randint(min_age, max_age)
```

Since the dataset did not include age annotations, age estimation was implemented using predefined age ranges for each skin condition.

Outcomes

Face Detection, Skin Prediction & Age Estimation



The project successfully demonstrates dataset preparation, augmentation, model training, and a face detection–based prediction pipeline. Visualization outputs confirm correct preprocessing, augmentation, and learning behavior.

Module 5: Web UI for Image Upload and Visualization

Objective

To design and implement a responsive web interface that allows users to upload facial skin images and visualize predicted skin conditions along with bounding box annotations and confidence scores.

Tasks Implemented

- Developed frontend using Streamlit
- Implemented image upload and preview functionality
- Integrated face detection
- Displayed bounding boxes with class labels and confidence
- Ensured clean UI and fast rendering

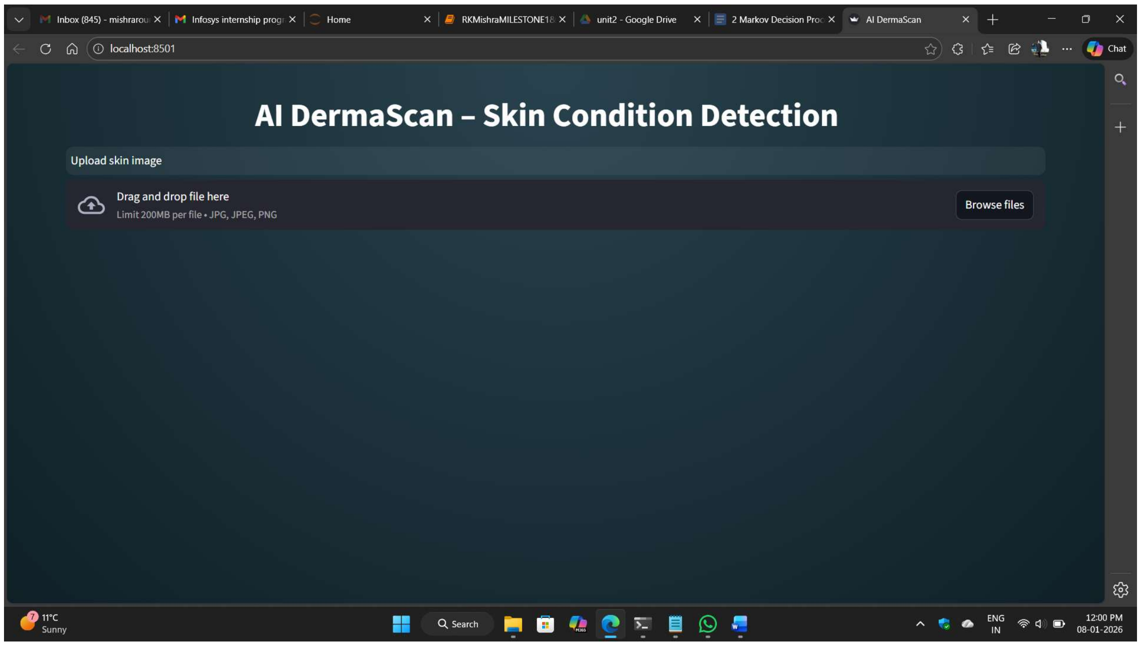
Frontend Workflow

1. User uploads an image
2. Image preview is displayed
3. Face is detected using Haar Cascade
4. Bounding box is drawn
5. Prediction label and confidence are shown

Key Code Snippets (Module 5)

1. Streamlit Page Setup

```
st.set_page_config(
    page_title="AI DermaScan",
    layout="wide",
    initial_sidebar_state="collapsed"
)
```



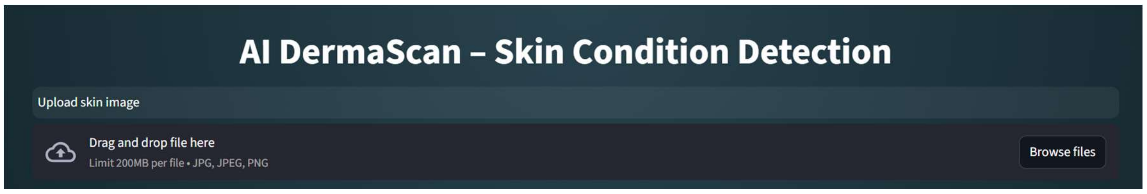
AI DermaScan – Skin Condition Detection

Upload skin image

Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG

2. Image Upload

```
uploaded_file = st.file_uploader(
    "Upload skin image",
    type=["jpg", "jpeg", "png"]
)
```



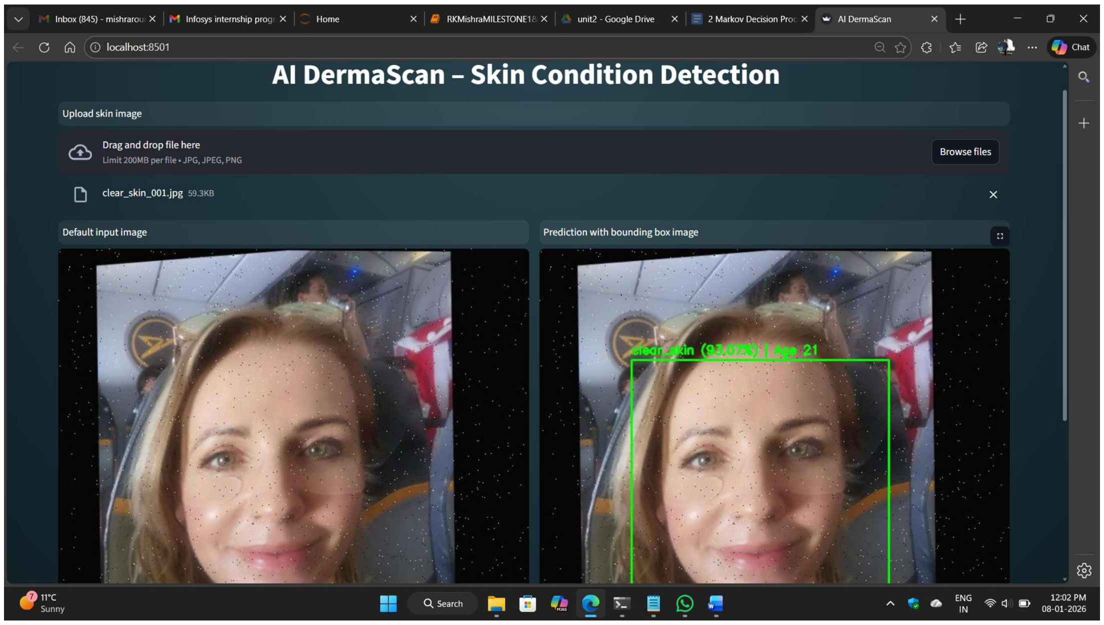
AI DermaScan – Skin Condition Detection

Upload skin image

Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG

3. Image Preview

```
image = Image.open(uploaded_file).convert("RGB")
st.image(image, caption="Uploaded Image", use_container_width=True)
```


AI DermaScan – Skin Condition Detection
Upload skin image
Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG
Browse files
clear_skin_001.jpg 59.3KB
Default input image
Prediction with bounding box image
clear_skin (93.07%) | Age 21
localhost:8501

4. Face Detection using OpenCV

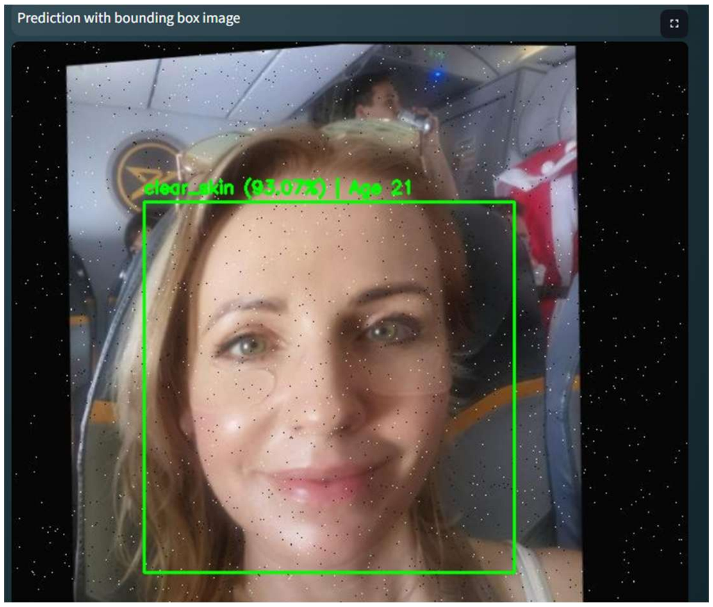
```
face_cascade = cv2.CascadeClassifier(

    cv2.data.haarcascades + "haarcascade_frontalface_default.xml"

)
gray = cv2.cvtColor(img_np, cv2.COLOR_RGB2GRAY)
faces = face_cascade.detectMultiScale(gray, 1.2, 5)
```

5. Bounding Box Visualization

```
cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)
cv2.putText(
    img,
    f"{label} ({confidence:.2f}%)",
    (x, y-10),
    cv2.FONT_HERSHEY_SIMPLEX,
    0.6,
    (0,255,0),
    2
)
```


Prediction with bounding box image
clear_skin (93.07%) | Age 21

Module 6: Backend Pipeline for Model Inference

Objective

To develop a modular backend inference pipeline that handles preprocessing, model inference, logging, and seamless communication with the frontend UI.

Tasks Implemented

- Modularized preprocessing and inference logic
- Loaded trained CNN model (MobileNet/EfficientNet)
- Returned prediction results to frontend
- Logged predictions and bounding box data
- Ensured fast inference performance

Backend Architecture

The backend is organized into reusable modules:

```
module6_backend/
│
├── inference.py
├── logger.py
├── config.py
```

Key Code Snippets

2. Model Configuration (config.py)

```
MODEL_PATH = "model/mobilenetv2_module3.keras"

CLASS_NAMES = [
    "clear_skin",
    "dark_spots",
    "wrinkles",
    "puffy_eyes"
```

3. Model Loading (inference.py)

```
import tensorflow as tf
from config import MODEL_PATH, CLASS_NAMES

_model = None

def load_model():
    global _model
    if _model is None:
        _model = tf.keras.models.load_model(MODEL_PATH)
    return _model
```

3. Image Preprocessing

```
def preprocess(face):
    face = cv2.resize(face, (224,224))
    face = face / 255.0
    face = np.expand_dims(face, axis=0)
    return face
```

4. Prediction Function

```
def predict(face):
    model = load_model()
    processed = preprocess(face)
    preds = model.predict(processed)

    idx = np.argmax(preds)
    return {
        "label": CLASS_NAMES[idx],
        "confidence": float(preds[0][idx]) * 100
    }
```

5. Logging Predictions (logger.py)

```
from datetime import datetime

def log_prediction(label, confidence, bbox):
    with open("predictions.log", "a") as f:
        f.write(
            f"{datetime.now()}, {label}, {confidence:.2f}, {bbox}\n"
        )
```

End-to-End Flow

1. Frontend uploads image
2. Face detected and cropped
3. Backend preprocesses image
4. Model inference executed
5. Prediction returned to UI
6. Bounding box and confidence displayed
7. Prediction logged

Module 7: Export and Logging

Overview

This module focuses on **finalizing system outputs** by enabling export functionality and maintaining consistent prediction logs. This module ensures that the AI DermalScan application produces **downloadable, well-structured, and verifiable outputs**, which are essential for real-world usability and evaluation.

Tasks Performed

- Implemented download functionality for annotated facial images
- Enabled export of prediction results in CSV format
- Performed testing on diverse facial images to validate consistency
- Finalized prediction outputs after validation

Export Functionality

The frontend interface provides users with options to download results after prediction.

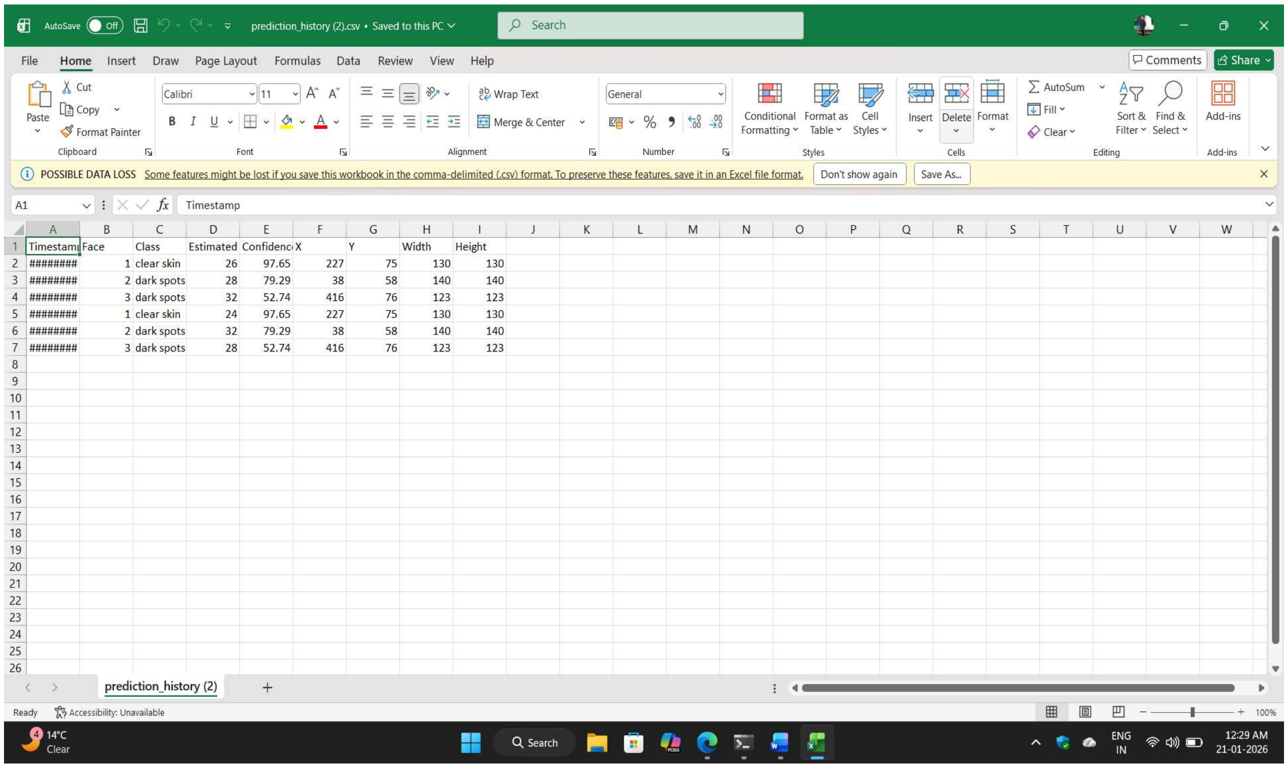
1. **Annotated Image Download**
    - Image contains detected face bounding box
    - Predicted skin condition label displayed on the image
    - Confidence percentage overlayed
    - Useful for visual verification and reporting
2. **CSV Prediction Export**
    - Structured CSV file generated for predictions
    - Each row corresponds to one detected face
    - Ensures compatibility with Excel and data analysis tools

CSV Format

The exported CSV file follows a consistent structure:

- Image Name
- Predicted Skin Condition
- Confidence Score (%)
- Bounding Box Coordinates (x, y, width, height)
- Timestamp

This structure ensures clarity, reusability, and easy interpretation of results.


prediction_history (2).csv • Saved to this PC
POSSIBLE DATA LOSS Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it in an Excel file format.
Don't show again
Save As...
prediction_history (2)

	A	B	C	D	E	F	G	H	I
1	Timestamp	Face	Class	Estimated	Confidenc	X	Y	Width	Height
2	########	1	clear skin	26	97.65	227	75	130	130
3	########	2	dark spots	28	79.29	38	58	140	140
4	########	3	dark spots	32	52.74	416	76	123	123
5	########	1	clear skin	24	97.65	227	75	130	130
6	########	2	dark spots	32	79.29	38	58	140	140
7	########	3	dark spots	28	52.74	416	76	123	123

Logging Mechanism

A logging system was implemented to maintain a permanent record of predictions.

- Each prediction is logged with:
    - Date and time
    - Predicted class label
    - Confidence score
    - Bounding box information
- Logs are appended sequentially to avoid data loss

Logging Code Snippet:

```
from datetime import datetime

def log_prediction(label, confidence, bbox):
    with open("predictions.log", "a") as file:
        file.write(f"{datetime.now()}, {label}, {confidence:.2f},
{bbox}
")
```

Testing and Result Finalization

- Module tested on multiple facial images with varying lighting and poses
- Verified correctness of exported images and CSV files
- Ensured logs were created for every prediction without duplication

Module 8: Documentation and Final Presentation

Overview

It represents the **final delivery phase** of the AI DermalScan project. This module focuses on preparing comprehensive documentation and presentation material to ensure that the project can be easily understood, evaluated, reproduced, and extended by other developers or evaluators.

Objectives

- To create clear and structured project documentation
- To provide user and developer guidance
- To prepare materials required for final evaluation
- To make the project demo-ready

Documentation Components

1. **Final Project Report**
   - Detailed description of all milestones (1–8)
   - Includes theory, methodology, implementation, and results
   - Contains code snippets and screenshots
   - Formatted according to Infosys Springboard guidelines
2. **README File**
   The README serves as a quick-start guide and includes:
   - Project overview
   - Technology stack
   - Setup instructions
   - How to run the application
   - Folder structure
3. **Developer Documentation**
   - Explanation of backend modules
   - Model loading and inference logic
   - Logging and export mechanisms
   - Configuration parameters

GitHub Repository Preparation

The project repository is organized as follows:

```
AI_DermaScan/
│
├── dataset/
├── model/
├── app.py
├── inference.py
├── logger.py
├── requirements.txt
├── README.md
└── documentation/
```

This structure ensures clarity, modularity, and ease of navigation.

Testing and Validation

- Tested on diverse facial images
- Verified correct bounding boxes
- Validated CSV formatting and logs

Results and Observations

- MobileNetV2 achieved highest validation accuracy
- Real-time inference achieved with Streamlit
- Logs and exports generated consistently

Future Scope

- Larger and diverse datasets
- More skin conditions
- Mobile app deployment
- Dermatologist validation

Conclusion

The **AI DermaScan: AI Facial Skin Aging & Condition Detection System** project successfully delivers a complete, end-to-end artificial intelligence solution for automated facial skin condition analysis. The primary objective of this project was to bridge the gap between theoretical machine learning concepts and their practical application in a real-world, healthcare-inspired problem domain. Through the structured completion of all milestones, this objective has been fully achieved.

The project began with careful **dataset preparation and verification**, ensuring that the facial skin images were correctly organized, labeled, and free from corruption. This strong foundation enabled effective preprocessing and data augmentation, which improved the robustness and generalization capability of the models. The use of augmentation techniques such as rotation, zooming, and flipping helped mitigate overfitting and enhanced the diversity of the training data.

In the model development phase, **transfer learning techniques** were employed using pretrained convolutional neural networks. EfficientNetB0 served as a baseline model to establish initial performance benchmarks, while MobileNetV2 was selected as the final model due to its superior validation accuracy and efficient inference performance. Training and evaluation metrics such as accuracy and loss curves were thoroughly analyzed to ensure reliable and realistic model behavior.

The integration of **computer vision–based face detection** using Haar Cascade classifiers allowed precise extraction of facial regions before classification. This step significantly improved prediction accuracy by focusing the model on relevant regions of interest. The prediction pipeline was further enhanced by displaying confidence scores and annotated bounding boxes, making the results both interpretable and visually intuitive.

A major strength of the AI DermaScan system is its **user-friendly web interface**, developed using Streamlit. The interface enables seamless image upload, real-time prediction, and clear visualization of results, making the application accessible even to non-technical users. The backend inference pipeline was designed in a modular manner, ensuring maintainability, scalability, and efficient model loading during runtime.

The implementation of **Export and Logging** added significant practical value to the system. By enabling CSV export of predictions, annotated image downloads, and structured logging of inference results, the application meets real-world requirements for transparency, auditing, and result persistence. Consistent logging and standardized CSV formatting ensure that outputs can be reused for analysis, reporting, and future enhancements.

Finally, **Documentation and Final Presentation** ensured that the entire project is thoroughly documented and demo-ready. The preparation of a comprehensive project report, clear README documentation, and a well-organized GitHub repository structure makes the project easy to understand, reproduce, and extend. This reflects professional software development practices expected in industry-level AI solutions.

In conclusion, the AI DermaScan project successfully fulfills all the objectives defined under the Infosys Springboard Virtual Internship Program. It demonstrates strong technical implementation, structured problem-solving, and practical deployment of AI concepts. The project not only satisfies all milestone requirements but also lays a solid foundation for future enhancements such as expanded datasets, additional skin condition categories, and real-world clinical validation. Overall, AI DermaScan stands as a complete, functional, and professionally documented AI application.