AI BASED LOCALIZATION AND CLASSIFICATION OF SKIN DISEASE WITH ERYTHEMA

TEAM ID: PNT2022TMID01254

TEAM LEAD: ASHWATHI A

TEAM MEMBER1: ASHWINI S

TEAM MEMBER2: JOYCE JONAFARC L

TEAM MEMBER3: PAVITHRA R

1.INTRODUCTION:

1.1 PROJECT OVERVIEW:

Skin diseases are most commonly occurring in people of all ages and are caused by bacteria, infection or radiation. These diseases have various dangerous effects on the skin and keep on spreading over time. A patient can recover from skin diseases if it is detected and treated in the early stages and this can achieve cure ratios of over 95%. Hence, it is important to identify these diseases at their initial stage to control them from spreading. Skin diseases are primarily diagnosed visually, beginning with an initial clinical screening and followed potentially by dermoscopic analysis. Such a system is often prone to errors. The main idea of this project is to improve the accuracy of diagnostic systems by using Image Processing and classification techniques. In the proposed system, an image captured on camera is taken as input. This image will be pre-processed in order to make it suitable for segmentation by using Contrast Enhancement and Grayscale Conversion. Global Thresholding technique is used to segment the preprocessed image through which the actual affected region is obtained. Texture features, such as Energy, Entropy, Contrast, IDM, are extracted from the segmented image using Grey Level Co-occurrence Matrix. Image Quality Assessment features such as MSE and PSNR are extracted. The extracted texture features will be used to detect the presence of skin disease and classify the disease as melanoma, leprosy or eczema, if present, using the Decision tree technique.

1.2 PURPOSE:

Skin disease is the most common disease in the world. The diagnosis of the skin disease requires a high level of expertise and accuracy for dermatologist, so computer aided skin disease diagnosis model is proposed to provide more objective and reliable solution. Many researches were done to help detect skin diseases like skin cancer and tumor skin. But the accurate recognition of the disease is extremely challenging due to the following reasons: low contrast between lesions and skin, visual similarity between Disease and non-Disease area, etc. This paper aims to detect skin disease from the skin image and to analyze this image by applying filter to remove noise or unwanted things, convert the image to grey to help in the processing and get the useful information. This help to give evidence for any type of skin disease and illustrate emergency orientation. Analysis result of this study can support doctor to help in initial diagnoses and to know the type of disease. That is compatible with skin and to avoid side effects. Skin disease makes as great an impact as other serious medical conditions when assessed by effects on health-related quality of life. .A detailed skin analysis will allow you to choose the most appropriate treatment and products to improve and maintain the condition of the skin. The objective of this process is to increase accuracy of skin disease detection. Three important features in image classification are texture, color, shape, and combination of these. In this work, color and texture features are used to classify the skin disease. Normal skin color is different from the skin with disease. Smoothness, coarseness, and regularity is effectively identified using texture features in the images.

2.LITERATURE SURVEY:

2.1 EXISTING PROBLEM:

Some of the most common skin diseases include Acne,blocked skin follicles that lead to oil,bacteria and dead skin buildup in your pores.Alopecia areata,losing your hair in small patches.Atopic dermatitis,dry,itchy skin leads to swelling,cracking or scaliness.

2.2 REFERENCES:

[1] K.INDUPRIYA, Dr. G. P. RAMESH KUMAR, “A SURVEY ON SKIN TEXTURE ANALYSIS FOR MEDICAL DIAGNOSIS USING IMAGE PROCESSING TECHNIQUES”, K. INDUPRIYA et al. Volume 3 Issue 5, 2015

[2] Md Nafiul Alam, Tamanna Tabassum Khan Munia, Kouhyar Tavakolian, Vasefi, Nick MacKinnon, Reza Fazel-Rezai, “Automatic Detection and Severity Measurement of Eczema
Using Image Processing”, IEEE, 2016.

[3] Manish Kumar and Rajiv Kumar, “AN INTELLIGENT SYSTEM TO DIAGNOSIS THE SKIN DISEASE”, VOL. 11, NO. 19, OCTOBER 2017.

[4] M. T. Habib, A. Majumder, A. Z. M. Jakaria, M. Akter, Md. S. Uddin, F. Ahmed “Machine vision-based papaya disease recognition,” Journal of King Saud University - Computer and Information Sciences (2018).

[5]L. Tizek, M.C. Schielein, F. Seifert, T. Biedermann, A. Bohner, A. Zink, “ Skin diseases are more common than we think: screening results of an unreferred population at the Munich Oktoberfest,” Journal of the European Academy of Dermatology and Venereology, 2019;

[6] C. Sagar and L. M. Saini. "Colour channel based segmentation of skin lesion from clinical images for the detection of melanoma," In 2019 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES), pp. 1-5. IEEE, 2019.

[7] Sheha Mariam A., Mabrouk Mai S, Sharawy Amr.(2020). Automatic Detection of Melanoma Skin Cancer. International Journal of Computer Applications (0975 –8887), 42(20),22-26

[8] N. S. A. ALEnezi, "A method of skin disease detection using image processing and machine learning," In 2019 16th International Learning & Technology Conference.

[9] Kumar, V., Kumar, S., & Saboo, V. (2016) "Dermatological Disease Detection Using Image Processing and Machine Learning." IEEE.

[10] A. Adeel et al., "Diagnosis and Recognition of Grape Leaf Diseases: An automated system based on a Novel Saliency approach and Canonical Correlation Analysis based multiple features fusion," Sustainable Computing: Informatics and Systems (2019).

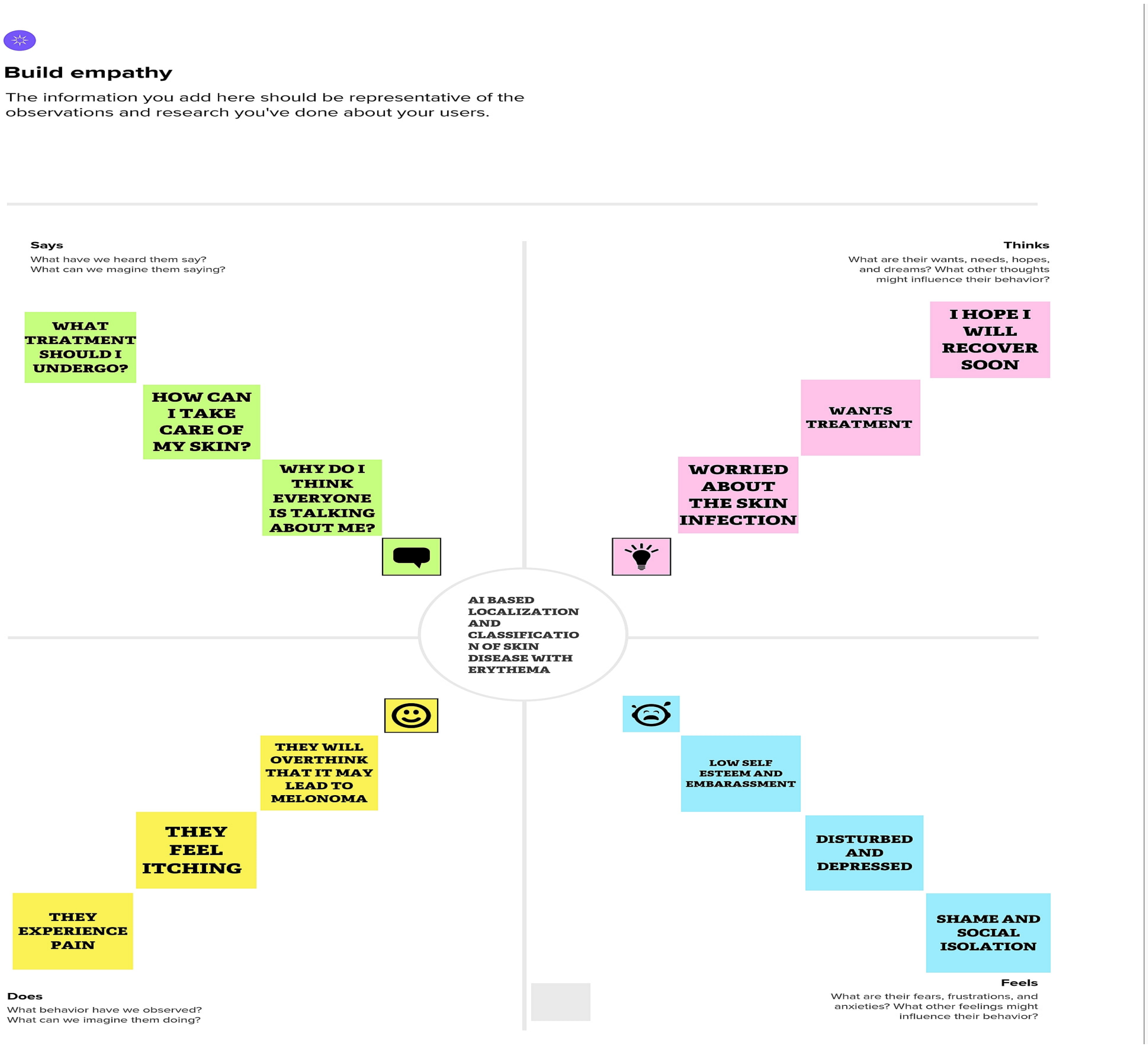
[11] Traditional Techniques for Skin Disease Image Classification. Authors : Tanvi Goswami; Vipul K. Dabhi; Harshadkumar B. Prajapati.Year:2020

2.3 PROBLEM STATEMENT DEFINITION:

In general, the advantage of detecting skin disease is that it can help doctors perform tedious repetitive tasks. For example, if sufficient blood is scanned, an powered microscope can detect low-density infections in micrographs of standard, field-prepared thick blood films, which is considered to be time-consuming, difficult, and tedious owing to the low density and small parasite size and abundance of similar non-parasite objects .The requirement for staff training and purchase of expensive equipment for creating dermoscopic images can be replaced by software using CNNs. In the future, the clinical application of Artificial intelligence for the diagnosis ofother diseases can be investigated. Transfer learning could be useful in developing CNN models for relatively rare diseases. Models could also evolve such that they require fewer preprocessing steps. In addition to these topics, a deeper understanding of the reconstruction kernel or image thickness could lead to improved AI model performance. Positive effects should continue to grow owing to the emergence of higher precision scanners and image reconstruction techniques. However, we must realize that AI has the ability to defeat humans in several specific fields.

3.IDEATION AND PROPOSED SOLUTION:

3.1 EMPATHY MAP CANVAS:



Build empathy

The information you add here should be representative of the observations and research you've done about your users.

Says

What have we heard them say?
What can we magine them saying?

WHAT TREATMENT SHOULD I UNDERGO?

HOW CAN I TAKE CARE OF MY SKIN?

WHY DO I THINK EVERYONE IS TALKING ABOUT ME?

Thinks

What are their wants, needs, hopes, and dreams? What other thoughts might influence their behavior?

I HOPE I WILL RECOVER SOON

WANTS TREATMENT

WORRIED ABOUT THE SKIN INFECTION

AI BASED LOCALIZATION AND CLASSIFICATIO N OF SKIN DISEASE WITH ERYTHEMA

THEY WILL OVERTHINK THAT IT MAY LEAD TO MELONOMA

THEY FEEL ITCHING

THEY EXPERIENCE PAIN

Does

What behavior have we observed?
What can we imagine them doing?

LOW SELF ESTEEM AND EMBARASSMENT

DISTURBED AND DEPRESSED

SHAME AND SOCIAL ISOLATION

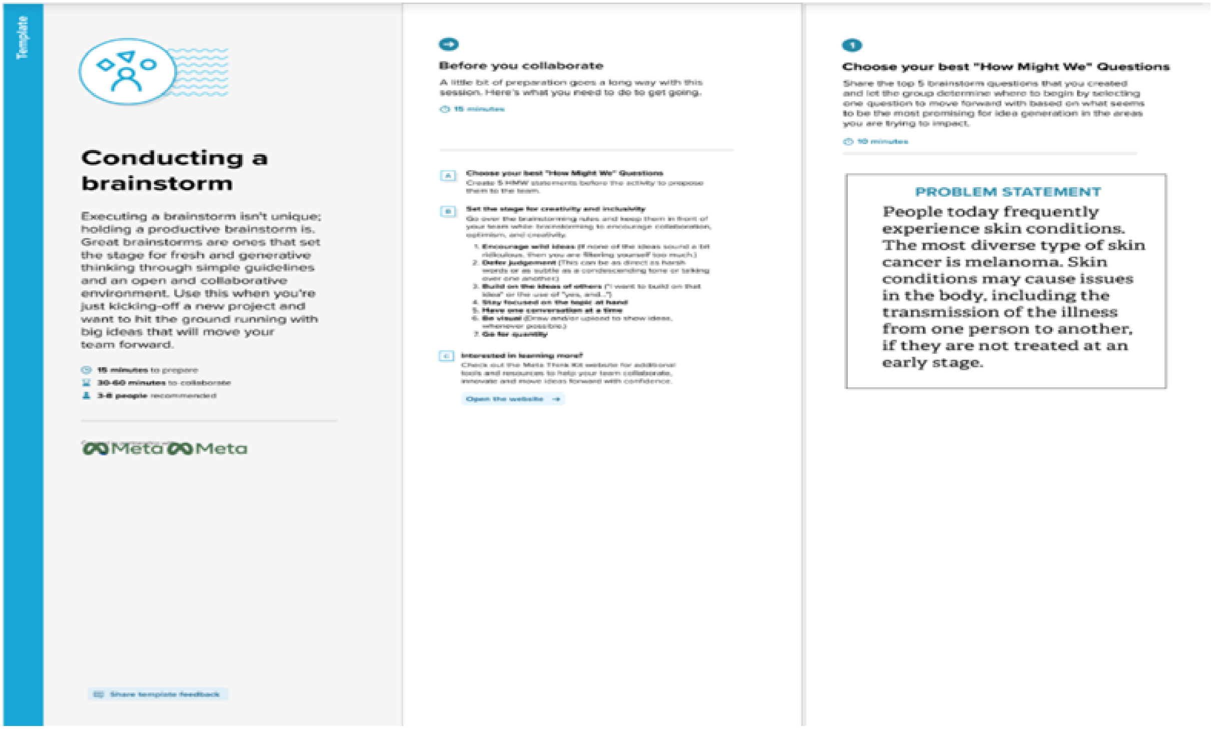
Feels

What are their fears, frustrations, and anxieties? What other feelings might influence their behavior?

3.2 IDEATION AND BRAINSTORMING:

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	Skin diseased person	Get over from the diseased skin	The spread is increasing day by day	I do not know where to consult	insecure
PS-2	The person having skin allergies	Recover my skin from the disease	It is getting bad on my skin	I could not find any perfect solution anywhere to start my treatment	afraid

Step-1: Team Gathering, Collaboration and Select the Problem Statement



Conducting a brainstorm

Executing a brainstorm isn't unique; holding a productive brainstorm is. Great brainstorms are ones that set the stage for fresh and generative thinking through simple guidelines and an open and collaborative environment. Use this when you're just kicking-off a new project and want to hit the ground running with big ideas that will move your team forward.

- **15 minutes** to prepare
- **30-60 minutes** to collaborate
- **3-8 people** recommended

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

15 minutes

A **Choose your best "How Might We" Questions**
Create 5 HMW statements before the activity to propose them to the team.

B **Set the stage for creativity and inclusivity**
Go over the brainstorming rules and keep them in front of your team while brainstorming to encourage collaboration, optimism, and creativity.

1. **Encourage wild ideas** (If none of the ideas sound a bit ridiculous, then you are filtering yourself too much.)
2. **Defer judgement** (This can be as direct as harsh words or as subtle as a condescending tone or talking over one another.)
3. **Build on the ideas of others** ("I want to build on that idea" or the use of "yes, and...")
4. **Stay focused on the topic at hand**
5. **Have one conversation at a time**
6. **Be visual** (Draw and/or upload to show ideas, whenever possible.)
7. **Go for quantity**

C **Interested in learning more?**
Check out the Meta Think Kit website for additional tools and resources to help your team collaborate, innovate and move ideas forward with confidence.

1 **Choose your best "How Might We" Questions**

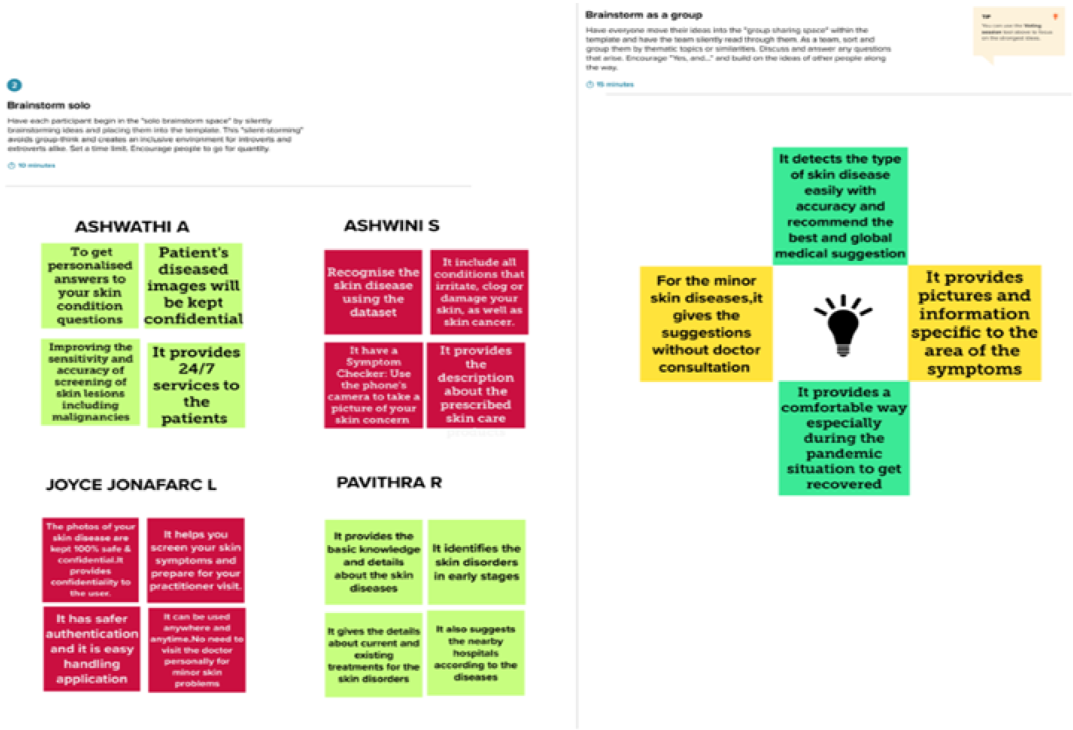
Share the top 5 brainstorm questions that you created and let the group determine where to begin by selecting one question to move forward with based on what seems to be the most promising for idea generation in the areas you are trying to impact.

10 minutes

PROBLEM STATEMENT

People today frequently experience skin conditions. The most diverse type of skin cancer is melanoma. Skin conditions may cause issues in the body, including the transmission of the illness from one person to another, if they are not treated at an early stage.

Step-2: Brainstorm, Idea Listing and Grouping



2

Brainstorm solo

Have each participant begin in the "solo brainstorm space" by silently brainstorming ideas and placing them into the template. This "silent-storming" avoids group-think and creates an inclusive environment for introverts and extroverts alike. Set a time limit. Encourage people to go for quantity.

10 minutes

ASHWATHI A

- To get personalised answers to your skin condition questions
- Patient's diseased images will be kept confidential
- Improving the sensitivity and accuracy of screening of skin lesions including malignancies
- It provides 24/7 services to the patients

ASHWINI S

- Recognise the skin disease using the dataset
- It include all conditions that irritate, clog or damage your skin, as well as skin cancer.
- It have a Symptom Checker: Use the phone's camera to take a picture of your skin concern
- It provides the description about the prescribed skin care products

JOYCE JONAFARC L

- The photos of your skin disease are kept 100% safe & confidential.It provides confidentiality to the user.
- It helps you screen your skin symptoms and prepare for your practitioner visit.
- It has safer authentication and it is easy handling application
- It can be used anywhere and anytime.No need to visit the doctor personally for minor skin problems

PAVITHRA R

- It provides the basic knowledge and details about the skin diseases
- It identifies the skin disorders in early stages
- It gives the details about current and existing treatments for the skin disorders
- It also suggests the nearby hospitals according to the diseases

Brainstorm as a group

Have everyone move their ideas into the "group sharing space" within the template and have the team silently read through them. As a team, sort and group them by thematic topics or similarities. Discuss and answer any questions that arise. Encourage "Yes, and..." and build on the ideas of other people along the way.

15 minutes

TIP

You can use the Voting session tool above to focus on the strongest ideas.

- It detects the type of skin disease easily with accuracy and recommend the best and global medical suggestion
- For the minor skin diseases,it gives the suggestions without doctor consultation
- It provides pictures and information specific to the area of the symptoms
- It provides a comfortable way especially during the pandemic situation to get recovered

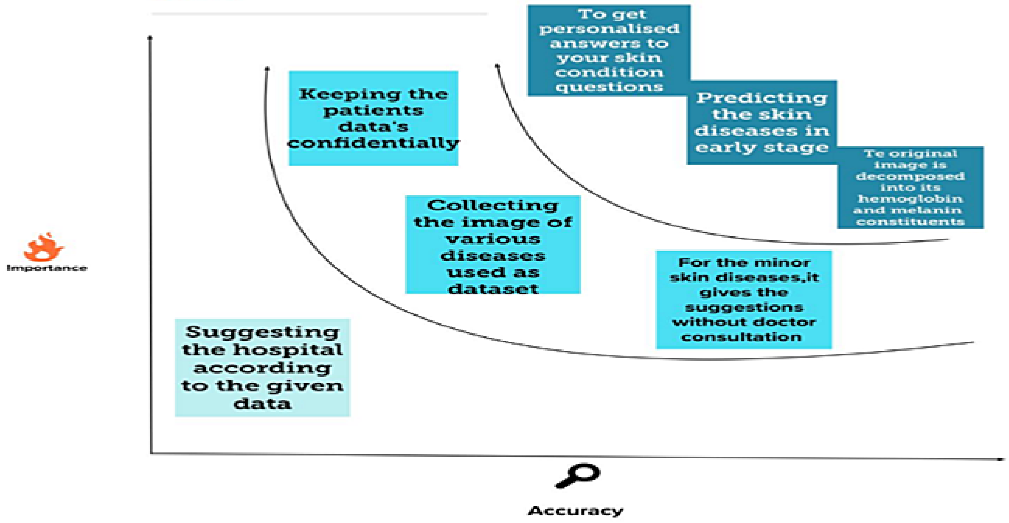
Step-3: Idea Prioritization

4

Idea Prioritization

Give each person two icons to vote which idea should your team focus on.

5 minutes

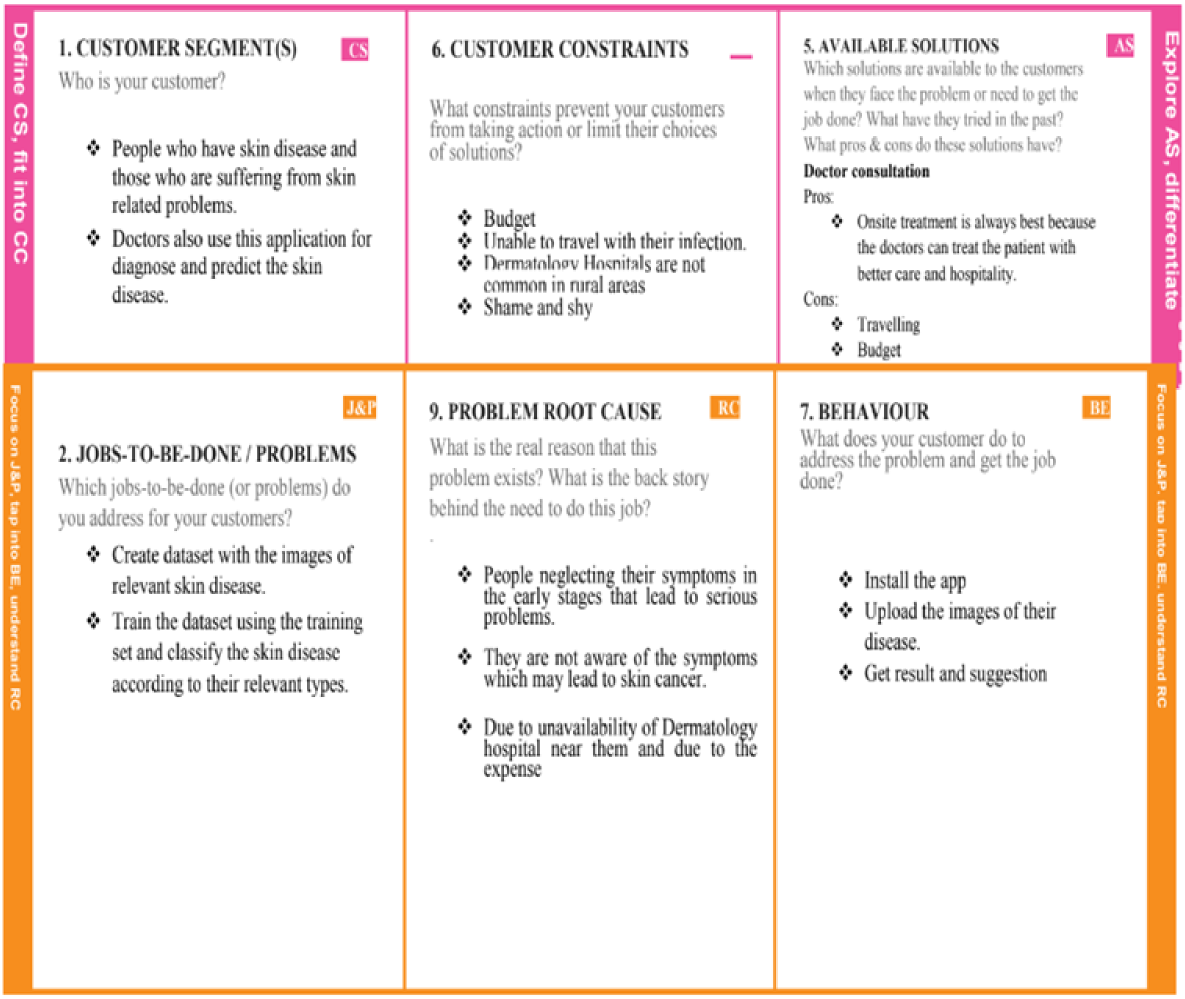

To get personalised answers to your skin condition questions
Keeping the patients data's confidentially
Predicting the skin diseases in early stage
Te original image is decomposed into its hemoglobin and melanin constituents
Collecting the image of various diseases used as dataset
For the minor skin diseases,it gives the suggestions without doctor consultation
Suggesting the hospital according to the given data
Importance
Accuracy

3.3 PROPOSED SOLUTION:

S .No.	Parameter	Description
1.	Problem Statement (Problem to besolved)	People today frequently experience skin conditions. The most diversetype of skincancer is melanoma. Skin conditions may cause issues in the body including the transmission of the illness from one person to another. If they are not treated at an earlystage.
2.	Idea / Solution description	We are developing a model that is used for the early detection and prevention of psoriasis and skin cancerin order to solve the aforementioned issue. In general, the diagnosis of skin diseases depends on many traits like colour, form,texture, etc. Here,a person can take skin-related pictures, whichwill subsequently be sent to a trained model. The model examines the image to determine whether or not the subject has a skin condition.
3.	Novelty / Uniqueness	● Higher Accuracy. ● Advanced prescription. ● Day to day Observation.
4.	Social Impact/ Customer Satisfaction	It predicts the type of skin disorder at an early stage and prescribes the needed treatment efficiently. It is an open application for everyone to make use of it.
5.	Business Model (Revenue Model)	This Application focuson curing the disease ofthe personhaving allergies. It acts like a platform to detect their illness and supporting them through theirhard times.

6.	Scalability of the Solution	It is scalable as it has a lot of features involved. The features includes image processing, disease classification, treatment prescription, severity, description of the disease, duration depends on the severity, Notifying nearby healthcare centre.

3.4 PROBLEM SOLUTION FIT:


Define CS, fit into CC
Explore AS, differentiate
Focus on J&P, tap into BE, understand RC
Focus on J&P, tap into BE, understand RC
CS
AS
J&P
RC
BE

1. CUSTOMER SEGMENT(S)

Who is your customer?

- People who have skin disease and those who are suffering from skin related problems.
- Doctors also use this application for diagnose and predict the skin disease.

6. CUSTOMER CONSTRAINTS

What constraints prevent your customers from taking action or limit their choices of solutions?

- Budget
- Unable to travel with their infection.
- Dermatology Hospitals are not common in rural areas
- Shame and shy

5. AVAILABLE SOLUTIONS

Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have?

Doctor consultation

Pros:

- Onsite treatment is always best because the doctors can treat the patient with better care and hospitality.

Cons:

- Travelling
- Budget

2. JOBS-TO-BE-DONE / PROBLEMS

Which jobs-to-be-done (or problems) do you address for your customers?

- Create dataset with the images of relevant skin disease.
- Train the dataset using the training set and classify the skin disease according to their relevant types.

9. PROBLEM ROOT CAUSE

What is the real reason that this problem exists? What is the back story behind the need to do this job?

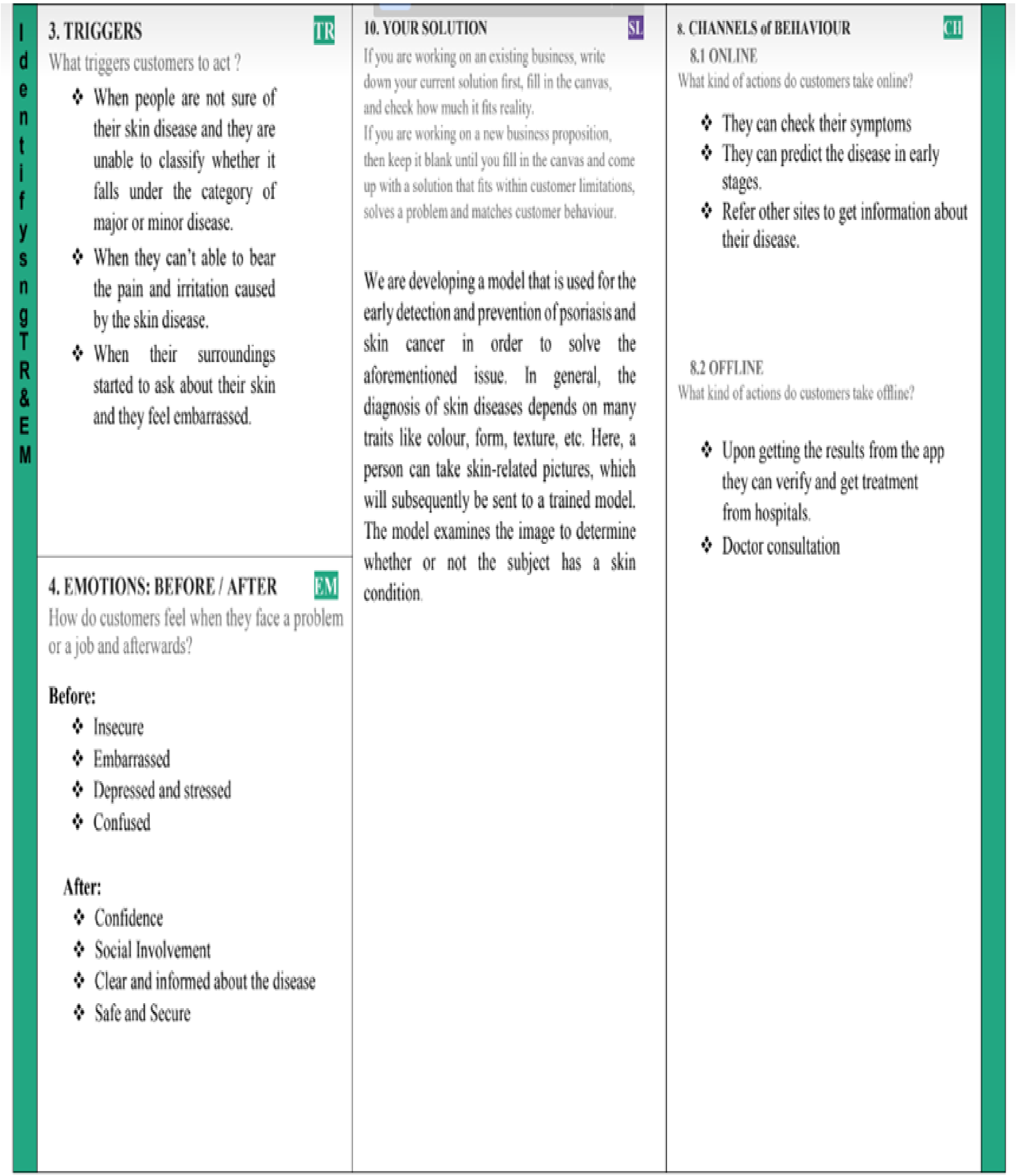
.

- People neglecting their symptoms in the early stages that lead to serious problems.
- They are not aware of the symptoms which may lead to skin cancer.
- Due to unavailability of Dermatology hospital near them and due to the expense

7. BEHAVIOUR

What does your customer do to address the problem and get the job done?

- Install the app
- Upload the images of their disease.
- Get result and suggestion



Identifysng TR&EM

3. TRIGGERS

TR

What triggers customers to act ?

- ❖ When people are not sure of their skin disease and they are unable to classify whether it falls under the category of major or minor disease.
- ❖ When they can't able to bear the pain and irritation caused by the skin disease.
- ❖ When their surroundings started to ask about their skin and they feel embarrassed.

4. EMOTIONS: BEFORE / AFTER

EM

How do customers feel when they face a problem or a job and afterwards?

Before:

- ❖ Insecure
- ❖ Embarrassed
- ❖ Depressed and stressed
- ❖ Confused

After:

- ❖ Confidence
- ❖ Social Involvement
- ❖ Clear and informed about the disease
- ❖ Safe and Secure

10. YOUR SOLUTION

SL

If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality.

If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour.

We are developing a model that is used for the early detection and prevention of psoriasis and skin cancer in order to solve the aforementioned issue. In general, the diagnosis of skin diseases depends on many traits like colour, form, texture, etc. Here, a person can take skin-related pictures, which will subsequently be sent to a trained model. The model examines the image to determine whether or not the subject has a skin condition.

8. CHANNELS of BEHAVIOUR

CH

8.1 ONLINE

What kind of actions do customers take online?

- ❖ They can check their symptoms
- ❖ They can predict the disease in early stages.
- ❖ Refer other sites to get information about their disease.

8.2 OFFLINE

What kind of actions do customers take offline?

- ❖ Upon getting the results from the app they can verify and get treatment from hospitals.
- ❖ Doctor consultation

4.REQUIREMENT ANALYSIS:

4.1 Functional Requirements:

Followingare the functional requirements of the proposedsolution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Get User Input	Upload image as jpeg Upload image as png
FR-4	Image Pre-processing	Image of the skin disease is pre-processed to analyze image more efficiently and quickly
FR-5	Feature Extraction	After image pre-processing , Feature extraction is done to achievebetter classification of the skindisease using erythema.
FR-6	Skin disease Type Prediction with erythema	Afterfeature extraction , According to the given image the type of skindisease is predicted.

4.2 Non-functional Requirements:

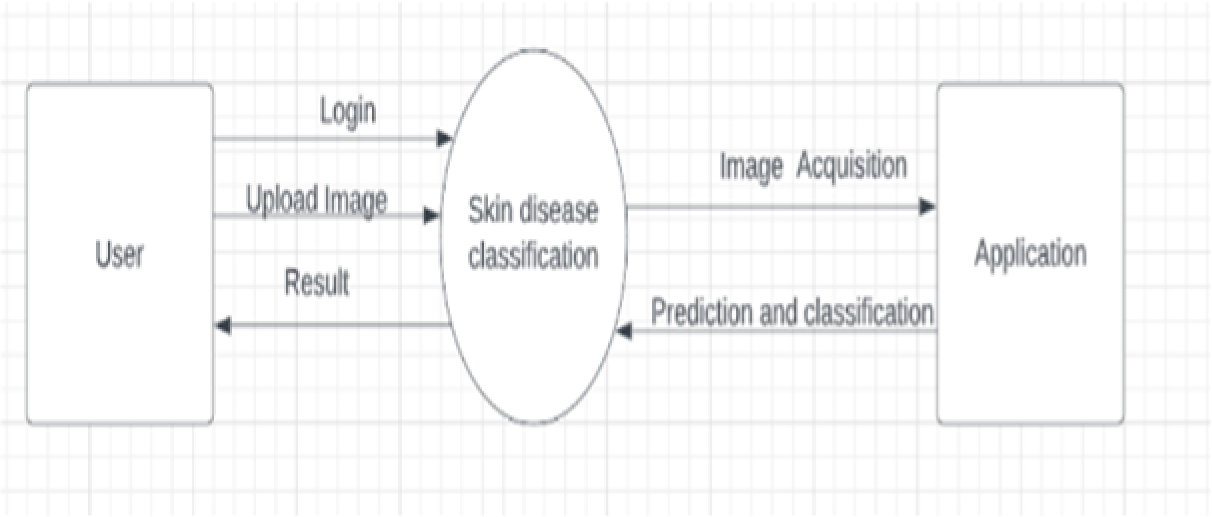
Followingare the non-functional requirements of the proposedsolution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The application shouldhave user friendly Graphics User Interface
NFR-2	**Security**	Only authorized users can view the data so that user data is secured
NFR-3	**Reliability**	User data shouldnot be shared to any third-party applications.
NFR-4	**Performance**	The application shoulddetect Erythema as fast as possible withmore accuracy
NFR-5	**Availability**	The software should be available for multiple user access simultaneously.
NFR-6	**Scalability**	The application shouldbe scalable to upload multiple imagesat a time for detection.

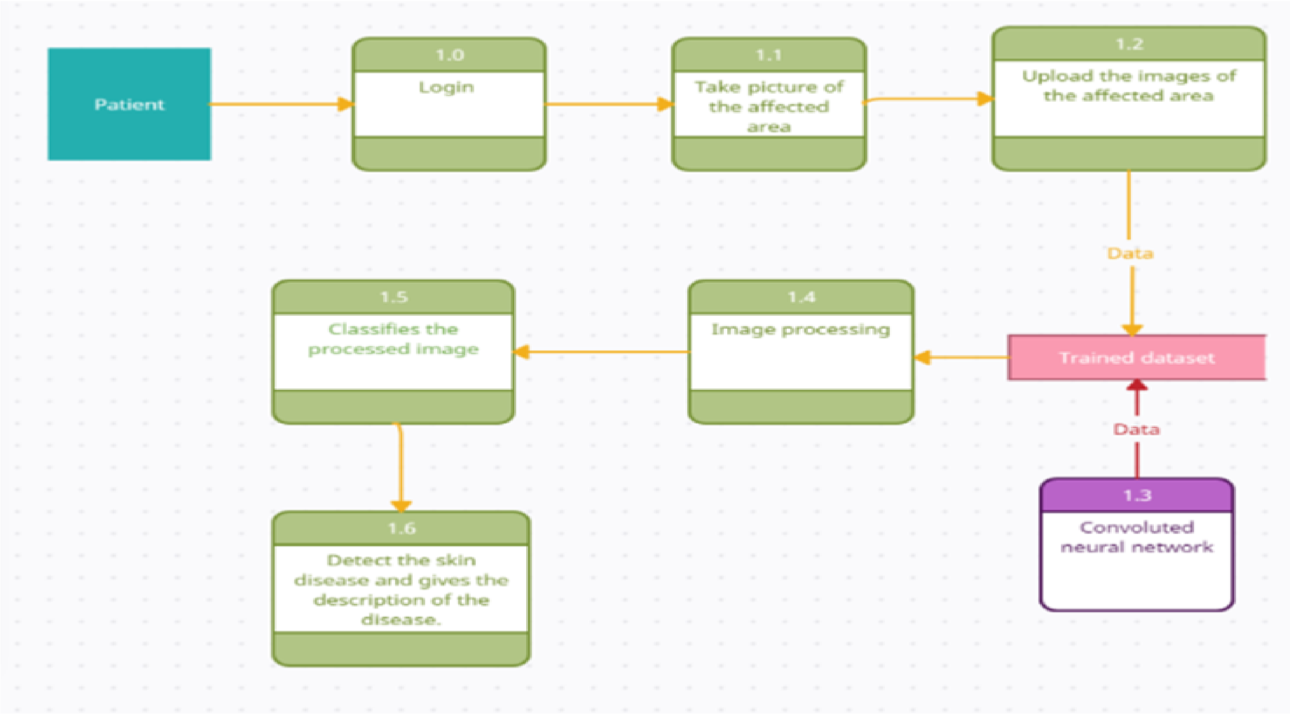
5.PROJECT DESIGN:

5.1 DATA FLOW DIAGRAMS:

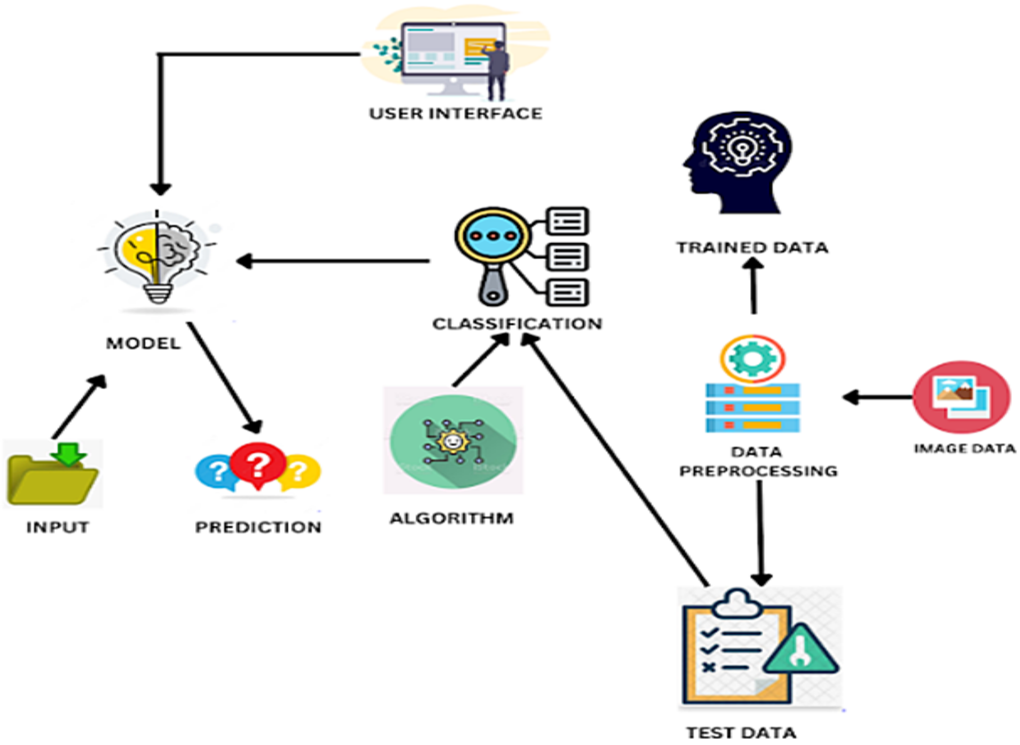
LEVEL 0:



LEVEL 1:


Patient
1.0
Login
1.1
Take picture of the affected area
1.2
Upload the images of the affected area
Data
Trained dataset
Data
1.3
Convoluted neural network
1.4
Image processing
1.5
Classifies the processed image
1.6
Detect the skin disease and gives the description of the disease.

5.2 SOLUTION AND TECHNICAL ARCHITECTURE:



5.3 USER STORIES:

User Type	Functional Requirement (Epic)	User Story Number	User Story/task	Acceptance criteria	Priority	Release
Customer (Mobile user)		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 oncel have registered for the application	I can receive confirmation email & clickconfirm	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
	Login	USN-4	As a user, I can register for the application through Gmail	I can register throughGmail.	Medium	Sprint-1
		USN-5	As a user, I can log into the application by entering email & password	I can alsoreceive logout credential.	High	Sprint-1
	Interface	USN-6	As a user,the interface should be easy toaccess.	I can receive login credentia	Medium	Sprint-2
Patient (Web user)	Dashboard	USN-7	As a user I can specify the information(Skin color,skin tone, skin texture,screening etc)	I can ableto know about how depth the disease is.	High	Sprint-1
Patient (input)	View manner	USN-8	As a user,I can view disease detailsin visual representation (images).	I can easily understand by using imagesvisually.	High	Sprint-1
	Color visibility	USN-9	As a user,I can ableto see the skin colordue to infected area	I can easily know about thecondition of skin color.	High	Sprint-2
	Knowledge	USN-10	As a user,I can able to know aboutthe disease detailsin early stage	I can easily know whether I have diseaseor not	High	Sprint-1

Administrator	Knowledge	USN-11	An administrator who Is handling the website should updateand take careof the application	Admin should monitorthe records properly.	Medium	Sprint-2

6.PROJECT PLANNING AND SCHEDULING:

6.1 SPRINT PLANNING AND ESTIMATION:

Sprint	Functional Requirement (Epic)	User Story Number	User Story /Task	Story Points	Priority	Team Members
Sprint-1	Create Dataset	USN-1	Create the dataset with50 images perskin disease	3	High	Ashwini S Ashwathi A
Sprint-2	Annotate images	USN-2	Annotate images usingMicrosoft VOTT into four phases	3	High	Pavithra R JoyceJonafarc L
Sprint-3	Training Yolo and Build pythoncode	USN-3	Download and Convert pre-trained weights. Train Yolov3 detector and build the source code	3	High	Ashwini S Ashwathi A

Sprint-4	Cloudant DB	USN-4	Create cloud account, create serviceinstance, launch cloudant DB and create the database	3	High	Pavithra R Joyce Jonafarc L
Sprint-4	Registration	USN-5	As a user, I can register for theapplication by entering my email,password, andconfirming my password.	3	High	Joyce Jonafarc L Pavithra R
Sprint-4		USN-6	As a user, I will receive confirmation email once I have registered for the application	2	Medium	Joyce Jonafarc L Pavithra R
Sprint-4		USN-7	As a user, I canregisterfor the application through mobile number	3	High	Joyce Jonafarc L Pavithra R
Sprint-4		USN-8	As a user. I will Receiveconfirmation SMS	3	High	Joyce Jonafarc L Pavithra R

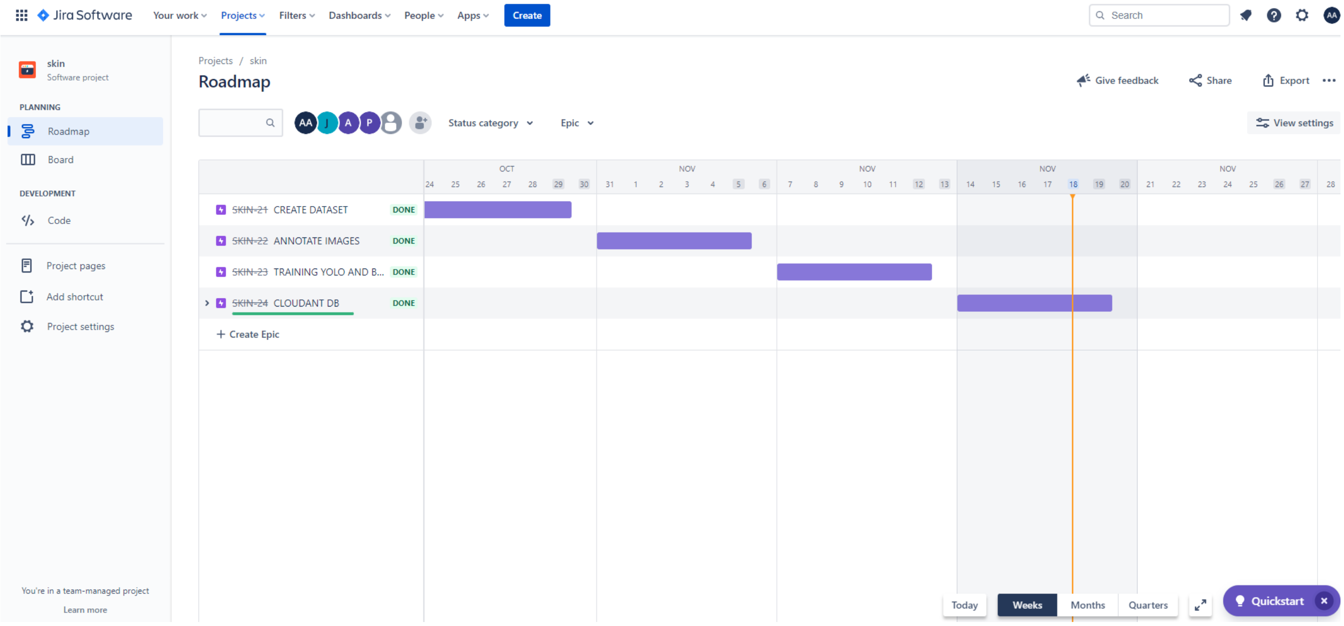
Sprint-4	Login	USN-9	As a user, I can log into the applicationby entering login credentials	3	High	Joyce Jonafarc L Pavithra R
Sprint-4	Dashboard	USN-10	As a user, I can upload myimages and get my details of skin diseases	3	High	Joyce Jonafarc L Pavithra R
Sprint -4	Logout	USN-11	As a user, I can logout successfully	2	Medium	Joyce Jonafarc L Pavithra R

6.2 SPRINT DELIVERY SCHEDULE:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned EndDate)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
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

6.3 REPORTS FROM JIRA:

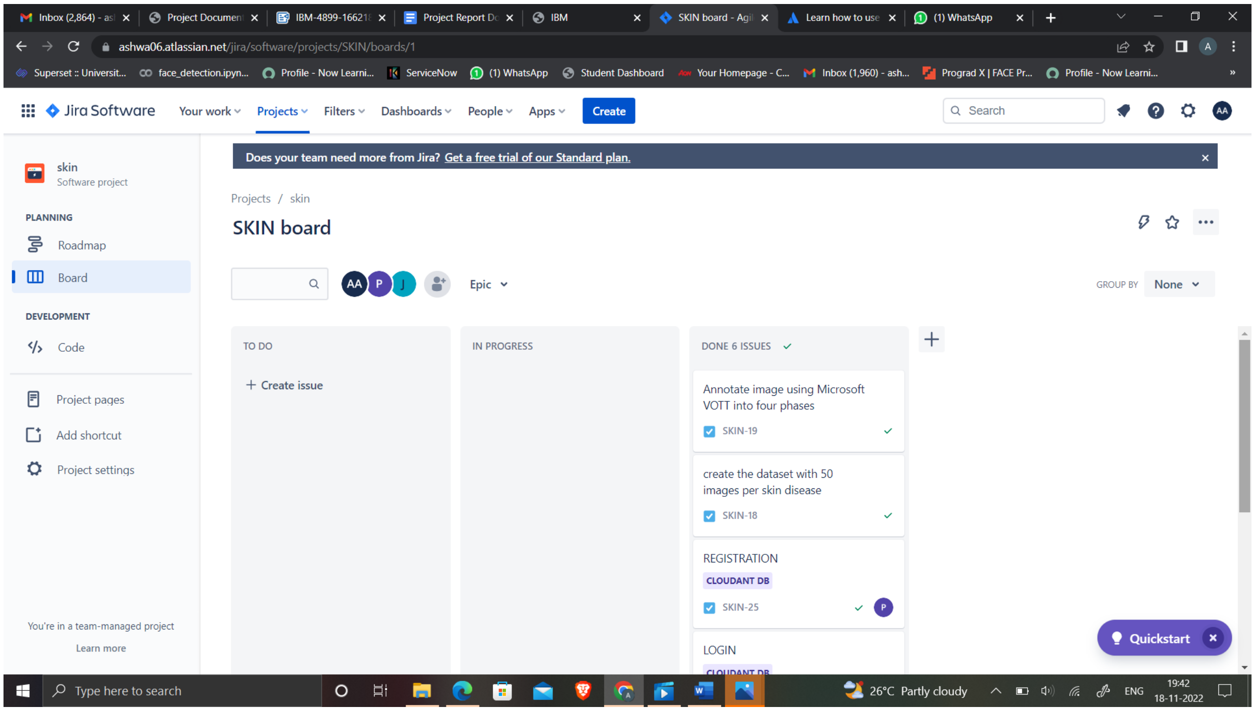
ROADMAP:



Roadmap

~~SKIN-21~~	CREATE DATASET	DONE
~~SKIN-22~~	ANNOTATE IMAGES	DONE
~~SKIN-23~~	TRAINING YOLO AND B...	DONE
~~SKIN-24~~	CLOUDANT DB	DONE

BOARD:



SKIN board

TO DO

IN PROGRESS

DONE 6 ISSUES

- Annotate image using Microsoft VOTT into four phases — SKIN-19
- create the dataset with 50 images per skin disease — SKIN-18
- REGISTRATION — CLOUDANT DB — SKIN-25
- LOGIN — CLOUDANT DB

7.CODING AND SOLUTIONING:

7.1 FEATURE 1:

Annotate Images Our detector needs some high-quality training examples before it can start learning. The images in our training folder are manually labelled using Microsoft's Visual Object Tagging Tool (VoTT). At least 100 images should be annotated for each category to get respectable results. The VoTT csv formatted annotation data is converted to YOLOv3 format by Convert_to_YOLO_format.py file.

Code:

```
from PIL import Image
from is import path, makedirs
import os
 import re
 import pandas as pd
 import sys
import argparse
def get_parent_dir(n=1):
    """ returns the n-the parent directory 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()
```

7.2 FEATURE 2:

Training Yolo:

To prepare for the training process, convert the YOLOv3 model to the Keras format.
The YOLOv3 Detector can then be trained by Train_YOLO.py file.

Code:

```
import os
import sys
import argparse
import warnings
def get_parent_dir(n=1):
    """ returns the n-th parent directory 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(0), "src")
sys.path.append(src_path)
utils_path = os.path.join(get_parent_dir(1), "Utils")
sys.path.append(utils_path)
import numpy as np
import keras.backend as K
from keras.layers import Input, Lambda
from keras.models import Model
from keras.optimizers import Adam
from keras.callbacks import (
    TensorBoard,
    ModelCheckpoint,
    ReduceLROnPlateau,
    EarlyStopping,
)
from keras_yolo3.yolo3.model import (
    preprocess_true_boxes,
    yolo_body,
    tiny_yolo_body,
    yolo_loss,
)
from keras_yolo3.yolo3.utils import get_random_data
from PIL import Image
from time import time
```

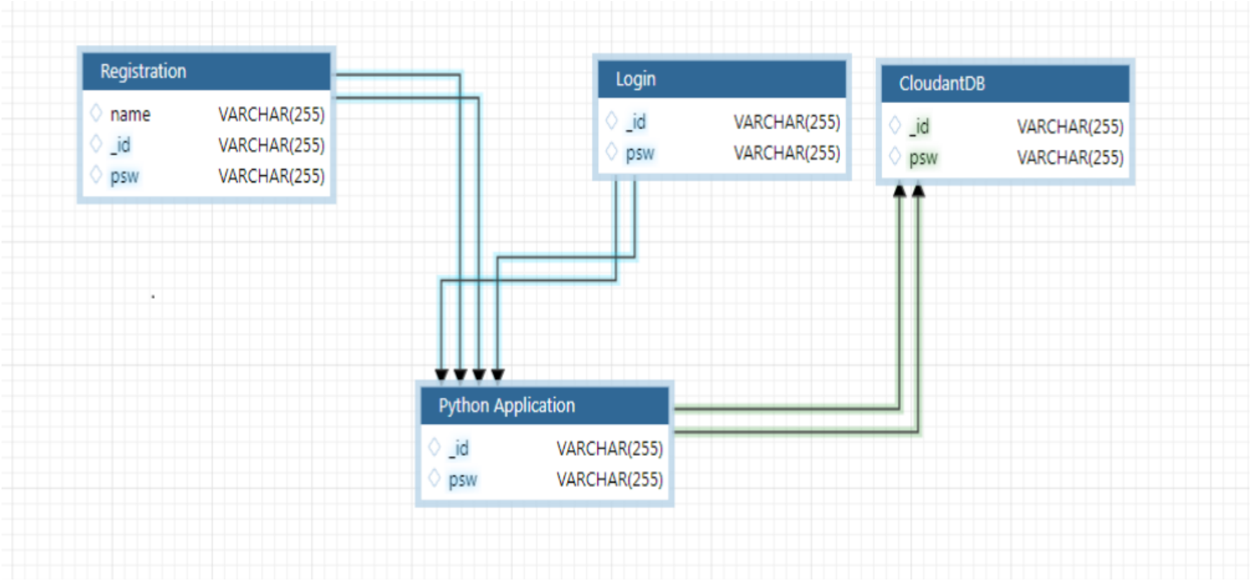
```
import tensorflow.compat.v1 as tf
import pickle
from Train_Utils import (
    get_classes,
    get_anchors,
create_model,
    create_tiny_model,
    data_generator,
    data_generator_wrapper,
    ChangeToOtherMachine,
)
keras_path = os.path.join(src_path, "keras_yolo3")
Data_Folder = os.path.join(get_parent_dir(1), "Data")
Image_Folder = os.path.join(Data_Folder, "Source_Images", "Training_Images")
VoTT_Folder = os.path.join(Image_Folder, "vott-csv-export")
YOLO_filename = os.path.join(VoTT_Folder, "data_train.txt")
Model_Folder = os.path.join(Data_Folder, "Model_Weights")
YOLO_classname = os.path.join(Model_Folder, "data_classes.txt")
log_dir = Model_Folder
anchors_path = os.path.join(keras_path, "model_data", "yolo_anchors.txt")
weights_path = os.path.join(keras_path, "yolo.h5")
FLAGS = None
if __name__ == "__main__":
    # Delete all default flags
    parser = argparse.ArgumentParser(argument_default=argparse.SUPPRESS)
    """
    Command line options
    """
    parser.add_argument(
        "--annotation_file",
        type=str,
        default=YOLO_filename,
        help="Path to annotation file for Yolo. Default is " + YOLO_filename,
    )
    parser.add_argument(
        "--classes_file",
        type=str,
        default=YOLO_classname,
        help="Path to YOLO classnames. Default is " + YOLO_classname,
    )
    parser.add_argument(
        "--log_dir",
        type=str,
        default=log_dir,
        help="Folder to save training logs and trained weights to. Default is "
        + log_dir,
    )
    parser.add_argument(
        "--anchors_path",
        type=str,
        default=anchors_path,
        help="Path to YOLO anchors. Default is " + anchors_path,
    )
    parser.add_argument(
        "--weights_path",
        type=str,
        default=weights_path,
        help="Path to pre-trained YOLO weights. Default is " + weights_path,
    )
   parser.add_argument(
        "--val_split",
        type=float,
        default=0.1,
        help="Percentage of training set to be used for validation. Default is 10%.",
    )
    parser.add_argument(
        "--is_tiny",
        default=False,
        action="store_true",
        help="Use the tiny Yolo version for better performance and less accuracy.
Default is False.",
    )
    parser.add_argument(
        "--random_seed",
        type=float,
        default=None,
        help="Random seed value to make script deterministic. Default is 'None', i.e.
non-deterministic.",
    )
    parser.add_argument(
        "--epochs",
        type=float,
        default=51,
        help="Number of epochs for training last layers and number of epochs for finetuning layers.
Default is 51.",
    )
    parser.add_argument(
        "--warnings",
        default=False,
        action="store_true",
        help="Display warning messages. Default is False.",
    )
    FLAGS = parser.parse_args()

    if not FLAGS.warnings:
        tf.logging.set_verbosity(tf.logging.ERROR)
        os.environ['TF_CPP_MIN_LOG_LEVEL']='3'
        warnings.filterwarnings("ignore")

    np.random.seed(FLAGS.random_seed)
    log_dir = FLAGS.log_dir
    class_names = get_classes(FLAGS.classes_file)
    num_classes = len(class_names)
    anchors = get_anchors(FLAGS.anchors_path)
    weights_path = FLAGS.weights_path
    input_shape = (416, 416)  # multiple of 32, height, width
    epoch1, epoch2 = FLAGS.epochs, FLAGS.epochs
    is_tiny_version = len(anchors) == 6  # default setting
    if FLAGS.is_tiny:
        model = create_tiny_model(
            input_shape, anchors, num_classes, freeze_body=2,
weights_path=weights_path
        )
    else:
        model = create_model(
            input_shape, anchors, num_classes, freeze_body=2,
weights_path=weights_path
        )  # make sure you know what you freeze
    log_dir_time = os.path.join(log_dir, "{}".format(int(time())))
    logging = TensorBoard(log_dir=log_dir_time)
    checkpoint = ModelCheckpoint(
        os.path.join(log_dir, "checkpoint.h5"),
        monitor="val_loss",
        save_weights_only=True,
        save_best_only=True,
        period=5,
    )
    reduce_lr = ReduceLROnPlateau(monitor="val_loss", factor=0.1, patience=3,
verbose=1)
    early_stopping = EarlyStopping(
        monitor="val_loss", min_delta=0, patience=10, verbose=1
    )
    val_split = FLAGS.val_split
    with open(FLAGS.annotation_file) as f:
        lines = f.readlines()
    # This step makes sure that the path names correspond to the local machine
    # This is important if annotation and training are done on different machines (e.g.
training on AWS)
    lines = ChangeToOtherMachine(lines, remote_machine="")
    np.random.shuffle(lines)
    num_val = int(len(lines) * val_split)
    num_train = len(lines) - num_val
    # Train with frozen layers first, to get a stable loss.
    # Adjust num epochs to your dataset. This step is enough to obtain a decent model.
    if True:
        model.compile(
            optimizer=Adam(lr=1e-3),
            loss={
                # use custom yolo_loss Lambda layer.
                "yolo_loss": lambda y_true, y_pred: y_pred
            },
        )
        batch_size = 32
        print(
            "Train on {} samples, val on {} samples, with batch size {}.".format(
                num_train, num_val, batch_size
            )
        )
        history = model.fit_generator(
            data_generator_wrapper(
                lines[:num_train], batch_size, input_shape, anchors, num_classes
            ),
            steps_per_epoch=max(1, num_train // batch_size),
            validation_data=data_generator_wrapper(
                lines[num_train:], batch_size, input_shape, anchors, num_classes
            ),
            validation_steps=max(1, num_val // batch_size),
            epochs=epoch1,
            initial_epoch=0,
            callbacks=[logging, checkpoint],
        )
        model.save_weights(os.path.join(log_dir, "trained_weights_stage_1.h5"))
        step1_train_loss = history.history["loss"]
        file = open(os.path.join(log_dir_time, "step1_loss.npy"), "w")
        with open(os.path.join(log_dir_time, "step1_loss.npy"), "w") as f:
            for item in step1_train_loss:
                f.write("%s\n" % item)
        file.close()
        step1_val_loss = np.array(history.history["val_loss"])
        file = open(os.path.join(log_dir_time, "step1_val_loss.npy"), "w")
        with open(os.path.join(log_dir_time, "step1_val_loss.npy"), "w") as f:
            for item in step1_val_loss:
                f.write("%s\n" % item)
        file.close()
    # Unfreeze and continue training, to fine-tune.
    # Train longer if the result is unsatisfactory.
    if True:
        for i in range(len(model.layers)):
            model.layers[i].trainable = True
        model.compile(
            optimizer=Adam(lr=1e-4), loss={"yolo_loss": lambda y_true, y_pred: y_pred}
        )  # recompile to apply the change
        print("Unfreeze all layers.")
        batch_size = (
            4  # note that more GPU memory is required after unfreezing the body
        )
        print(
            "Train on {} samples, val on {} samples, with batch size {}.".format(
                num_train, num_val, batch_size
            )
        )
        history = model.fit_generator(
            data_generator_wrapper(
                lines[:num_train], batch_size, input_shape, anchors, num_classes
            ),
            steps_per_epoch=max(1, num_train // batch_size),
            validation_data=data_generator_wrapper(
                lines[num_train:], batch_size, input_shape, anchors, num_classes
            ),
            validation_steps=max(1, num_val // batch_size),
            epochs=epoch1 + epoch2,
            initial_epoch=epoch1,
            callbacks=[logging, checkpoint, reduce_lr, early_stopping],
        )
        model.save_weights(os.path.join(log_dir, "trained_weights_final.h5"))
        step2_train_loss = history.history["loss"]
        file = open(os.path.join(log_dir_time, "step2_loss.npy"), "w")
        with open(os.path.join(log_dir_time, "step2_loss.npy"), "w") as f:
            for item in step2_train_loss:
                f.write("%s\n" % item)
        file.close()
        step2_val_loss = np.array(history.history["val_loss"])
        file = open(os.path.join(log_dir_time, "step2_val_loss.npy"), "w")
        with open(os.path.join(log_dir_time, "step2_val_loss.npy"), "w") as f:
            for item in step2_val_loss:
                f.write("%s\n" % item)
        file.close()
```

7.3 DATABASE SCHEMA:

- ☆ **Registration:** When a new user registers, the backend connects to the IBM Cloudant and stores the user's credentials in the database.
- ☆ **Login:** To check if a user is already registered, the backend connects to Cloudant when they attempt to log in. They are an invalid user if they are not already registered.
- ☆ **IBM cloudant:** Stores the data which is registered.
- ☆ **app.py:** Connects both Frontend and the cloudant for the verification of user credentials



8.TESTING:

8.1 Test Case:

Test Case No.	Action	Expected Output	Actual Output	Result
1	Register for the website	Stores name, email, and password in Database	Stores name, email, and password in Database	Pass
2	Login to the website	Giving the right credentials, results in a successful login.	Giving the right credentials, results in a successful login.	Pass
3	Detecting the disease	It should predict the disease	It should predictthe disease	Pass

8.2 USER ACCEPTANCE TESTING:

Section	Total Cases	Not Tested	Fail	Pass
Registration	9	0	0	9
Login	40	0	0	40
Security	2	0	0	2
Disease Detection	10	0	0	10
Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2

9.RESULTS:

9.1 PERFORMANCE METRICS:

S.No.	**Parameter**	**Values**
1.	Model Summary	To evaluate object detection models like R-CNN and YOLO, the mean average precision (mAP) is used. ThemAP compares theground-truth bounding boxto thedetected box and returns a score.
2.	Accuracy	Training Accuracy – 89% Validation Accuracy – 95%
3.	Confidence Score (OnlyYolo Projects)	Class Detected – 91% Confidence Score – 90%

10.ADVANTAGES AND DISADVANTAGES:

ADVANTAGES:

- Treatment regimens are now more precise and effective to image processing technologies.
- It is time and money-saving process.
- Even with a bigger volume of users, the model will perform well.
- The pixels in an image can be altered in image processing to achieve any desired density and contrast.
- High pixel quality is provided, making it simple

DISADVANTAGES:

- AI-Models are Susceptible to security risks.
- Inaccuracies are still possible.
- Although AI has come a long way, human surveillance is still essential.

11.CONCLUSION:

It is feasible to attain adequate accuracy rates in this AI model even without a sizable dataset and high-quality photos. Accurate segmentation gives us information about the disease's location, which is helpful in the pre-processing of data used in classification since it enables the YOLO model to concentrate on the relevant area. Our approach offers a way to categorise various illnesses with better quality and more data. Our AI-based techniques help patients save time and money by reducing their medical expenses.

12.FUTURE SCOPE:

From simple to complicated duties, such as taking phone calls to reviewing medical records, evaluating radiology pictures, developing clinical diagnosis and treatment plans, and even conversing with patients, may be included in the future of AI in the detection of skin disorders. Already in use, AI improves comfort and effectiveness, lowers costs and errors, and generally makes it simpler for more patients to access the healthcare they require. Much while AI is already being utilised in healthcare, its potential to improve patient engagement in their own treatment and speed up patient access to care will make it even more crucial.

13.APPENDIX:

SOURCE CODE:

```
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
import argparse

from yolo3 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

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\LENOVO\Desktop\yolo_structure\2_Training\src'
print(src_path)
utils_path = r'C:\Users\LENOVO\Desktop\yolo_structure\Utils'
print(utils_path)
sys.path.append(src_path)
sys.path.append(utils_pathos.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('f9477f0b-8afb-4cc8-87de-be3cda3aebd9-bluemix','_GEL-0DBs-
R8T6msJoTCVTIeoygra8oMX-vEGp1s_UK7', connect=True
# Create a database using an initialized client
my_database = client.create_database('skin disease')
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)
    rint(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)
```

GitHub & Project Demo Link:

Github: https://github.com/IBM-EPBL/IBM-Project-4899-1658742294

Project Demo Link: https://drive.google.com/file/d/1qQWuyoYC_9-JFYCkwnu6_DoQ2OiYLcfy/view?usp=share_link