1. INTRODUCTION

1.1 Project Overview

Artificial intelligence and computer technology both heavily rely on machine learning and deep learning. Human effort in identifying, learning, making predictions, and many other areas can be decreased with the application of deeplearning and machine learning.

A computer-based method called computer-aided diagnosis (CAD) is utilised in the field of medical imaging to assist healthcare professionals in their diagnoses1. In various medical specialties, including colonoscopy and mammography,CAD has emerged as a standard tool1.

However, dermatologists perform the majority of noninvasive screening tests simply with the naked eye, despite thefact that skin illness is a frequent disease for which early detection and classification are essential for patient successand recovery.

Due to the ease with which the condition might be missed, this may result in unnecessary diagnostic errors caused by human error. Furthermore, because the symptoms of many common skin diseases share a great deal of similarity, disease classification is challenging. Therefore, it would be advantageous to utilise the advantages

1.2 Purpose

This process describes the process of understanding and analyzing skin disease
Using computer aided diagnosis (CAD) here the non-invasive screening test can be improved from nailed eye to CAD

.here the images of skin can be segmented and then the all are clustered together and we can also we can classify eachcluster into different common skin diseases using another neural network model
Our classification model is more accurate than a baseline model, while also being able to classify multiple diseases within a single image .this improved performance may be sufficient to use CAD in the field of dermatology

2. LITERATURE SURVEY

2.1 Existing problem

Skin disease analysis using computer aided diagnosis (CAD) involves seeing the microscopic image of that disease and checks whether some other disease also accompanied with it . A civilian, who may get affected by skin disease like allergies, rashes, acne etc. Seeing my disease with naked eye will not identify the type of disease correctly and alsowill not be able to get the proper solution People with two or more skin disease may not be able to identify .so using CAD we canidentify the disease properly.

2.2 References

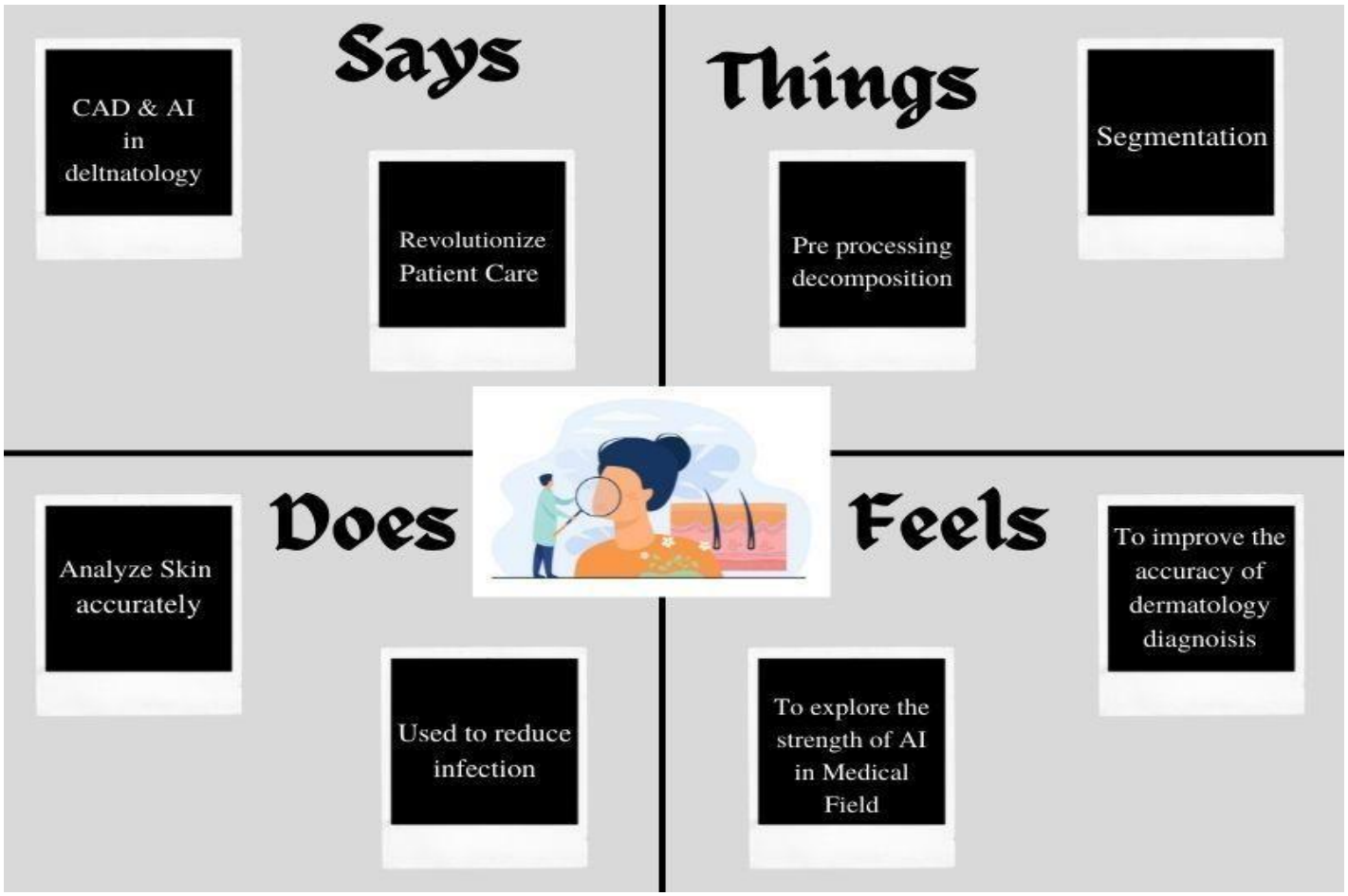
PROJECT TITLE	AUTHOR	OBJECTIVE/OUTCOME

2.3 Problem Statement Definition

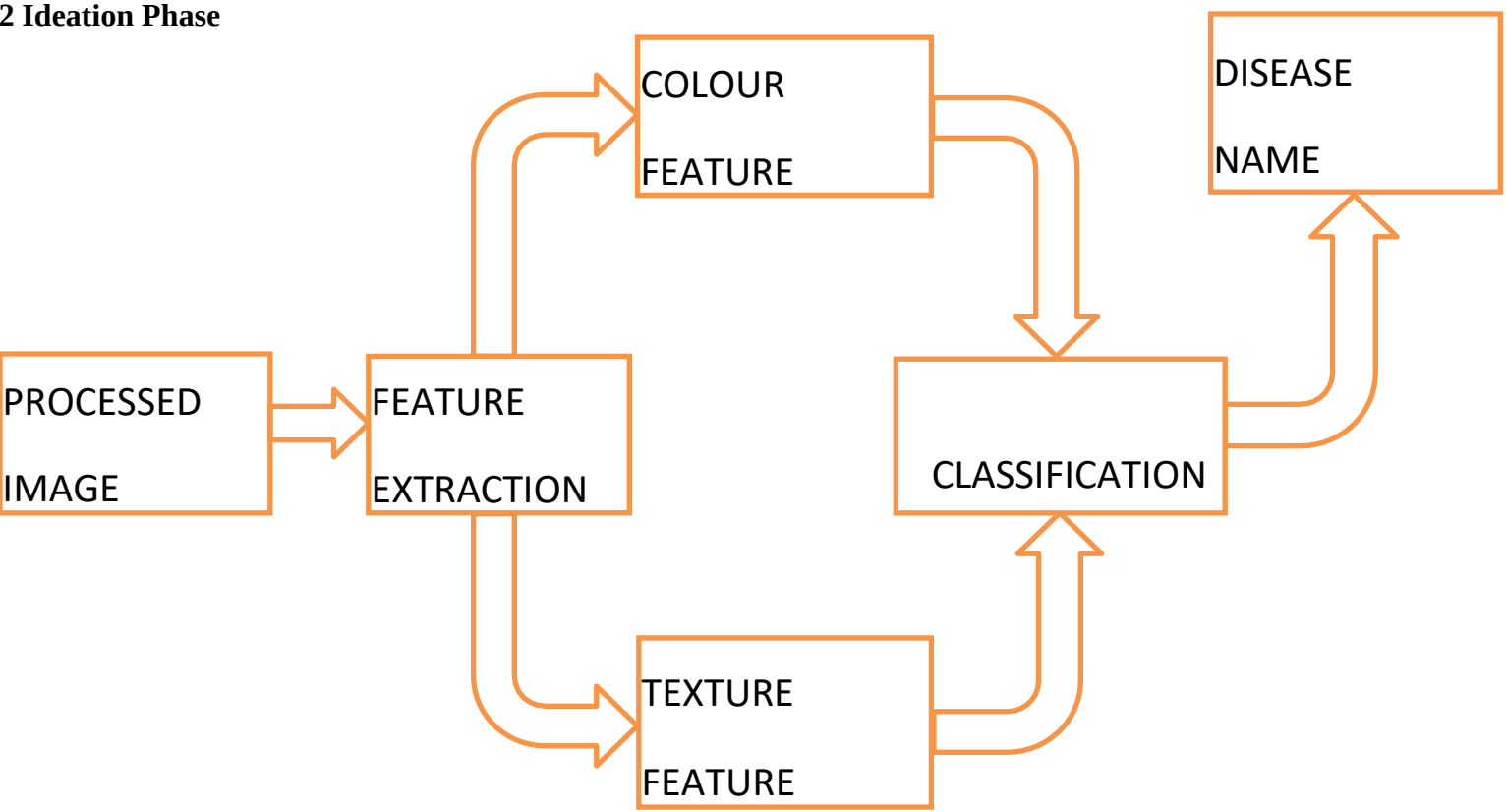
Skin disease analysis using computer aided diagnosis (CAD) involves seeing the microscopic image of that disease and checks whether some other disease also accompanied with it . A civilian, who may get affected by skin disease like allergies, rashes, acne etc. Seeing my disease with naked eye will not identify the type of disease correctly and alsowill not be able to get the proper solution People with two or more skin disease may not be able to identify .so using CAD we canidentify the disease properly. The doctors won't use any of the microscopicinstruments to view the disease.

3. IDEATION & PROPOSED SOLUTION

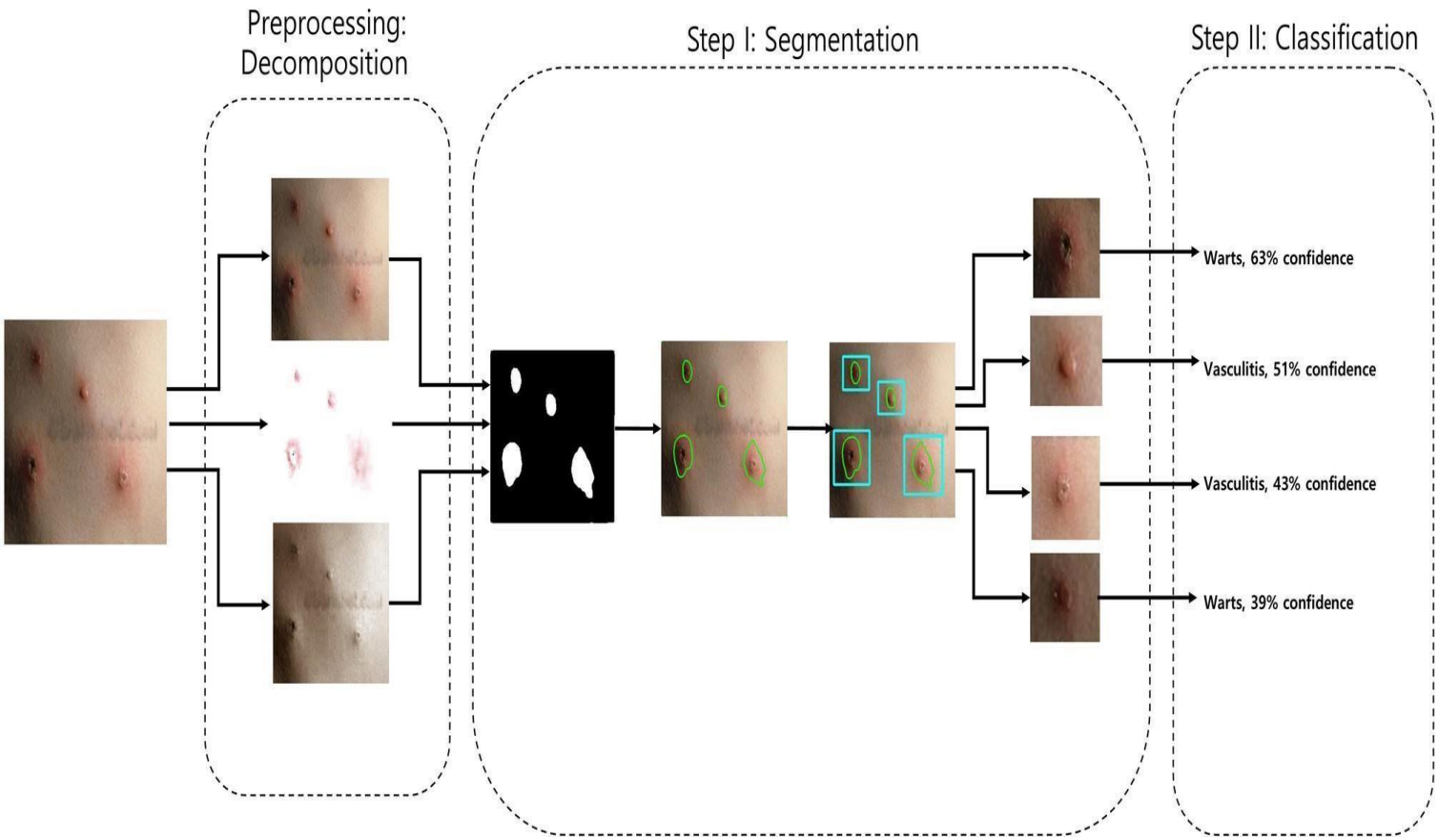
3.1 Empathy Map Canvas



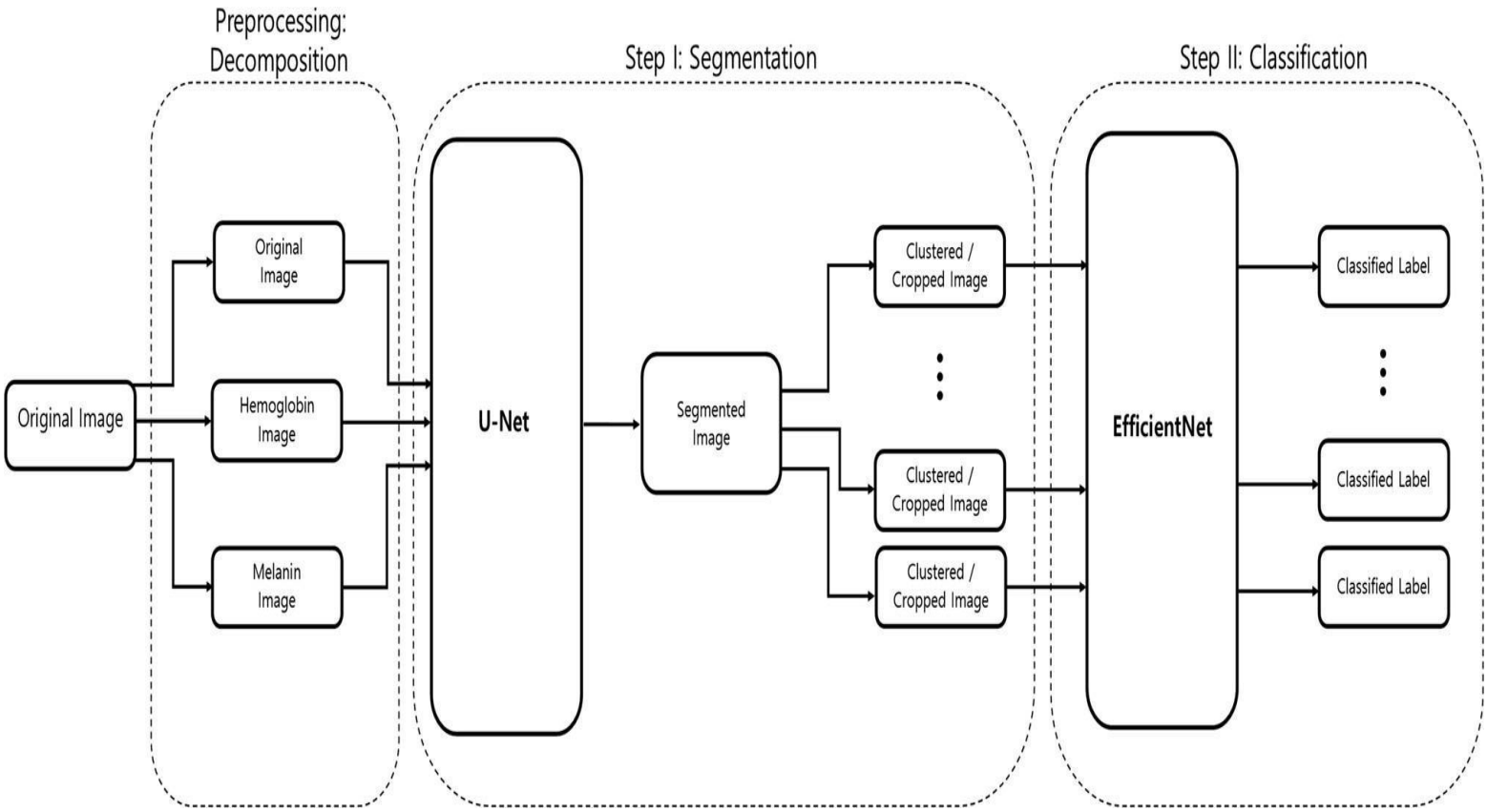
2 Ideation Phase



Phase II


Preprocessing:
Decomposition
Step I: Segmentation
Step II: Classification
Warts, 63% confidence
Vasculitis, 51% confidence
Vasculitis, 43% confidence
Warts, 39% confidence

Phase III



3.3 Proposed Solution

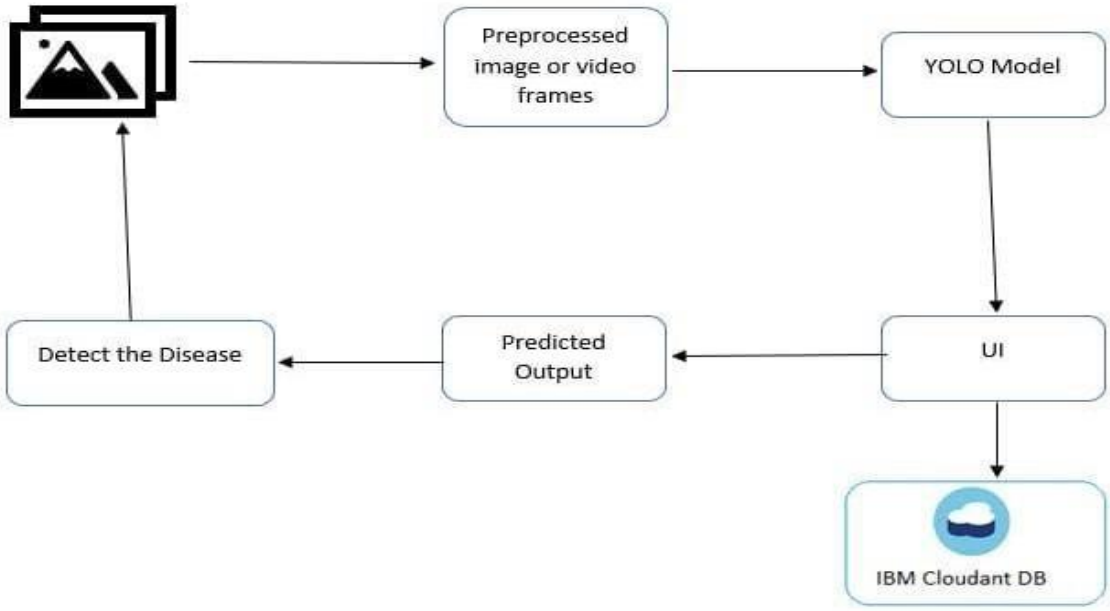
1 . Customer Segment	5.	8. Channels of Behaviour
The patient who deals with Skin disease like allergy, acne,rashes, bumps etc…	The field of dermatology don't have computer aided diagnosis.They only check with eye.	With basic instrument and methodology we cant break theimages into pieces and we cannot cluster the images to getthe proper result.

2 JOBS to-be-done/ Problem Analysis of skin disease with naked eye may some time causeerrors, and some times may not be identified clearly.	6. Customer constraints They believe that this new invention may be costly andunsafe .	9. Problem Root cause We may sometimes have 2 or more skin disease at a time, that time finding that is a difficult taskand giving proper medicine is also a big deal.
3. Triggers To identify the skin diseaseaccurately and quickly 4. Emotions: Before/After Feels discomfort and sad whendisease is not treated.	7. Behaviours Finding the best solution (CAD) fordetecting accurate disease in efficient manner.	10. Your solution Our solution is that we may use CAD in dermatology, this CAD willbreak the image clearly and analysis it separately then they will cluster the image together and will give the solution.

3.4 Problem Solution fit

Although computer-aided diagnosis (CAD) is used to improve the quality of diagnosis in various medical fields such as mammography and colonography, it is not used in dermatology, where non-invasive screening tests are performed only with the naked eye, and avoidable inaccuracies may exist. This study shows that CAD may also be a viable option in dermatology by presenting anovel method to sequentially combine accurate segmentation and classification models. Given animage of the skin, we decompose the image to normalize and extract high-level features. Using aneural network-based segmentation model to create a segmented map of the image, we then cluster sections of abnormal skin and pass this information to a classification model. We classify each cluster into different common skin diseases using another neural network model. Our segmentation model achieves better performance compared to previous studies, and also achieves a near-perfect sensitivity score in unfavorable conditions. Our classification model is more accurate than a baseline model trained without segmentation, while also being able to classify multiple diseases within a single image. This improved performance may be sufficient to use CAD in the field of dermatology.

TECHNICAL ARCHITECTURE:



SOLUTION:

Computer-aided diagnosis (CAD) is a computer-based system that is used in the medical imaging field to aid healthcare workers in their diagnoses CAD has become a mainstream tool in several medical fields such as mammography and colonography1,2 However, in dermatology, although skin disease is a common disease, one in which early detection and classification is crucial for the successful treatment and recovery of patients, dermatologists perform most non-invasive screening tests only with the naked eye. This may result in avoidable diagnostic inaccuracies as a result of human error, as the detection of the disease can be easily overlooked. Furthermore, classification of a disease is difficult due to the strong similarities between common skin disease symptoms. Therefore, it would be beneficial to exploit the strengths of CAD using artificial intelligence techniques, in order to improve the accuracy of dermatology diagnosis. This paper shows that CAD may be a viable option in the field of dermatology using state-of-the-art deep learning models. The segmentation and classification of skin diseases has been gaining attention in the field of artificial intelligence because of its promising results. Two of the more prominent approaches for skin disease segmentation and classification are clustering algorithms and support vector machines (SVMs). Clustering algorithms generally have the advantage of being flexible, easy to implement, with the ability to generalize features that have a similar statistical variance. Travels with various clustering algorithms, such as fuzzy c-means, improved fuzzy c-means, and K-means, achieving approximately 83% true positive rates in segmenting a skin disease. An inherent disadvantage of clustering a skin disease is its lack of robustness against noise. Clustering algorithms rely on the identification of a centroid that can generalize a cluster of data. Noisy data, or the presence of outliers, can significantly degrade the performance of these algorithms. Therefore, with noisy datasets, caused by images with different types of lighting, non-clustering algorithms may be preferred; however, Keke et al. implemented an improved version of the fuzzy clustering algorithm using the RGB, HSV, and LAB colour spaces to create a model that is more robust to noisy data. SVMs have gained attention for their effectiveness in high-dimensional data and their capability to decipher "…subtle patterns in noisy and complex dataset". Although more robust than clustering algorithms, SVMs are more reliant on the pre-processing of data for feature extraction. Without pre-processing that allows a clear definition of hyperplanes, SVMs may also underperform. Owing to the disadvantages of these traditional approaches, convolution neural networks (CNNs) have gained popularity because of their ability to extract high-level features with minimal pre-processing CNNs can expand the advantages of SVMs, such as robustness in noisy datasets without the need for optimal pre-processing, by capturing image context and extracting high-level features through down-sampling. CNNs can interpret the pixels of an image within its own image-level context, as opposed to viewing each pixel in a dataset-level context. However, although down-sampling allows CNNs to view an image in its own context, it degrades the resolution of the image. Although context is gained, the location of a target is lost through down-sampling. This is not a problem for classification, but causes some difficulty for segmentation, as both the context and location of the target are essential for optimal performance. To solve this, up-sampling is needed, which works in a manner opposite to that of down-sampling, in the sense that it increases the resolution of the image. While down-sampling takes a matrix and decreases it to a smaller feature map, up-sampling takes a feature map and increases it to a larger matrix. By learning to accurately create a higher-resolution image, CNNs can determine the location of the targets to segment. Thus, for segmentation, we use a combination of downsampling and up-sampling, whereas for classification, we use only down-sampling. To further leverage the advantages of CNNs, skip-connections were introduced, which provided a solution to the degradation problem that occurs when CNN models become too large and complex. We implement skip-connections in both segmentation and classification models. In the segmentation model, blocks of equal feature numbers are connected between the down and up-sampling sections. In the classification model, these skip-connections exist in the form of inverted residual blocks. This allows our models to grow in complexity without any performance and degradation **Approach**:

In this project, we present a method to sequentially combine two separate models to solve a larger problem. In the past, skin disease models have been applied to either segmentation or classification. In this study, we

sequentially combine both models by using the output of a
segmentation model as input to a classification model. In addition, although past studies of non-CNN segmentation models used innovative pre-processing methods, recent CNN developments have focused

1: **procedure** SEGMENT(*x*)

2: *h, m* = DECOMPOSE(*x*)

3: *mask* = U-NET([*x, h, m*])

4: CLASSIFY(*mask*)

5: **end procedure**

6: **procedure** CLASSIFY(*mask*)

7: *clusters* = FINDCLUSTERS(*mask*)

8: **for** *cluster* **in** *clusters* **do**

9: *cluster* = FIXRATIO(*cluster*)

10: *cluster* = RESIZE(*cluster*)

11: *class* = EFFICIENTNET(*cluster*)

12. *top prediction* = GETHIGHESTCONFIDENCE(*class*)

12: print(*top prediction*)

13: **end for**

more on the architecture of the model than on the preprocessing of data. As such, we apply an innovative pre-processing method to the data of our CNN segmentation model. The methods described above lack the ability to localize and classify multiple diseases within one image; however, we have developed a method to address this problem. Our objective is two-fold. First, we show that CAD can be used in the field of dermatology. Second, we show that state-of-the-art models can be used with current computing power to solve a wider range of complex problems than previously imagined. We begin by explaining the results of our experimentation, followed by a discussion of our findings, a more detailed description of our methodology, and finally, the conclusions that can be drawn from our study.

METHODOLOGY:

Our 2-phase analysis model for localization and classification. We decomposed the original image into its haemoglobin and melanin constituents using pre-processing, to help our model extract valuable information from data that would have been otherwise unavailable. We provide these images as input to our segmentation model, the U-Net, which generated a segmented image. This segmented image was then analysed for clusters, which were subsequently cropped and input to our classification model, the Efficient Net, which then produced a classified label, thus completing our analysis model.

Algorithm 1 Analyse Skin

1: **procedure** SEGMENT(*x*)

2: *h, m* = DECOMPOSE(*x*)

3: *mask* = U-NET([*x, h, m*])
4: CLASSIFY(*mask*)

5: **end procedure**

6: **procedure** CLASSIFY(*mask*)

7: *clusters* = FINDCLUSTERS(*mask*)

8: **for** *cluster* **in** *clusters* **do**

9: *cluster* = FIXRATIO(*cluster*)

10: *cluster* = RESIZE(*cluster*)

11: *class* = EFFICIENTNET(*cluster*)
12. *top prediction* = GETHIGHESTCONFIDENCE(*class*)

12: print(*top prediction*)

13: **end for**

14: **end procedure**

ALGORITHM:

Pre-processing: decomposition

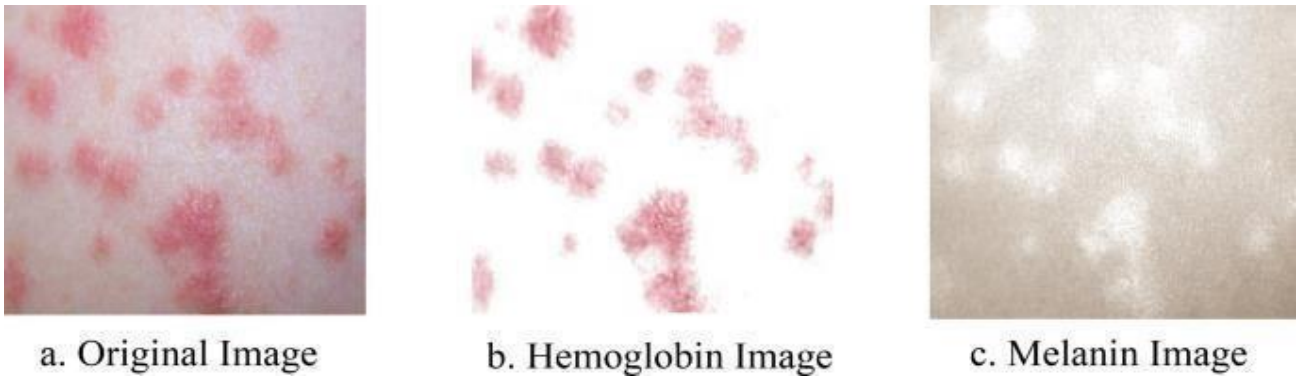
The main constituents of the skin that are visible to humans are melanin and haemoglobin. These constituents provide valuable information for the segmentation of abnormal skin. To ensure that our model can learn to use these features, we used independent component analysis (ICA) to extract the melanin and haemoglobin constituents. Assuming that these components are linearly separable, the separated linear vectors can be represented by the following formula7:

$$L_{x,y} = d_m q_{mx,y} + d_h q_{hx,y} + \Delta \quad L_{x,y} = d_m q_{x,ym} + d_h q_{x,yh} + \Delta$$

where $d_m$dm and $d_h$dh represent the density vectors of melanin and hemoglobin, respectively, $q_{mx,y}$qx,ym and $q_{hx,y}$qx,yh represent the quantity of these components, and ΔΔ represents values that are caused by other colors. As shown in7, by applying ICA, we can decompose skin as

[qmx,y,qhx,y]=D−−1L(x,y)−E[qx,ym,qx,yh]=D−−1L(x,y)−E

E=minx,y(D−−1L(x,y))E=minx,y(D−−1L(x,y)) Ix,y=exp(−L′x,y)Ix,y=exp(−L′x,y) where D−D− represents the estimated values of $d_m$dm and $d_h$dh, and $I_{x,y}$Ix,y represents the decomposed result. Figure shows an example of one of these decompositions.



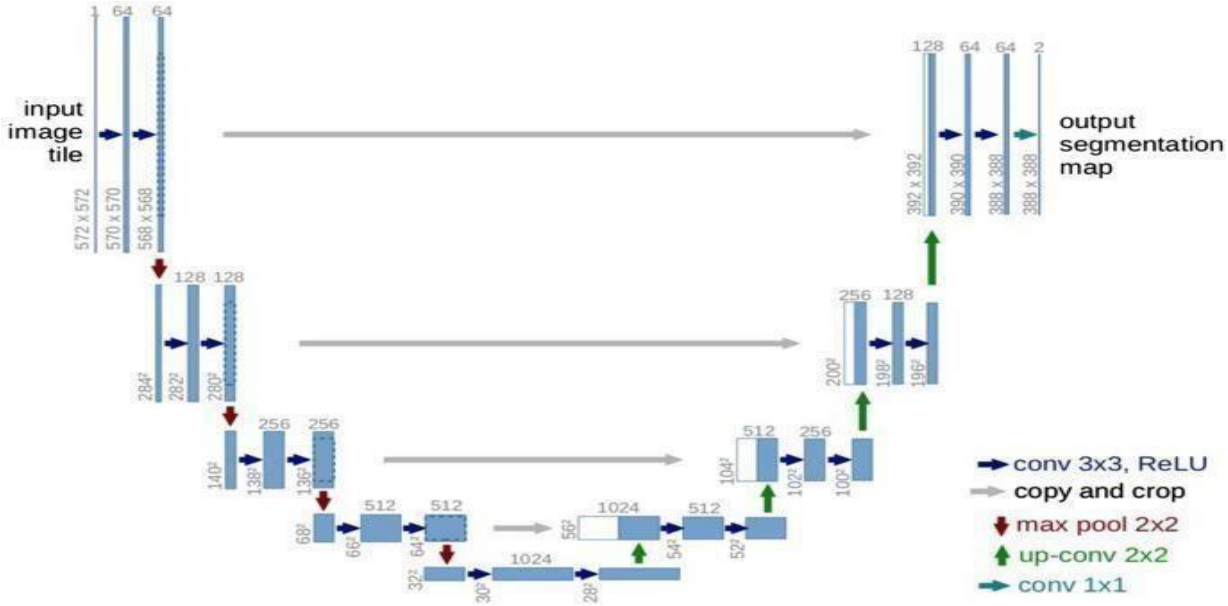
a. Original Image b. Hemoglobin Image c. Melanin Image

Decomposed result of skin. The original image is decomposed into its haemoglobin and melanin constituents through ICA.

Segmentation :

The U-Net, , is an architecture created by CNNs, that has attracted attention for accurate biomedical image segmentation through the combination of down-sampling, up-sampling, and skip connections. Its name is attributed to the shape of its architecture, the first half of the 'U' representing down-sampling. Here, the context and key features of the input images are gained at the cost of a decrease in resolution. The second half of the 'U' represents upsampling. Here, the resolution is increased to gain knowledge of the location of the target segment. To combat degradation due to the complexity of the model, skip connections are added to each up-sampling block.

Figure



U-Net architecture. A fully CNN network, comprised of down-sampling, up-sampling, and skip connections.

Although in the original paper, the resolutions of input and output were different, that is, 572 × 572 and 388 × 388 pixels, respectively, we chose to keep our input and output resolution consistent at 304 × 304 pixels. This was done because the images in our dataset were not large enough to warrant the tiling strategy required for extremely large images. Thus, zeropadding allowed us to keep the input and output resolutions consistent, thereby allowing the retention of information present on the border of our images.

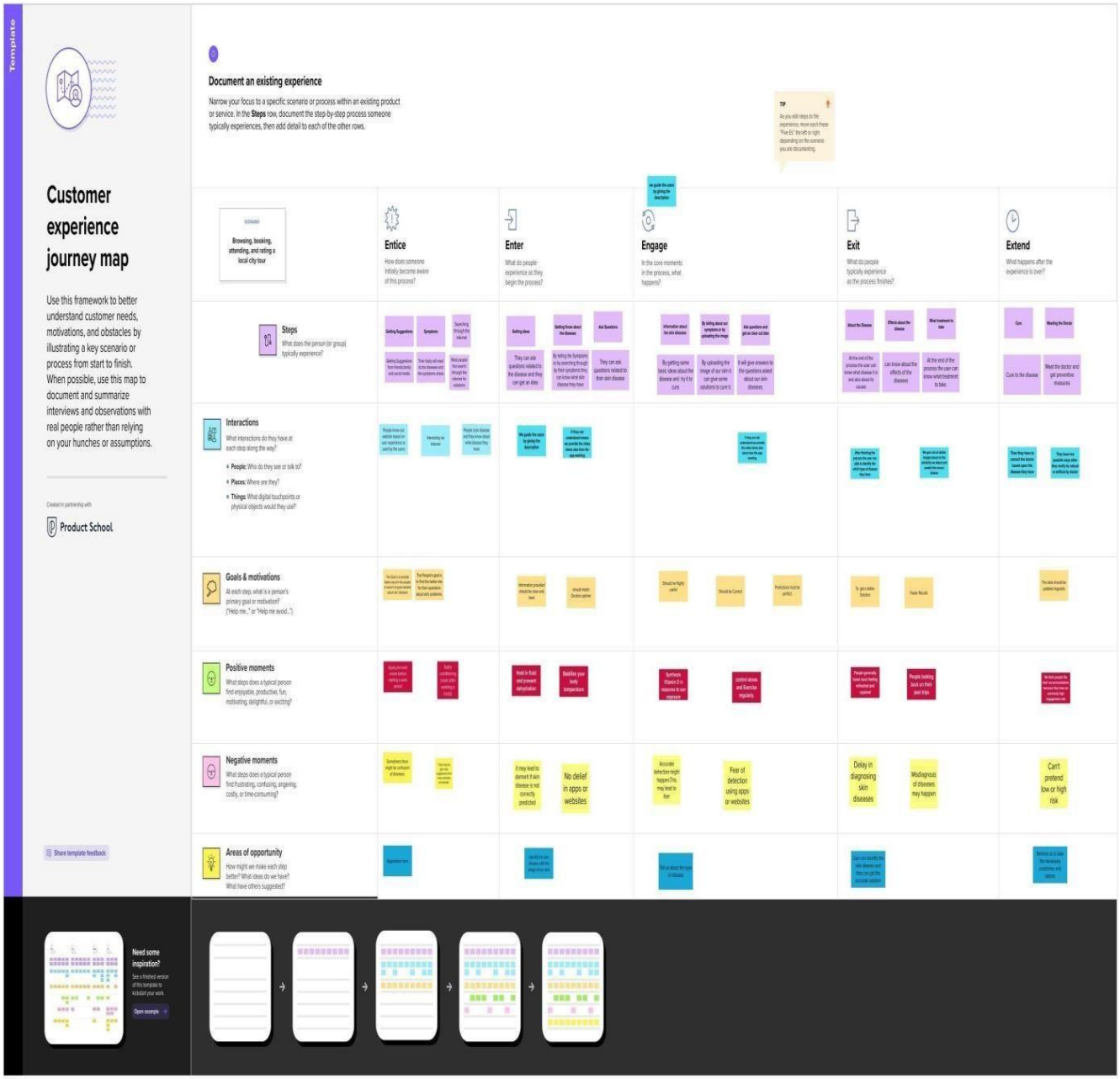
Using the decomposed images, in one instance, we input three images, namely, the original, the haemoglobin, and the melanin images, to our U-Net and obtained a single black-andwhite mask image as output as shown

RESULT:

We have shown that even without a large dataset and high-quality images, it is possible to achieve sufficient accuracy rates. In addition, we have shown that current state-of-the-art CNN models can outperform models created by previous research, through proper data pre-processing, selfsupervised learning, transfer learning, and special CNN architecture techniques. Furthermore, with accurate segmentation, we gain knowledge of the location of the disease, which is useful in the preprocessing of data used in classification, as it allows the CNN model to focus on the area of interest. Lastly, unlike previous studies, our method provides a solution to classify multiple diseases within a single image. With higher quality and a larger quantity of data, it will be viable to use state-of-the-art models to enable the use of CAD in the field of dermatology .

PROJECT DESIGN PHASE 2

Customer Journey Map :



Solution Requirements :

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Mobile Number Registration through Google Account Registration through Facebook
FR-2	User Confirmation	Confirmation via Email Confirmation via Call Confirmation via OTP
FR-3	Patient Image Capturing Process	Provide Access to Capture Image Through Camera Provide Access to Upload Image Through Gallery
FR-4	Patient Medicine Reminder	Remind the Patients to take their Medicines/ointments At right time through remaindering alarm.
FR-5	Suggestion Box	Patients can take suggestions from the Doctors through Chats.
FR-6	Flareup Cycles	Patients can know their medicine level from doctors Through message.

Non-functional Requirements:

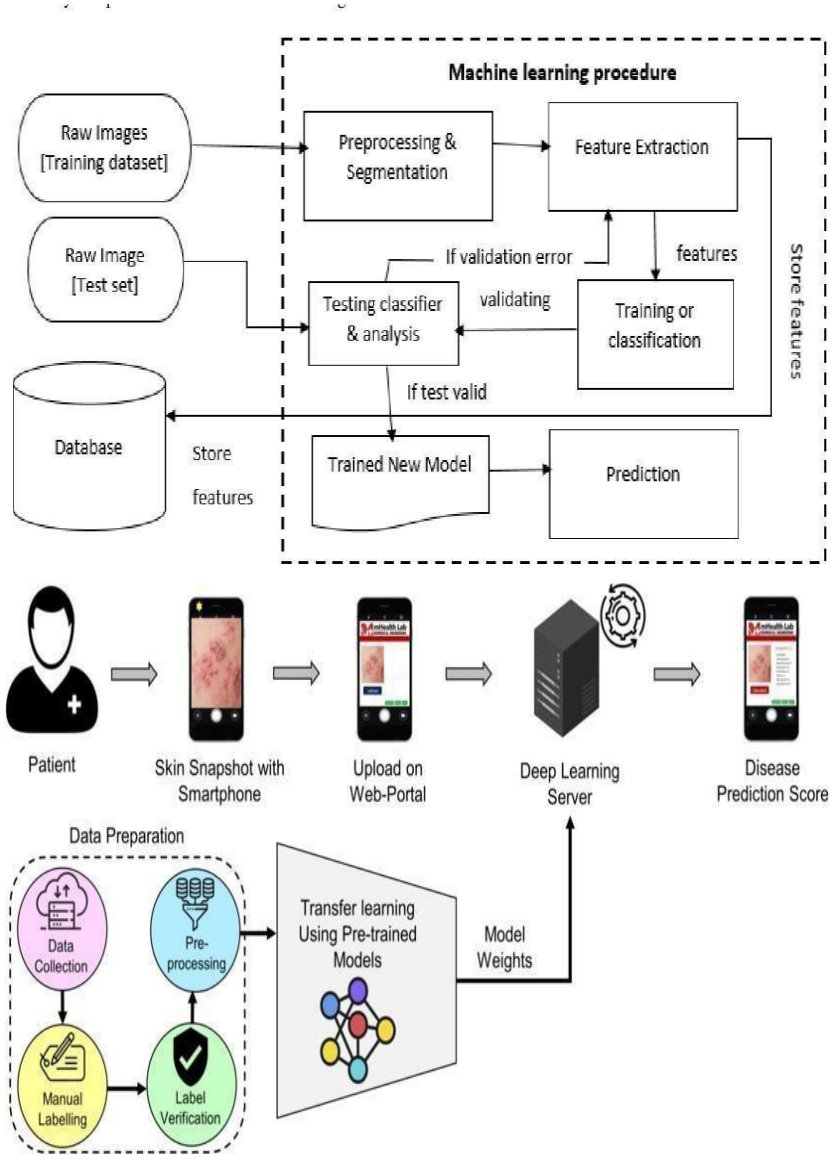
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Our Mobile phone application designed to improve the quality of patient-held photos, and was developed to generate and hold their own skin images to help guide their skin care.
NFR-2	**Security**	Data privacy and security practices may vary based on users and their age

NFR-3	**Reliability**	Easy to use app to get personalized answers to your skin conditions questions.
NFR-4	**Performance**	Good treatments are available for a variety of skin conditions including rash, itchy skin, skin fungus etc.
NFR-5	**Availability**	Our app helps you to screen your skin symptoms and prepare for your practitioner visit.
NFR-6	**Scalability**	The app gives users evidence-based dermatologist approved health information insights on diseases affecting various parts of our body.

Data Flow Diagrams :

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail.		Medium	Sprint-1

		USN-4	As a user, I can register for the application through Gmail.		Medium	Sprint-1

User Stories

Use the below template to list all the user stories for the product

	Login	USN-5	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard	USN-5	As a user, I can Access my Dashboard.		Medium	Sprint-3
Customer (Web user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-4
Customer Care Executive	Solution	USN-5	Responding to each email you receive can make a lasting impression on customers.	Offer a solution for how your company can improve the customer's experience.	High	Sprint-3
Administrator	Manage	USN-5	Do-it-yourself service for delivering Everything.	set of predefined requirements that must be met to mark a user story complete.	High	Sprint-4

Technical Architecture :

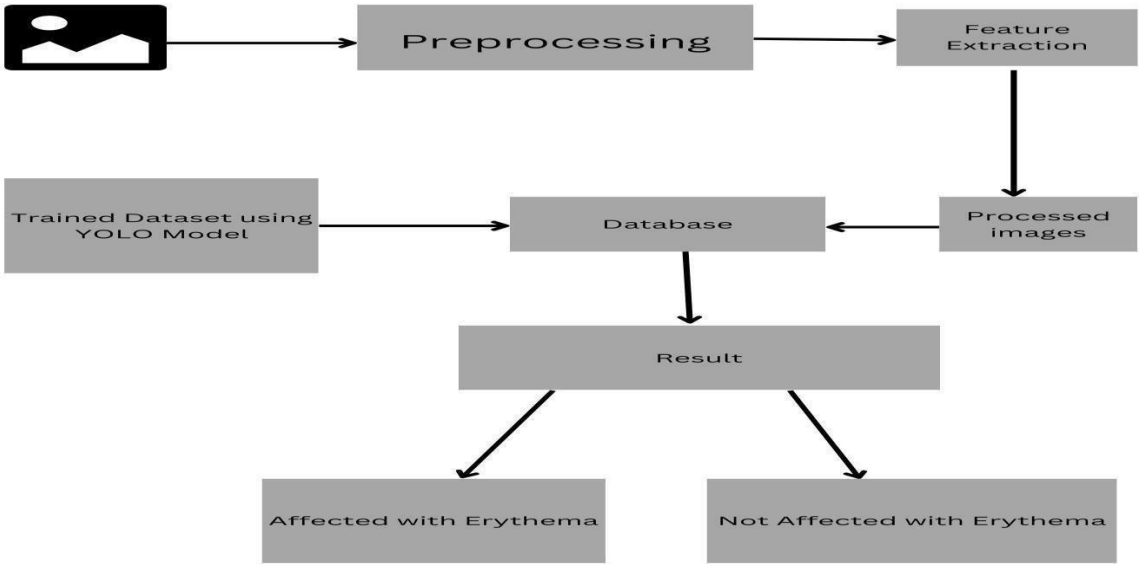


Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI, Chatbot etc.	HTML, CSS, JavaScript .
2.	Application Logic-1	Logic for a process in the application	Python
3.	Application Logic-2	Logic for a process in the application	IBM Watson Assistant
4.	Cloud Database	Database Service on Cloud	IBM Cloudant
5.	Machine Learning Model	Purpose of Machine Learning Model	Object Recognition Model
6.	Infrastructure (Server / Cloud)	Application Deployment on Cloud Server Configuration :	Kubernetes

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	Django
2.	Security Implementations	List all the security / access controls implemented, use of firewalls etc.	Encryptions
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Micro-services)	3-Tier Architecture

PROJECT DEVELOPMENT

INTERFACE

DETECTOR:

```
import os
import sys


def get_parent_dir(n=1):
    """ returns the n-th parent dicrectory of the current
    working directory """
    current_path = os.path.dirname(os.path.abspath(__file__))
    for k in range(n):
        current_path = os.path.dirname(current_path)
    return current_path


src_path = os.path.join(get_parent_dir(1), "2_Training", "src")
utils_path = os.path.join(get_parent_dir(1), "Utils")

sys.path.append(src_path)
sys.path.append(utils_path)

import argparse
from keras_yolo3.yolo import YOLO, detect_video
from PIL import Image
from timeit import default_timer as timer
from utils import load_extractor_model, load_features, parse_input, detect_object
import test
import utils
import pandas as pd
import numpy as np
from Get_File_Paths import GetFileList
import random

os.environ["TF_CPP_MIN_LOG_LEVEL"] = "3"

# Set up folder names for default values
```

```
data_folder = os.path.join(get_parent_dir(n=1), "Data")

image_folder = os.path.join(data_folder, "Source_Images")

image_test_folder = os.path.join(image_folder, "Test_Images")

detection_results_folder = os.path.join(image_folder, "Test_Image_Detection_Results")
detection_results_file = os.path.join(detection_results_folder, "Detection_Results.csv")

model_folder = os.path.join(data_folder, "Model_Weights")

model_weights = os.path.join(model_folder, "trained_weights_final.h5")
model_classes = os.path.join(model_folder, "data_classes.txt")

anchors_path = os.path.join(src_path, "keras_yolo3", "model_data", "yolo_anchors.txt")

FLAGS = None

if __name__ == "__main__":
    # Delete all default flags
    parser = argparse.ArgumentParser(argument_default=argparse.SUPPRESS)
    """
    Command line options
    """

    parser.add_argument(
        "--input_path",
        type=str,
        default=image_test_folder,
        help="Path to image/video directory. All subdirectories will be included. Default is "
        + image_test_folder,
    )

    parser.add_argument(
        "--output",
        type=str,
        default=detection_results_folder,
        help="Output path for detection results. Default is "
        + detection_results_folder,
    )

    parser.add_argument(
        "--no_save_img",
        default=False,
        action="store_true",
        help="Only save bounding box coordinates but do not save output images with annotated boxes. Default
is False.",
    )

    parser.add_argument(
        "--file_types",
        "--names-list",
```

```
    nargs="*",
    default=[],
    help="Specify list of file types to include. Default is --file_types .jpg .jpeg .png .mp4",
)

parser.add_argument(
    "--yolo_model",
    type=str,
    dest="model_path",
    default=model_weights,
    help="Path to pre-trained weight files. Default is " + model_weights,
)

parser.add_argument(
    "--anchors",
    type=str,
    dest="anchors_path",
    default=anchors_path,
    help="Path to YOLO anchors. Default is " + anchors_path,
)

parser.add_argument(
    "--classes",
    type=str,
    dest="classes_path",
    default=model_classes,
    help="Path to YOLO class specifications. Default is " + model_classes,
)

parser.add_argument(
    "--gpu_num", type=int, default=1, help="Number of GPU to use. Default is 1"
)

parser.add_argument(
    "--confidence",
    type=float,
    dest="score",
    default=0.25,
    help="Threshold for YOLO object confidence score to show predictions. Default is 0.25.",
)

parser.add_argument(
    "--box_file",
    type=str,
    dest="box",
    default=detection_results_file,
    help="File to save bounding box results to. Default is "
    + detection_results_file,
)

parser.add_argument(
    "--postfix",
```

```
    type=str,
    dest="postfix",
    default="_disease",
    help='Specify the postfix for images with bounding boxes. Default is "_disease"',
)

FLAGS = parser.parse_args()

save_img = not FLAGS.no_save_img

file_types = FLAGS.file_types

if file_types:
    input_paths = GetFileList(FLAGS.input_path, endings=file_types)
else:
    input_paths = GetFileList(FLAGS.input_path)

# Split images and videos
img_endings = (".jpg", ".jpeg", ".png")
vid_endings = (".mp4", ".mpeg", ".mpg", ".avi")

input_image_paths = []
input_video_paths = []
for item in input_paths:
    if item.endswith(img_endings):
        input_image_paths.append(item)
    elif item.endswith(vid_endings):
        input_video_paths.append(item)

output_path = FLAGS.output
if not os.path.exists(output_path):
    os.makedirs(output_path)

# define YOLO detector
yolo = YOLO(
    **{
        "model_path": FLAGS.model_path,
        "anchors_path": FLAGS.anchors_path,
        "classes_path": FLAGS.classes_path,
        "score": FLAGS.score,
        "gpu_num": FLAGS.gpu_num,
        "model_image_size": (416, 416),
    }
)

# Make a dataframe for the prediction outputs
out_df = pd.DataFrame(
    columns=[
        "image",
        "image_path",
        "xmin",
        "ymin",
```

```
        "xmax",
        "ymax",
        "label",
        "confidence",
        "x_size",
        "y_size",
    ]
)

# labels to draw on images
class_file = open(FLAGS.classes_path, "r")
input_labels = [line.rstrip("\n") for line in class_file.readlines()]
print("Found {} input labels: {} ...".format(len(input_labels), input_labels))

if input_image_paths:
    print(
        "Found {} input images: {} ...".format(
            len(input_image_paths),
            [os.path.basename(f) for f in input_image_paths[:5]],
        )
    )
    start = timer()
    text_out = ""

    # This is for images
    for i, img_path in enumerate(input_image_paths):
        print(img_path)
        prediction, image,lat,lon= detect_object(
            yolo,
            img_path,
            save_img=save_img,
            save_img_path=FLAGS.output,
            postfix=FLAGS.postfix,
        )
        print(lat,lon)
        y_size, x_size, _ = np.array(image).shape
        for single_prediction in prediction:
            out_df = out_df.append(
                pd.DataFrame(
                    [
                        [
                            os.path.basename(img_path.rstrip("\n")),
                            img_path.rstrip("\n"),
                        ]
                        + single_prediction
                        + [x_size, y_size]
                    ],
                    columns=[
                        "image",
                        "image_path",
                        "xmin",
                        "ymin",
```

```
                "xmax",
                "ymax",
                "label",
                "confidence",
                "x_size",
                "y_size",
            ],
        )
    )
end = timer()
print(
    "Processed {} images in {:.1f}sec - {:.1f}FPS".format(
        len(input_image_paths),
        end - start,
        len(input_image_paths) / (end - start),
    )
)
out_df.to_csv(FLAGS.box, index=False)

# This is for videos
if input_video_paths:
    print(
        "Found {} input videos: {} ...".format(
            len(input_video_paths),
            [os.path.basename(f) for f in input_video_paths[:5]],
        )
    )
    start = timer()
    for i, vid_path in enumerate(input_video_paths):
        output_path = os.path.join(
            FLAGS.output,
            os.path.basename(vid_path).replace(".", FLAGS.postfix + "."),
        )
        detect_video(yolo, vid_path, output_path=output_path)

    end = timer()
    print(
        "Processed {} videos in {:.1f}sec".format(
            len(input_video_paths), end - start
        )
    )
# Close the current yolo session
        yolo.close_session()


    SETUP TOOL

    DISTUTILS:
    """distutils.command

    Package containing implementation of all the standard Distutils
    commands."""
```

```
__all__ = ['build',
           'build_py',
           'build_ext',
           'build_clib',
           'build_scripts',
           'clean',
           'install',
           'install_lib',
           'install_headers',
           'install_scripts',
           'install_data',
           'sdist',
           'register',
           'bdist',
           'bdist_dumb',
           'bdist_rpm',
           'bdist_wininst',
           'check',
           'upload',
           # These two are reserved for future use:
           #'bdist_sdux',
           #'bdist_pkgtool',
           # Note:
           # bdist_packager is not included because it only provides
           # an abstract base class
          ]
```

TEMPLATES

INDEX.HTML:

```
<!doctype html>
<html>
   <head>
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <title>Skinnovation</title>
      <link rel="stylesheet" type="text/css" href="../static/css/trial.css">
      </head>
      <body>
         <div class="conatiner">
         <header>
            <a href="#" class="logo">Skinnovation</a>
            <ul>
         <li><a href="{{url_for('index')}}"class="active">Home</a></li>
         <li><a href="{{url_for('login')}}">Login</a></li>
         <li><a href="{{ url_for('register')}}">Register</a></li>
         <li><a href="{{ url_for('prediction')}}">Prediction</a></li>
         </ul>
         </header>
         <section>
            <img src="../static/img/stars.png" id="stars">
            <img src="../static/img/moon.png" id="moon">
```

```
        <img src="../static/img/mountains_behind.png" id="mountains_behind">

        <h2 id="text1" style="top: 10px;">Skinnovation</h2>
        <a href="#sec" id="btn">About</a>
        <img src="../static/img/mountains_front.png" id="mountains_front"  >

      </section>
      <div class="divi" >.</div>
      <div class="sec" id="sec">


          <h2>Problem</h2>
          <p>Now a day’s people are suffering from skin diseases, More than 125 million people suffering from Psoriasis also skin cancer rate is rapidly increasing over the last few decades especially Melanoma is most diversifying skin cancer. If skin diseases are not treated at an earlier stage, then it may lead to complications in the body including spreading of the infection from one individual to the other. The skin diseases can be prevented by investigating the infected region at an early stage. The characteristic of the skin images is diversified so that it is a challenging job to devise an efficient and robust algorithm for automatic detection of skin disease and its severity. Skin tone and skin colour play an important role in skin disease detection. Colour and coarseness of skin are visually different. Automatic processing of such images for skin analysis requires quantitative discriminator to differentiate the diseases.<br>
            <br>
            <br>
            <!-- <video controls loop autoplay muted playsinline>

              <source src="main.mp4"
                type="video/mp4">
            </video> -->

          </p>


          <h2>Solution</h2>
          <p> To overcome the above problem we are building a model which is used for the prevention and early detection of skin cancer, psoriasis. Basically, skin disease diagnosis depends on the different characteristics like colour, shape, texture etc. Here the person can capture the images of skin and then the image will be sent the trained model. The model analyses the image and detect whether the person is having skin disease or not.
        <br><br>
          </p>
          <div class="align">
            <p class="l">what is this ?</p> <img src="../static/img/1.gif" class="two">

        <br>
      </div>
      <br>
      <div class="align">
        <p class="j">How to do? </p>  <img src="../static/img/2.gif" class="one"><br>
      <br>
    </div>
    <br>
```

```
      <div class="align">
        <p class="l">what happens ?</p>  <img src="../static/img/3.gif" class="two"><br>
      <br>
    </div>
    <br>
    <div class="align">
      <p class="j">What's Final result ?</p>  <img src="../static/img/4.gif" class="one"><br>
    <br>
</div>
<br>
<div class="align">
    <p class="l">So....</p>  <img src="../static/img/5.gif" class="two"><br>
<br>
</div>


            <!-- <h2>Parallax Scrolling Effect</h2>
            <p>Lorem, ipsum dolor sit amet consectetur adipisicing elit. Asperiores molestiae placeat eius quo neque earum! Lorem ipsum dolor sit, amet consectetur adipisicing elit. Saepe accusantium ducimus velit repellat veritatis placeat facere dicta incidunt, quod illum natus, cupiditate rerum veniam. Error minus voluptatum iste beatae tempore modi neque ullam esse quos possimus est corporis adipisci, dolores dolor minima eligendi sunt sed!
            Lorem ipsum, dolor sit amet consectetur adipisicing elit. Dolore dolor laboriosam voluptate assumenda corrupti pariatur ullam nulla totam, quos consequuntur. Reprehenderit cumque voluptatum aliquam molestias consequatur totam blanditiis error vitae?
            Lorem ipsum dolor sit amet consectetur adipisicing elit. Eveniet cum obcaecati, inventore sed voluptatibus aliquam sint sunt magnam, expedita blanditiis ea. Eum, delectus. Explicabo dolorum eius ducimus provident asperiores corporis perspiciatis, numquam maxime saepe labore tempore unde culpa deleniti blanditiis corrupti itaque nisi commodi cupiditate sunt, cumque qui! A laudantium maiores pariatur repellendus doloribus sint tenetur tempore rem quas in ut voluptatum accusamus illo harum corporis expedita laborum, vero necessitatibus beatae minima, temporibus autem inventore. Itaque harum modi omnis molestias voluptates error expedita voluptatibus optio nobis? Voluptate accusantium atque provident maiores ullam nihil adipisci, suscipit sed, sit laborum illo temporibus!<br><br>
            Lorem, ipsum dolor sit amet consectetur adipisicing elit. Asperiores molestiae placeat eius quo neque earum! Lorem ipsum dolor sit, amet consectetur adipisicing elit. Saepe accusantium ducimus velit repellat veritatis placeat facere dicta incidunt, quod illum natus, cupiditate rerum veniam. Error minus voluptatum iste beatae tempore modi neque ullam esse quos possimus est corporis adipisci, dolores dolor minima eligendi sunt sed!
            Lorem ipsum, dolor sit amet consectetur adipisicing elit. Dolore dolor laboriosam voluptate assumenda corrupti pariatur ullam nulla totam, quos consequuntur. Reprehenderit cumque voluptatum aliquam molestias consequatur totam blanditiis error vitae?
            Lorem ipsum dolor sit amet consectetur adipisicing elit. Eveniet cum obcaecati, inventore sed voluptatibus aliquam sint sunt magnam, expedita blanditiis ea. Eum, delectus. Explicabo dolorum eius ducimus provident asperiores corporis perspiciatis, numquam maxime saepe labore tempore unde culpa deleniti blanditiis corrupti itaque nisi commodi cupiditate sunt, cumque qui! A laudantium maiores pariatur repellendus doloribus sint tenetur tempore rem quas in ut voluptatum accusamus illo harum corporis expedita laborum, vero necessitatibus beatae minima, temporibus autem inventore. Itaque harum modi omnis molestias voluptates error expedita voluptatibus optio nobis? Voluptate accusantium atque provident maiores ullam nihil adipisci, suscipit sed, sit laborum illo temporibus!<br><br>
            Lorem, ipsum dolor sit amet consectetur adipisicing elit. Asperiores molestiae placeat eius
```

quo neque earum! Lorem ipsum dolor sit, amet consectetur adipisicing elit. Saepe accusantium ducimus velit repellat veritatis placeat facere dicta incidunt, quod illum natus, cupiditate rerum veniam. Error minus voluptatum iste beatae tempore modi neque ullam esse quos possimus est corporis adipisci, dolores dolor minima eligendi sunt sed!

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Dolore dolor laboriosam voluptate assumenda corrupti pariatur ullam nulla totam, quos consequuntur. Reprehenderit cumque voluptatum aliquam molestias consequatur totam blanditiis error vitae?

Lorem ipsum dolor sit amet consectetur adipisicing elit. Eveniet cum obcaecati, inventore sed voluptatibus aliquam sint sunt magnam, expedita blanditiis ea. Eum, delectus. Explicabo dolorum eius ducimus provident asperiores corporis perspiciatis, numquam maxime saepe labore tempore unde culpa deleniti blanditiis corrupti itaque nisi commodi cupiditate sunt, cumque qui! A laudantium maiores pariatur repellendus doloribus sint tenetur tempore rem quas in ut voluptatum accusamus illo harum corporis expedita laborum, vero necessitatibus beatae minima, temporibus autem inventore. Itaque harum modi omnis molestias voluptates error expedita voluptatibus optio nobis? Voluptate accusantium atque provident maiores ullam nihil adipisci, suscipit sed, sit laborum illo temporibus!


```
            </p> -->
        </div>
        <script>
            let stars = document.getElementById('stars');
            let moon = document.getElementById('moon');
            let mountains_behind = document.getElementById('mountains_behind');
            let mountains_front = document.getElementById('mountains_front');
            let btn = document.getElementById('btn');
            let text1 = document.getElementById('text1');
            let header= document.querySelector('header');

            window.addEventListener('scroll',function(){
                let value = window.scrollY;
                stars.style.left = value *0.25 + 'px';
                // moon.style.top = value *1.05 + 'px';
                mountains_behind.style.top = value *0.5 + 'px';
                mountains_front.style.top = value * 0 + 'px';
                text1.style.marginRight=value*4+'px';
                text1.style.marginTop = value * 1.5 + 'px';
                btn.style.marginTop = value * 1.5 + 'px';
                header.style.top=value*0.5+'px';
            })
        </script>
        </div>
    </body>
    </html>
```

LOGIN.HTML:

```
<html lang="en">
<head>
<meta charset="UTF-8">
<meta http-euiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Login & Register</title>
<link rel="stylesheet" href="../static/css/style1.css">
```

```
<link rel="stylesheet"  href="https://use.fontawesome.com/releases/v5.6.3/css/all.css">
</head>
<body>
        <div class="container" id="container">
                <div class="form-container sign-up-container">
                        <form action="{{ url_for('register')}}">
                                <h1>Click sign-up for Registration</h1><br>
                                <!-- <div class="social-container">
                                        <a href="#" class="social"><i class="fab fa-facebook"></i></a>
                                        <a href="#" class="social"><i class="fab fa-google-plus-g"></i></a>
                                        <a href="#" class="social"><i class="fab fa-linkedin-in"></i></a>
                                </div> -->
                                        <!-- <span>or use you email for registration</span>
                                        <input type="text" placeholder="Name">
                                        <input type="email" placeholder="Email">
                                        <input type="password" placeholder="Password"> -->

                                        <button>Sign-up</button>


                        </form>
                </div>
                <div class="form-container sign-in-container">
                        <form action="{{url_for('afterlogin')}}" method="post">
                                <h1>Sign in</h1>
                                <!-- <div class="social-container">
                                        <a href="#" class="social"><i class="fab fa-facebook"></i></a>
                                        <a href="#" class="social"><i class="fab fa-google-plus-g"></i></a>
                                        <a href="#" class="social"><i class="fab fa-linkedin-in"></i></a>
                                </div>
                                        <span>or use your account</span> -->

                                        <input type="email"  name="_id" placeholder="Email">
                                        <input type="password" name="psw" placeholder="Password">
                                        <button type="submit">Sign In</button>{{pred}}


                        </form>

                </div>
                <div class="overlay-container">
                        <div class="overlay">
                                <div class="overlay-panel overlay-left">
                                        <h2>Already have an account?</h2>
                                        <button class="press" id="signIn">Sign In</button>

                                </div>


                                <div class="overlay-panel overlay-right">
                                        <h2> NEW HERE?</h2>
                                        <p>Signup to make a journey with us... </p>
```

```
                        <button class="press" id="signUp">Sign Up</button>
                    </div>


                </div>
            </div>
        </div>
        <script>
            const signUpButton=document.getElementById("signUp");
            const signInButton=document.getElementById("signIn");
            const container=document.getElementById("container");
            signUpButton.addEventListener('click',()=>{
                container.classList.add("right-panel-active");});

            signInButton.addEventListener('click',()=>{
                container.classList.remove("right-panel-active");
            });
        </script>



</body>
</html>
```

LOGOUT.HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Skinnovation</title>
<link rel="stylesheet" href="../static/css/logout.css">
</head>
<body>
<nav>
<div class='heading'>

<h1>Skinnovation</h1>
</div>

<ul class='nav-links'>

<li><a href="{{ url_for('index')}}">Home</a></li>
<li><a href="{{ url_for('login')}}">Login</a></li>
<li><a  href="{{ url_for('register')}}">Registration</a></li>

</ul>
</nav>
<div class="title">Successfully Logged Out!</div>
```

```
<div class="wordings">Login for more information</div>
<button class='loginbtn' type='submit'>Login</button>
</body>
</html>
```

PREDICTION.HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <!--Bootstrap -->
   <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
   <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-
KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
crossorigin="anonymous"></script>
   <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"
integrity="sha384-
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
crossorigin="anonymous"></script>
   <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"
integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
crossorigin="anonymous"></script>


   <script src="https://kit.fontawesome.com/8b9cdc2059.js"
crossorigin="anonymous"></script>
   <link
href="https://fonts.googleapis.com/css2?family=Akronim&family=Roboto&display=swap"
rel="stylesheet">
   <link rel="stylesheet" href="../static/css/predict.css">

   <script defer src="../static/js/JScript.js"></script>
   <title>Prediction</title>
</head>
<body>
   <header id="head" class="header">
      <section  id="navbar">
            <h1 class="nav-heading"></i>Skin Disease Detection</h1>
         <div class="nav--items">
            <ul>
               <li><a  href="{{ url_for('index')}}">Home</a></li>
                              <li><a href="{{ url_for('logout')}}">Logout</a></li>
           <!-- <li><a href="#about">About</a></li>
```

```
            <li><a href="#services">Services</a></li> -->

            </ul>
          </div>
        </section>
    </header>
    <!-- dataset/Training/metal/metal326.jpg -->
        </br>
    <section id="prediction">
    <h2 class="title text-muted">Skinnovation- AI-based localization and classification of skin disease with erythema</h1>
    <div class="line" style="width: 1000px;"></div>
                    </section>
                    </br>
          <section id="about">

<div class="body">
<div class="left">
    <p>
       Nowadays people are suffering from skin diseases, More than 125 million people suffering from Psoriasis also skin cancer rate is rapidly increasing over the last few decades especially Melanoma is most diversifying skin cancer. If skin diseases are not treated at an earlier stage, then it may lead to complications in the body including spreading of the infection from one individual to the other. The skin diseases can be prevented by investigating the infected region at an early stage. The characteristic of the skin images is diversified so that it is a challenging job to devise an efficient and robust algorithm for automatic detection of skin disease and its severity. Skin tone and skin colour play an important role in skin disease detection. Colour and coarseness of skin are visually different. Automatic processing of such images for skin analysis requires quantitative discriminator to differentiate the diseases.

                    </p>
</div>
<div class="left">

    <div class="prediction-input">
        <img class="d-block" src="../static/img/img1.jpg" alt="Second slide" style="width:80%!important; align-items:center; padding-left:100px">
        </br>
              <form id="form" action="/result" method="post"  enctype="multipart/form-data">

                <input type="submit" class="submitbtn" value="Click Me! For a Demo">
              </form>
          </div>
          <h5 style="color:Red;">
          <b style="color:Red">prediction<b>
         </h5>
</div>
</div>
</section>
    <section id="footer">
```

```
   </section>
</body>

</html>
```

REGISTER.HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Skinnovation</title>
<link rel="stylesheet" href="../static/css/sti.css">
</head>
<body>



<header>
   <a href="#" class="logo">Skinnovation</a>
   <ul>
<li><a href="{{ url_for('index')}}" class="active">Home</a></li>
<li><a href="{{ url_for('login')}}">Login</a></li>
<li><a href="{{ url_for('register')}}">Register</a></li>
<li><a href="{{ url_for('prediction')}}">Prediction</a></li>
</ul>
</header>




<form action="{{url_for('afterreg')}}" method="post">

<div class="container">

<h1 class="reg">Registration</h1>
<hr/>

<div class="name"><label  for="fn"><b>First Name</b></label>
   <input type="text" placeholder="First Name" name='fn' id='fn' required/>

   <label for="eml"><b>Enter Email ID</b></label>
   <input type="email" placeholder="Enter Email ID" name="_id" required/>

   <label for="psw"><b>Password</b></label>

   <input type="password" placeholder="Enter Password" name="psw"  required/>

   <hr/>
```

```
   <button class="registerbtn" type="submit">Register</button>{{pred}}
   </form><footer>
   <p class="account">Already have an Account? <a href="{{ url_for('login')}}">Login
Here</a></p>
   </footer></div>




</body>

</html>
```

APPY.PY

```
import re
import numpy as np
import os
from flask import Flask, app,request,render_template
import sys
from flask import Flask, request, render_template, redirect, url_for
import argparse
from tensorflow import keras
from PIL import Image
from timeit import default_timer as timer
import test
import pandas as pd
import numpy as np
import random

def get_parent_dir(n=1):
   """ returns the n-th parent dicrectory of the current
   working directory """
   current_path = os.path.dirname(os.path.abspath(__file__))
   for k in range(n):
      current_path = os.path.dirname(current_path)
   return current_path


src_path =r'C:\Users\MadhuVasanth1606\Desktop\yolo_structure\2_Training\src'
print(src_path)
utils_path = r'C:\Users\MadhuVasanth1606\Desktop\yolo_structure\Utils'
print(utils_path)

sys.path.append(src_path)
sys.path.append(utils_path)

import argparse
from keras_yolo3.yolo import YOLO, detect_video
from PIL import Image
from timeit import default_timer as timer
from utils import load_extractor_model, load_features, parse_input, detect_object
```

```
import test
import utils
import pandas as pd
import numpy as np
from Get_File_Paths import GetFileList
import random

os.environ["TF_CPP_MIN_LOG_LEVEL"] = "3"

# Set up folder names for default values
data_folder = os.path.join(get_parent_dir(n=1), "yolo_structure", "Data")

image_folder = os.path.join(data_folder, "Source_Images")

image_test_folder = os.path.join(image_folder, "Test_Images")

detection_results_folder = os.path.join(image_folder, "Test_Image_Detection_Results")
detection_results_file = os.path.join(detection_results_folder, "Detection_Results.csv")

model_folder = os.path.join(data_folder, "Model_Weights")

model_weights = os.path.join(model_folder, "trained_weights_final.h5")
model_classes = os.path.join(model_folder, "data_classes.txt")

anchors_path = os.path.join(src_path, "keras_yolo3", "model_data", "yolo_anchors.txt")

FLAGS = None


from cloudant.client import Cloudant

# Authenticate using an IAM API key
client = Cloudant.iam('5b73f72f-2449-4298-88e8-3f887f8bbd2d-
bluemix','t3wXXORf8KoIMLzYFX2sk4e22uluSBKhM9-K4Q5b1zuK', connect=True)


# Create a database using an initialized client
my_database = client.create_database('skindisease')


app=Flask(__name__)

#default home page or route
@app.route('/')
def index():
    return render_template('index.html')



@app.route('/index.html')
def home():
    return render_template("index.html")
```

```
#registration page
@app.route('/register')
def register():
    return render_template('register.html')

@app.route('/afterreg', methods=['POST'])
def afterreg():
    x = [x for x in request.form.values()]
    print(x)
    data = {
    '_id': x[1], # Setting _id is optional
    'name': x[0],
    'psw':x[2]
    }
    print(data)

    query = {'_id': {'$eq': data['_id']}}

    docs = my_database.get_query_result(query)
    print(docs)

    print(len(docs.all()))

    if(len(docs.all())==0):
        url = my_database.create_document(data)
        #response = requests.get(url)
        return render_template('register.html', pred="Registration Successful, please login using
your details")
    else:
        return render_template('register.html', pred="You are already a member, please login
using your details")

#login page
@app.route('/login')
def login():
    return render_template('login.html')

@app.route('/afterlogin',methods=['POST'])
def afterlogin():
    user = request.form['_id']
    passw = request.form['psw']
    print(user,passw)

    query = {'_id': {'$eq': user}}

    docs = my_database.get_query_result(query)
    print(docs)

    print(len(docs.all()))
```

```
    if(len(docs.all())==0):
        return render_template('login.html', pred="The username is not found.")
    else:
        if((user==docs[0][0]['_id'] and passw==docs[0][0]['psw'])):
            return redirect(url_for('prediction'))
        else:
            print('Invalid User')


@app.route('/logout')
def logout():
    return render_template('logout.html')

@app.route('/prediction')
def prediction():
    return render_template('prediction.html')


@app.route('/result',methods=["GET","POST"])
def res():
    # Delete all default flags
    parser = argparse.ArgumentParser(argument_default=argparse.SUPPRESS)
    """
    Command line options
    """

    parser.add_argument(
        "--input_path",
        type=str,
        default=image_test_folder,
        help="Path to image/video directory. All subdirectories will be included. Default is "
        + image_test_folder,
    )

    parser.add_argument(
        "--output",
        type=str,
        default=detection_results_folder,
        help="Output path for detection results. Default is "
        + detection_results_folder,
    )

    parser.add_argument(
        "--no_save_img",
        default=False,
        action="store_true",
        help="Only save bounding box coordinates but do not save output images with annotated
boxes. Default is False.",
    )

    parser.add_argument(
```

```
    "--file_types",
    "--names-list",
    nargs="*",
    default=[],
    help="Specify list of file types to include. Default is --file_types .jpg .jpeg .png .mp4",
)

parser.add_argument(
    "--yolo_model",
    type=str,
    dest="model_path",
    default=model_weights,
    help="Path to pre-trained weight files. Default is " + model_weights,
)

parser.add_argument(
    "--anchors",
    type=str,
    dest="anchors_path",
    default=anchors_path,
    help="Path to YOLO anchors. Default is " + anchors_path,
)

parser.add_argument(
    "--classes",
    type=str,
    dest="classes_path",
    default=model_classes,
    help="Path to YOLO class specifications. Default is " + model_classes,
)

parser.add_argument(
    "--gpu_num", type=int, default=1, help="Number of GPU to use. Default is 1"
)

parser.add_argument(
    "--confidence",
    type=float,
    dest="score",
    default=0.25,
    help="Threshold for YOLO object confidence score to show predictions. Default is 0.25.",
)

parser.add_argument(
    "--box_file",
    type=str,
    dest="box",
    default=detection_results_file,
    help="File to save bounding box results to. Default is "
    + detection_results_file,
)
```

```
parser.add_argument(
    "--postfix",
    type=str,
    dest="postfix",
    default="_disease",
    help='Specify the postfix for images with bounding boxes. Default is "_disease"',
)

FLAGS = parser.parse_args()

save_img = not FLAGS.no_save_img

file_types = FLAGS.file_types
#print(input_path)

if file_types:
    input_paths = GetFileList(FLAGS.input_path, endings=file_types)
    print(input_paths)
else:
    input_paths = GetFileList(FLAGS.input_path)
    print(input_paths)

# Split images and videos
img_endings = (".jpg", ".jpeg", ".png")
vid_endings = (".mp4", ".mpeg", ".mpg", ".avi")

input_image_paths = []
input_video_paths = []
for item in input_paths:
    if item.endswith(img_endings):
        input_image_paths.append(item)
    elif item.endswith(vid_endings):
        input_video_paths.append(item)

output_path = FLAGS.output
if not os.path.exists(output_path):
    os.makedirs(output_path)

# define YOLO detector
yolo = YOLO(
    **{
        "model_path": FLAGS.model_path,
        "anchors_path": FLAGS.anchors_path,
        "classes_path": FLAGS.classes_path,
        "score": FLAGS.score,
        "gpu_num": FLAGS.gpu_num,
        "model_image_size": (416, 416),
    }
)

# Make a dataframe for the prediction outputs
out_df = pd.DataFrame(
```

```
    columns=[
        "image",
        "image_path",
        "xmin",
        "ymin",
        "xmax",
        "ymax",
        "label",
        "confidence",
        "x_size",
        "y_size",
    ]
)

# labels to draw on images
class_file = open(FLAGS.classes_path, "r")
input_labels = [line.rstrip("\n") for line in class_file.readlines()]
print("Found {} input labels: {} ...".format(len(input_labels), input_labels))

if input_image_paths:
    print(
        "Found {} input images: {} ...".format(
            len(input_image_paths),
            [os.path.basename(f) for f in input_image_paths[:5]],
        )
    )
    start = timer()
    text_out = ""

    # This is for images
    for i, img_path in enumerate(input_image_paths):
        print(img_path)
        prediction, image,lat,lon= detect_object(
            yolo,
            img_path,
            save_img=save_img,
            save_img_path=FLAGS.output,
            postfix=FLAGS.postfix,
        )
        print(lat,lon)
        y_size, x_size, _ = np.array(image).shape
        for single_prediction in prediction:
            out_df = out_df.append(
                pd.DataFrame(
                    [
                        [
                            os.path.basename(img_path.rstrip("\n")),
                            img_path.rstrip("\n"),
                        ]
                        + single_prediction
                        + [x_size, y_size]
                    ],
```

```
                columns=[
                    "image",
                    "image_path",
                    "xmin",
                    "ymin",
                    "xmax",
                    "ymax",
                    "label",
                    "confidence",
                    "x_size",
                    "y_size",
                ],
            )
        )
    end = timer()
    print(
        "Processed {} images in {:.1f}sec - {:.1f}FPS".format(
            len(input_image_paths),
            end - start,
            len(input_image_paths) / (end - start),
        )
    )
    out_df.to_csv(FLAGS.box, index=False)

# This is for videos
if input_video_paths:
    print(
        "Found {} input videos: {} ...".format(
            len(input_video_paths),
            [os.path.basename(f) for f in input_video_paths[:5]],
        )
    )
    start = timer()
    for i, vid_path in enumerate(input_video_paths):
        output_path = os.path.join(
            FLAGS.output,
            os.path.basename(vid_path).replace(".", FLAGS.postfix + "."),
        )
        detect_video(yolo, vid_path, output_path=output_path)

    end = timer()
    print(
        "Processed {} videos in {:.1f}sec".format(
            len(input_video_paths), end - start
        )
    )
# Close the current yolo session
yolo.close_session()
return render_template('prediction.html')


""" Running our application """
```

```
    if __name__ == "__main__":
        app.run(debug=True)
```

DOWNLOAD AND CONVERT YOLO:

```
import os
import subprocess
import time
import sys
import argparse
import requests
import progressbar

FLAGS = None

root_folder = os.path.dirname(os.path.abspath(__file__))
download_folder = os.path.join(root_folder, "src", "keras_yolo3")

if __name__ == "__main__":
    # Delete all default flags
    parser = argparse.ArgumentParser(argument_default=argparse.SUPPRESS)
    """
    Command line options
    """
    parser.add_argument(
        "--download_folder",
        type=str,
        default=download_folder,
        help="Folder to download weights to. Default is " + download_folder,
    )

    FLAGS = parser.parse_args()

    url = "https://pjreddie.com/media/files/yolov3.weights"
    r = requests.get(url, stream=True)

    f = open(os.path.join(download_folder, "yolov3.weights"), "wb")
    file_size = int(r.headers.get("content-length"))
    chunk = 100
    num_bars = file_size // chunk
    bar = progressbar.ProgressBar(maxval=num_bars).start()
    i = 0
    for chunk in r.iter_content(chunk):
        f.write(chunk)
        bar.update(i)
        i += 1
    f.close()

    call_string = "python convert.py yolov3.cfg yolov3.weights yolo.h5"

    subprocess.call(call_string, shell=True, cwd=download_folder)
```

IMAGE ANNOTATION:

```
from PIL import Image
from os import path, makedirs
import os
import re
import pandas as pd
import sys
import argparse


def get_parent_dir(n=1):
    """ returns the n-th parent dicrectory of the current
    working directory """
    current_path = os.path.dirname(os.path.abspath(__file__))
    for k in range(n):
        current_path = os.path.dirname(current_path)
    return current_path


sys.path.append(os.path.join(get_parent_dir(1), "Utils"))
from Convert_Format import convert_vott_csv_to_yolo

Data_Folder = os.path.join(get_parent_dir(1), "Data")
VoTT_Folder = os.path.join(
    Data_Folder, "Source_Images", "Training_Images", "vott-csv-export"
)
VoTT_csv = os.path.join(VoTT_Folder, "Annotations-export.csv")
YOLO_filename = os.path.join(VoTT_Folder, "data_train.txt")

model_folder = os.path.join(Data_Folder, "Model_Weights")
classes_filename = os.path.join(model_folder, "data_classes.txt")

if __name__ == "__main__":
    # surpress any inhereted default values
    parser = argparse.ArgumentParser(argument_default=argparse.SUPPRESS)
    """
    Command line options
    """
    parser.add_argument(
        "--VoTT_Folder",
        type=str,
        default=VoTT_Folder,
        help="Absolute path to the exported files from the image tagging step with VoTT. Default is "
        + VoTT_Folder,
    )

    parser.add_argument(
        "--VoTT_csv",
        type=str,
        default=VoTT_csv,
        help="Absolute path to the *.csv file exported from VoTT. Default is "
```

```
    + VoTT_csv,
  )
  parser.add_argument(
    "--YOLO_filename",
    type=str,
    default=YOLO_filename,
    help="Absolute path to the file where the annotations in YOLO format should be saved. Default is "
    + YOLO_filename,
  )

  FLAGS = parser.parse_args()

  # Prepare the dataset for YOLO
  multi_df = pd.read_csv(FLAGS.VoTT_csv)
  labels = multi_df["label"].unique()
  labeldict = dict(zip(labels, range(len(labels))))
  multi_df.drop_duplicates(subset=None, keep="first", inplace=True)
  train_path = FLAGS.VoTT_Folder
  convert_vott_csv_to_yolo(
    multi_df, labeldict, path=train_path, target_name=FLAGS.YOLO_filename
  )

  # Make classes file
  file = open(classes_filename, "w")

  # Sort Dict by Values
  SortedLabelDict = sorted(labeldict.items(), key=lambda x: x[1])
  for elem in SortedLabelDict:
    file.write(elem[0] + "\n")
  file.close()
```

REQUIREMENTS:

```
setuptools>=41.0.0
pip>=19.0.0
absl-py==0.7.1
astor==0.8.0
attrs==19.1.0
backcall==0.1.0
bleach==3.1.4
certifi==2019.6.16
chardet==3.0.4
cycler==0.10.0
decorator==4.4.0
defusedxml==0.6.0
progressbar2==3.46.1
entrypoints==0.3
gast==0.2.2
google-pasta==0.1.7
grpcio==1.22.0
h5py==2.9.0
idna==2.8
```

ipykernel==5.1.1
ipython==7.6.1
ipython-genutils==0.2.0
ipywidgets==7.5.0
jedi==0.14.0
Jinja2==2.10.1
joblib==0.13.2
jsonschema==3.0.1
jupyter==1.0.0
jupyter-client==5.3.0
jupyter-console==6.0.0
jupyter-core==4.5.0
Keras==2.2.4
Keras-Applications==1.0.8
Keras-Preprocessing==1.1.0
kiwisolver==1.1.0
Markdown==3.1.1
MarkupSafe==1.1.1
matplotlib==3.0.3
mistune==0.8.4
mpmath==1.1.0
nbconvert==5.5.0
nbformat==4.4.0
notebook==5.7.8
numpy==1.16.4
opencv-python==4.1.0.25
pandas==0.24.2
pandocfilters==1.4.2
parso==0.5.0
pexpect==4.7.0
pickleshare==0.7.5
Pillow==6.2.2
prometheus-client==0.7.1
prompt-toolkit==2.0.9
protobuf==3.8.0
ptyprocess==0.6.0
Pygments==2.4.2
pyparsing==2.4.0
pyrsistent==0.15.3
python-dateutil==2.8.0
pytz==2019.1
PyYAML==5.1.1
pyzmq==18.0.2
qtconsole==4.5.1
requests==2.22.0
scikit-learn==0.21.2
scipy==1.3.0
Send2Trash==1.5.0
six==1.12.0
sklearn==0.0
sympy==1.4
tensorboard==1.15.0

tensorflow==1.15.2
tensorflow-estimator==1.15.0
termcolor==1.1.0
terminado==0.8.2
testpath==0.4.2
tornado==6.0.3
traitlets==4.3.2
urllib3==1.25.3
wcwidth==0.1.7
webencodings==0.5.1
Werkzeug==0.15.4
widgetsnbextension==3.5.0
wrapt==1.11.2

IMAGE PROCESSING FINAL

PROJECT PLANNING PHASE

Prepare Milestone and Activity List :

TITLE	DESCRIPTION	DATE
Literature Survey & Information Gathering	Literature survey on the selected project & gathering information by referring the, technical papers,research publications etc.	22 September 2022

Prepare Empathy Map	Prepare Empathy Map Canvas to capture the user Pains & Gains, Prepare list of problem statements	21 September 2022
Ideation	List the by organizing the brainstormingsession and prioritize the top 3 ideas based on the feasibility & importance.	28 September 2022

Proposed Solution	Prepare the proposed solution document, which includes the novelty, feasibility of idea, business model, social impact, scalability of solution, etc.	09 october2022
Problem Solution Fit	Prepare problem - solution fit document.	10 october2022
Solution Architecture	Prepare solution architecture document.	10 october2022
Customer Journey	Prepare the customer journey maps to understand the user interactions & experiences with the application (entry to exit).	14 october2022
Functional Requirement	Prepare the functional requirement document.	13 october2022
Data Flow Diagrams	Draw the data flow diagrams and submit for review.	20 october2022

Sprint Delivery Plan	Allocate time for each and every functions	04 November2022
Prepare Milestone & Activity List	Prepare the milestones & activity list of the project.	04 November2022

Project Development - Delivery of Sprint-1, 2, 3 & 4	Develop & submit the developed code by testing it.	IN PROGRESS

Product Backlog, Sprint Schedule, and Estimation :

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	3	High	Anish

Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	2	Medium	Archana

Sprint-2		USN-3	As a user, I can register for the application through mobile number	3	High	Anto maria swetha
Sprint-2		USN-4	As a user. I will receive confirmation SMS	3	High	Arul kumar
Sprint-2	Login	USN-5	As a user, Ican log intothe application by enteringlogin credentials	3	High	Anish Archana
Sprint-3	Dashboard	USN-6	As a user, I can upload my images and get my details of skin diseases	3	High	Anto maria swetha Arul kumar
Sprint-1	Logout	USN-7	As a user, I can logout successfully	2	Medi um	Archana
Sprint-	Feedback	USN-8	As a customer	2	Medi	

			care executive, I can able to interact withall the customer and get their feedback which is used to enhance the scope of the project		um	Anish
Sprint-3	Image processing, localization	USN-9	The uploaded image is preprocessed and fed into the trained YOLO model	3	High	Archana Anto maria swetha
Sprint-4	Classificati on and prediction	USN-9	The YOLO model classify and predict the type of disease and the area affected	3	High	Arul kumar Anish Anto maria swetha

Sprint	**Total Story Points**	**Duration**	**Sprint Start Date**	**Sprint End Date (Planned)**	**Story Points Completed (as on Planned End Date)**	**Sprint Release Date (Actual)**
Sprint-1	20	6 Days	29 Oct 2022	O4 Nov 2022	20	4 Nov 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022

						Archana
Sprint-4 4	Report Generation	USN- 10	ased on the prediction of skin diseases, the health care report generatedto provide feedbacks	2	Medi um	Arul kumar Anish

Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

Average Velocity = Story Points per Day Sprint Duration = Number of (Duration) days perSprint Velocity = Points per Sprint

AV = **20 /6** ≈ 4

Therefore, the AVERAGE VELOCITY IS 4 POINTS PER SPRINT

Burn down Chart:

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can beapplied to any project containing measurable progress over time.