AI-BASED LOCALIZATION AND CLASSIFICATION OF SKIN DISEASE WITH ERYTHEMA

Submited by

Nithish Kumar S

(19ECR122)

Raja Kumaran R

(19ECR135)

Ranjith R

(19ECR140)

Santhaprakash M

(19ECR162)

TEAM ID : PNT2022TMID04689

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

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 noninvasive 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 a novel method to sequentially combine accurate segmentation and classification models. Given an image of the skin, we decompose the image to normalize and extract high-level features. Using a neural 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.

1.2 PURPOSE

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.

CHAPTER 2

LITERATURE SURVEY

2.1 EXISTING PROBLEMS

Computer-aided diagnosis (CAD) is a computer-based system that is used in the medical imaging field to aid healthcare workers in their diagnoses[1] . CAD has become a mainstream tool in several medical fields such as mammography and colonography[1,2] . 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 noninvasive 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. Trabelsi et al.[3] experimented 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. Rajab et al.[4] implemented an ISODATA clustering algorithm to find the optimal threshold for the segmentation of skin lesions. 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.[5] implemented an improved version of the fuzzy clustering algorithm using the RGB, HSV, and LAB color spaces to create a model that is more robust to noisy data.

2.2 REFERENCE

[1] Son, H. M., Jeon, W., Kim, J., Heo, C. Y., Yoon, H. J., Park, J. U., & Chung, T. M. (2021). AI-based localization and classification of skin disease with erythema. Scientific Reports, 11(1), 1-14.

[2] Kumar, N. V., Kumar, P. V., Pramodh, K., & Karuna, Y. (2019, March). Classification of Skin diseases using Image processing and SVM. In 2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN) (pp. 1-5). IEEE.

[3] Filimon, D. M., & Albu, A. (2014, May). Skin diseases diagnosis using artificial neural networks. In 2014 IEEE 9th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI) (pp. 189-194). IEEE.

[4] Grzesiak-Kopeć, K., Nowak, L., & Ogorzałek, M. (2015, June). Automatic diagnosis of melanoid skin lesions using machine learning methods. In International Conference on Artificial Intelligence and Soft Computing (pp. 577-585). Springer, Cham.

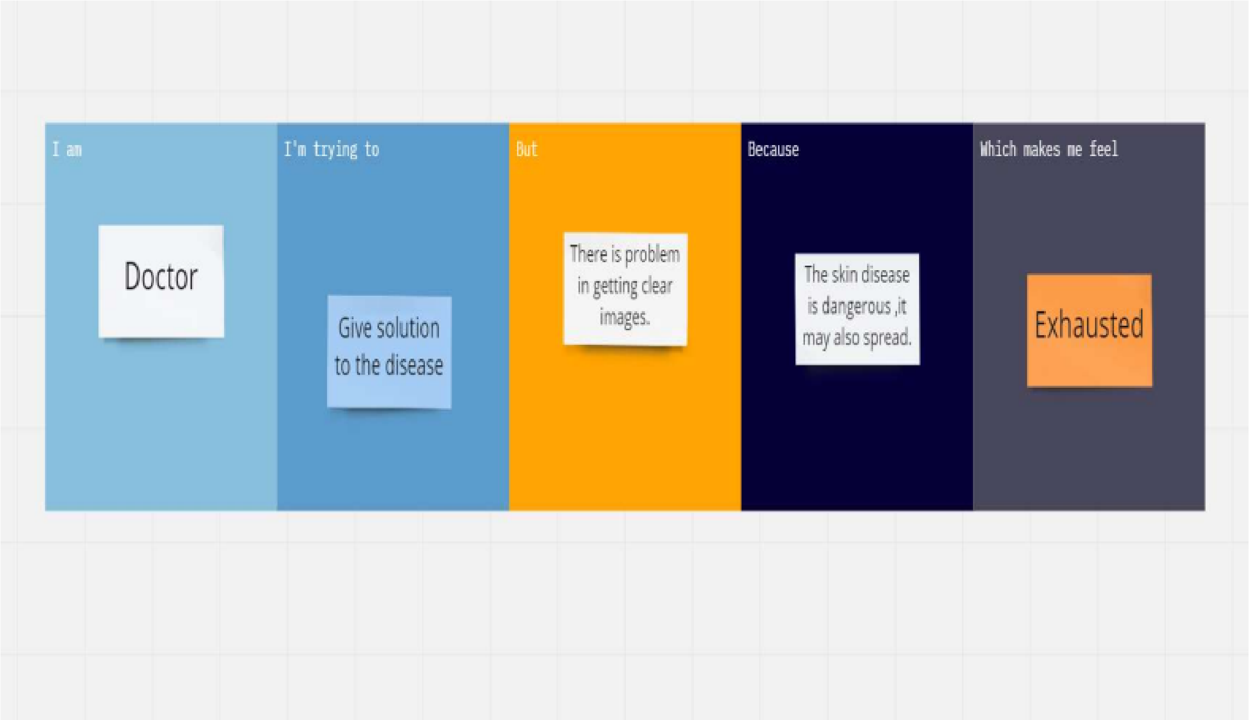
[5] Sumithra, R., Suhil, M., & Guru, D. S. (2015). Segmentation and classification of skin lesions for disease diagnosis. Procedia Computer Science, 45, 76-85.

[6] Kolkur, S., & Kalbande, D. R. (2016, November). Survey of texture based feature extraction for skin disease detection. In 2016 International Conference on ICT in Business Industry & Government (ICTBIG) (pp. 1-6). IEEE.

[7] Wu, Z. H. E., Zhao, S., Peng, Y., He, X., Zhao, X., Huang, K., ... & Li, Y. (2019). Studies on different CNN algorithms for face skin disease classification based on clinical images. IEEE Access, 7, 66505-66511.

[8] ALEnezi, N. S. A. (2019). A method of skin disease detection using image processing and machine learning. Procedia Computer Science, 163, 85-92.

2.3 PROBLEM STATEMENT DEFINITION

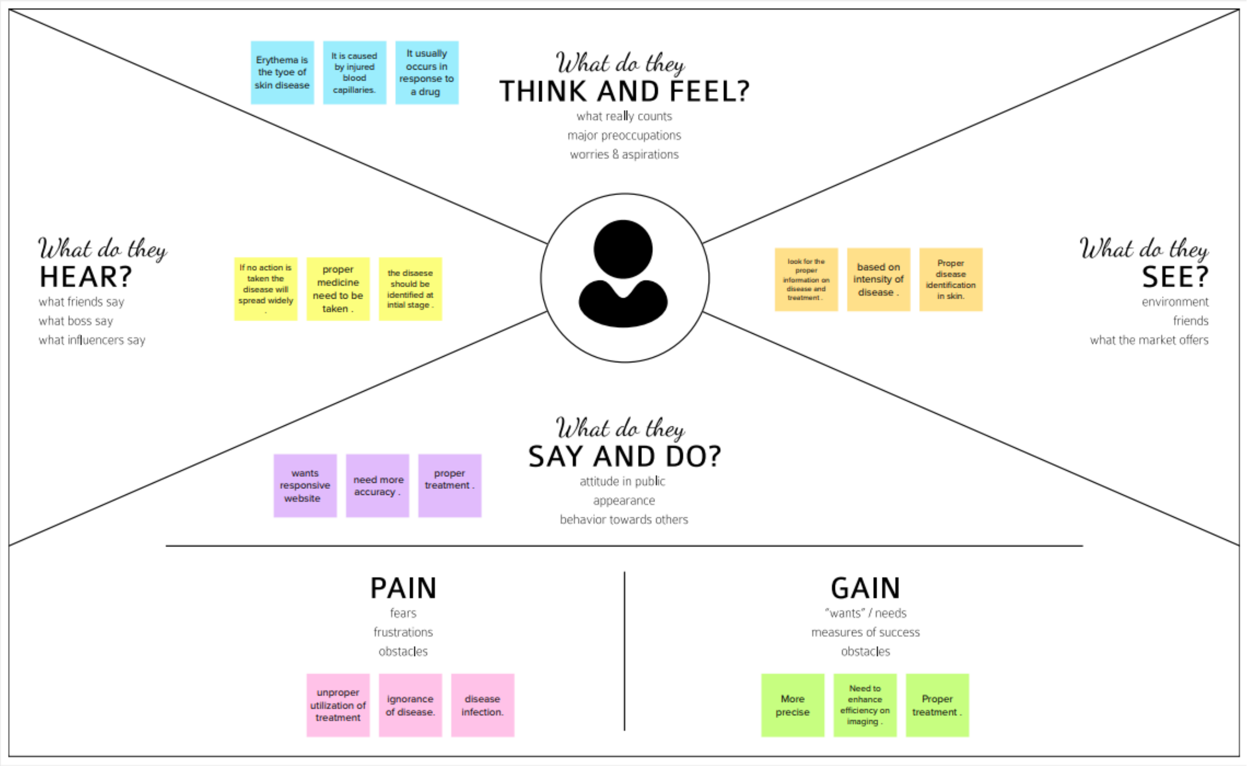


I am	I'm trying to	But	Because	Which makes me feel
Doctor	Give solution to the disease	There is problem in getting clear images.	The skin disease is dangerous ,it may also spread.	Exhausted

CHAPTER 3

IDEATION AND PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS



Erythema is the tyoe of skin disease

It is caused by injured blood capillaries.

It usually occurs in response to a drug

If no action is taken the disease will spread widely .

proper medicine need to be taken .

the disaese should be identified at intial stage .

look for the proper information on disease and treatment .

based on intensity of disease .

Proper disease identification in skin.

wants responsive website

need more accuracy .

proper treatment .

unproper utilization of treatment

ignorance of disease.

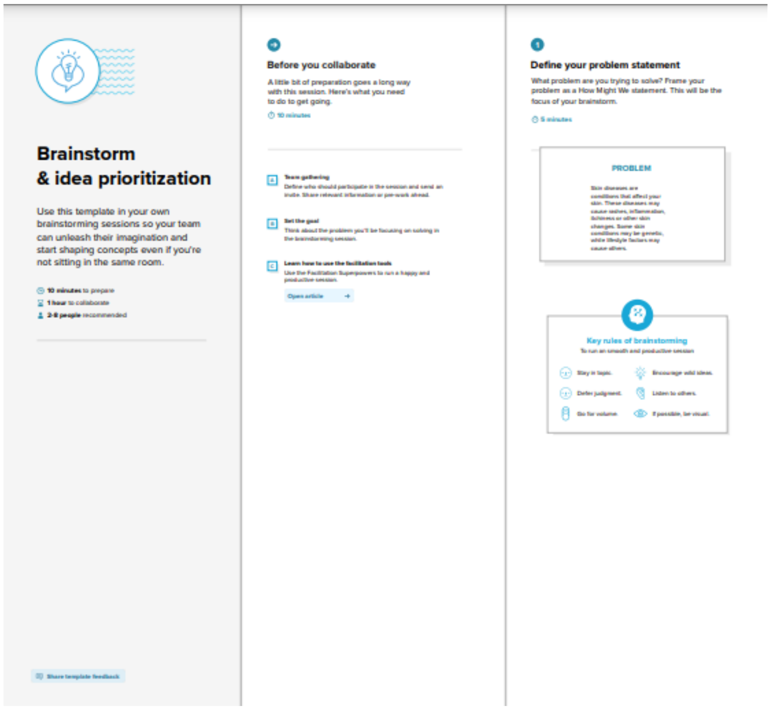
disease infection.

More precise

Need to enhance efficiency on imaging .

Proper treatment .

3.2 IDEATION AND BRAINSTORMING



Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

10 minutes to prepare
1 hour to collaborate
2-8 people recommended

Share template feedback

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.

10 minutes

A **Team gathering**
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B **Set the goal**
Think about the problem you'll be focusing on solving in the brainstorming session.

C **Learn how to use the facilitation tools**
Use the Facilitation Superpowers to run a happy and productive session.

Open article →

1 Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

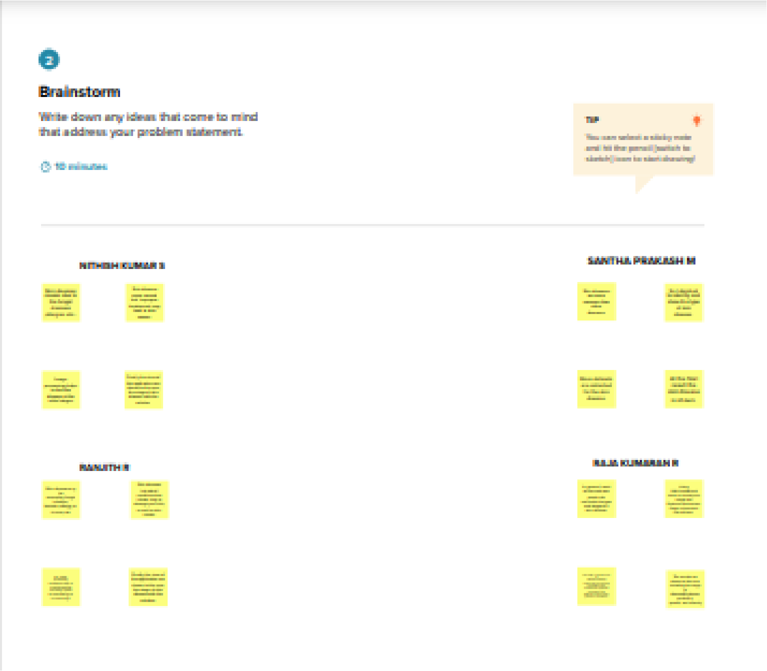
PROBLEM

Skin diseases are conditions that affect your skin. These diseases may cause rashes, inflammation, itchiness or other skin changes. Some skin conditions may be genetic, while lifestyle factors may cause others.

Key rules of brainstorming

To run an smooth and productive session

Stay in topic.
Encourage wild ideas.
Defer judgment.
Listen to others.
Go for volume.
If possible, be visual.



2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

TIP

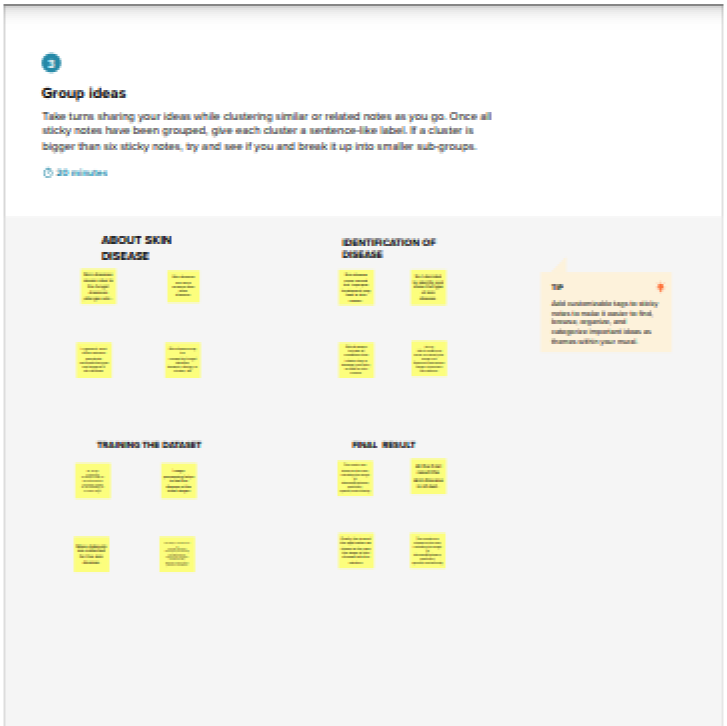
You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

NITHISH KUMAR S

SANTHA PRAKASH M

RANJITH R

RAJA KUMARAN R



3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

ABOUT SKIN DISEASE

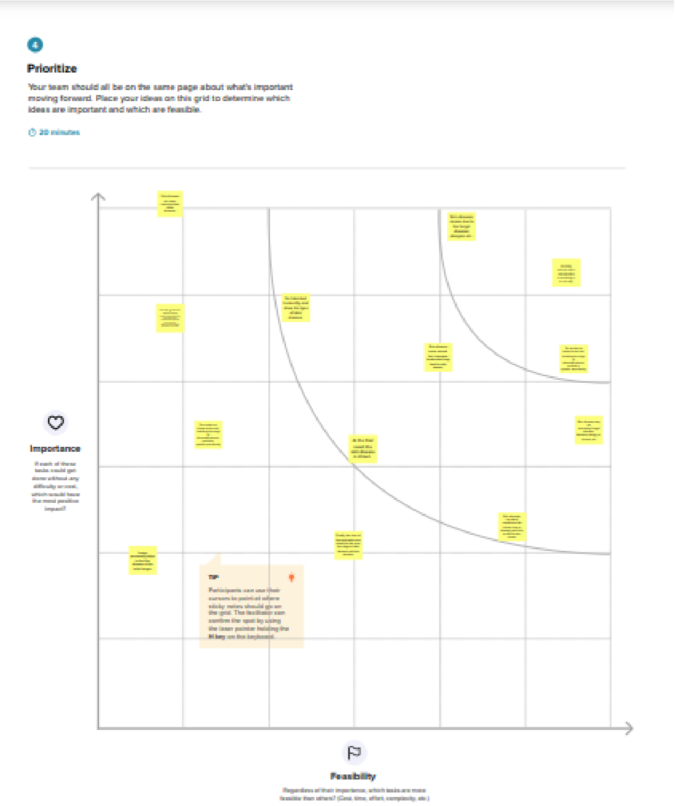
IDENTIFICATION OF DISEASE

TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

TRAINING THE DATASET

FINAL RESULT



4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes

Importance

If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?

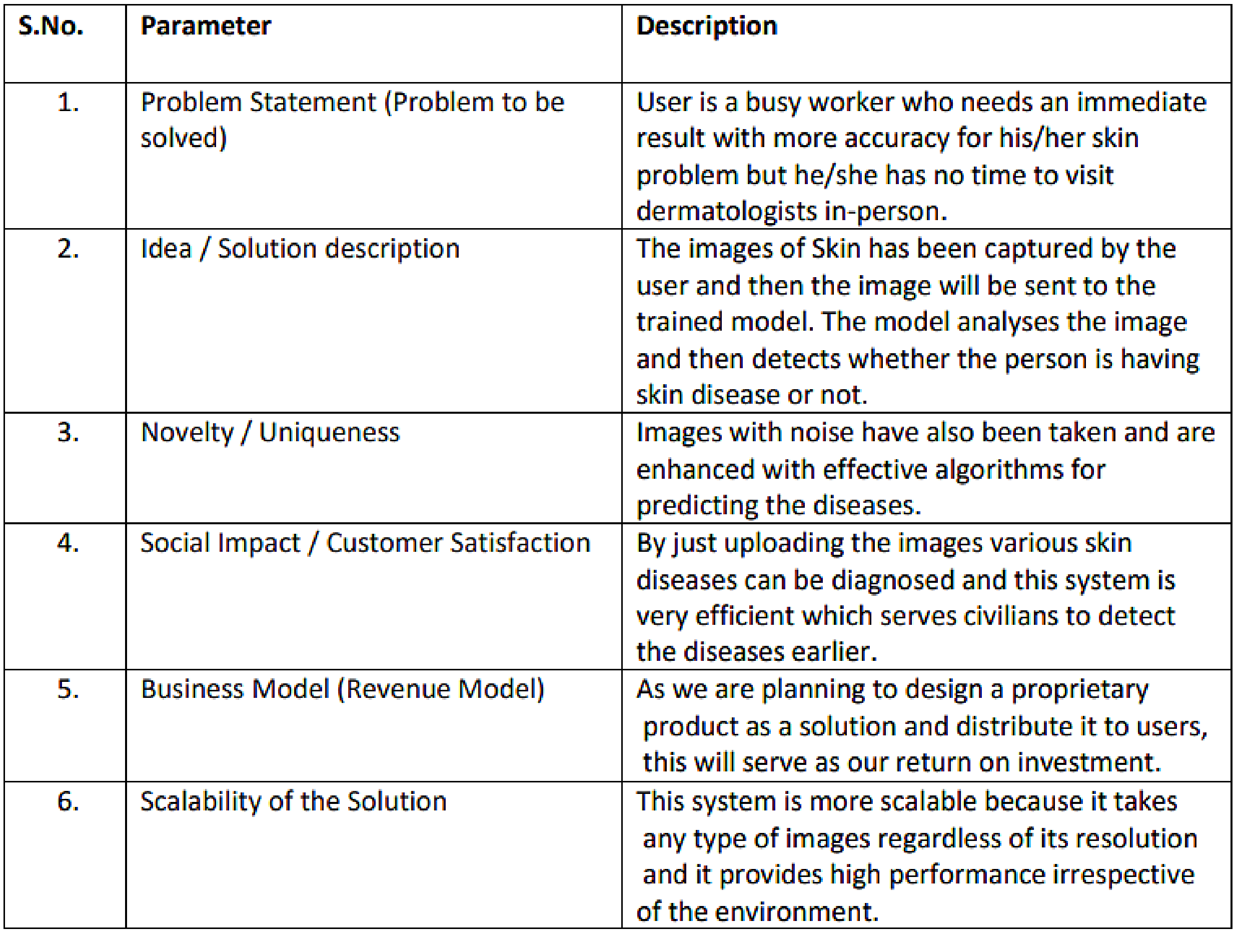
TIP

Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the **H key** on the keyboard.

Feasibility

Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

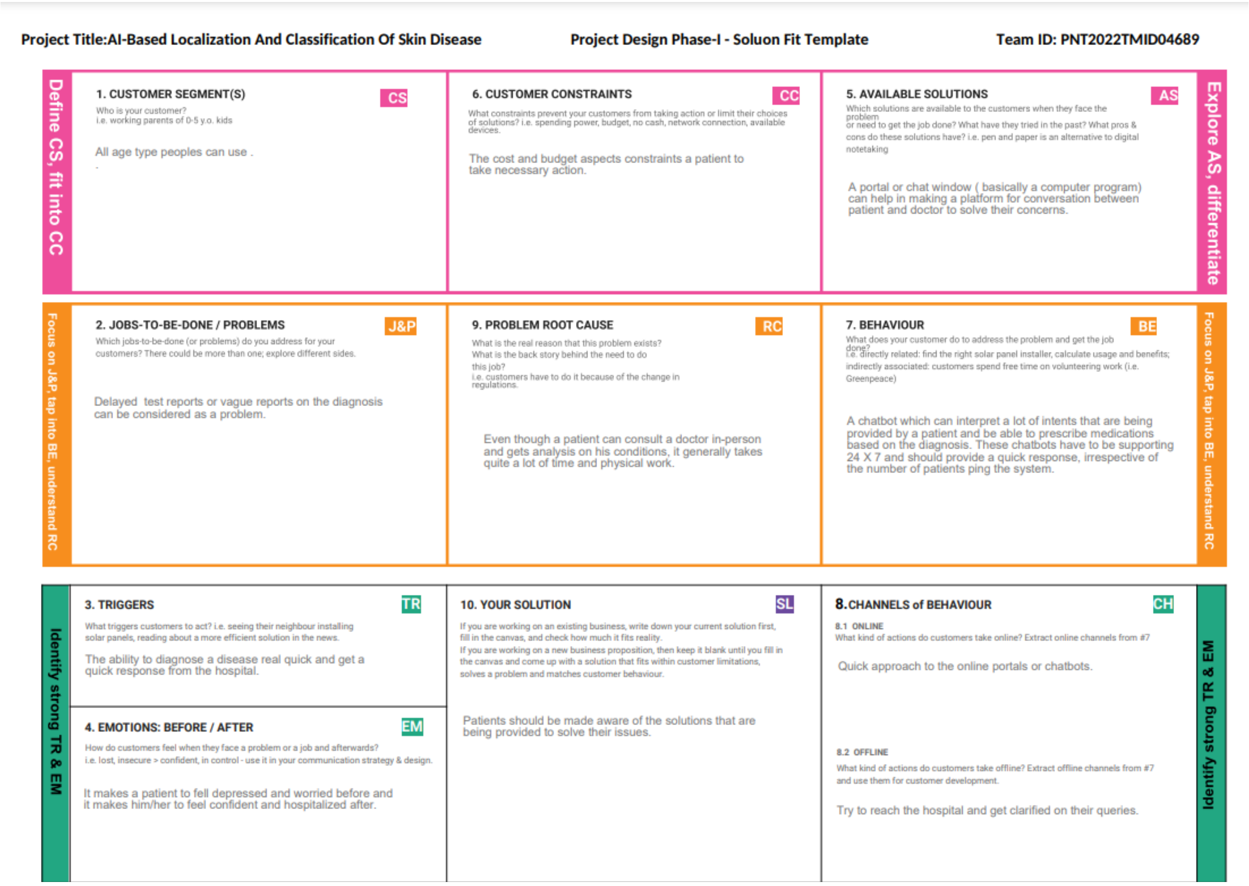
3.3 PROPOSED SOLUTION



S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	User is a busy worker who needs an immediate result with more accuracy for his/her skin problem but he/she has no time to visit dermatologists in-person.
2.	Idea / Solution description	The images of Skin has been captured by the user and then the image will be sent to the trained model. The model analyses the image and then detects whether the person is having skin disease or not.
3.	Novelty / Uniqueness	Images with noise have also been taken and are enhanced with effective algorithms for predicting the diseases.
4.	Social Impact / Customer Satisfaction	By just uploading the images various skin diseases can be diagnosed and this system is very efficient which serves civilians to detect the diseases earlier.
5.	Business Model (Revenue Model)	As we are planning to design a proprietary product as a solution and distribute it to users, this will serve as our return on investment.
6.	Scalability of the Solution	This system is more scalable because it takes any type of images regardless of its resolution and it provides high performance irrespective of the environment.

The proposed solution is a prototype with a database of six common skin diseases, using which a patient can self-diagnose and get some prior knowledge of their skin disease before consulting a dermatologist. The proposed prototype provides a non-invasive method of skin disease detection where the patient provides a picture of the infected area as an input to the prototype and any further analysis is done on this input image.

3.4 PROBLEM SOLUTION FIT



Project Title:AI-Based Localization And Classification Of Skin Disease

Project Design Phase-I - Soluon Fit Template

Team ID: PNT2022TMID04689

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Who is your customer?
i.e. working parents of 0-5 y.o. kids

All age type peoples can use .

-

6. CUSTOMER CONSTRAINTS CC

What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices.

The cost and budget aspects constraints a patient to take necessary action.

5. AVAILABLE SOLUTIONS AS

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? i.e. pen and paper is an alternative to digital notetaking

A portal or chat window (basically a computer program) can help in making a platform for conversation between patient and doctor to solve their concerns.

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P

Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.

Delayed test reports or vague reports on the diagnosis can be considered as a problem.

9. PROBLEM ROOT CAUSE RC

What is the real reason that this problem exists?
What is the back story behind the need to do this job?
i.e. customers have to do it because of the change in regulations.

Even though a patient can consult a doctor in-person and gets analysis on his conditions, it generally takes quite a lot of time and physical work.

7. BEHAVIOUR BE

What does your customer do to address the problem and get the job done?
i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)

A chatbot which can interpret a lot of intents that are being provided by a patient and be able to prescribe medications based on the diagnosis. These chatbots have to be supporting 24 X 7 and should provide a quick response, irrespective of the number of patients ping the system.

Focus on J&P, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TR

What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news.

The ability to diagnose a disease real quick and get a quick response from the hospital.

4. EMOTIONS: BEFORE / AFTER EM

How do customers feel when they face a problem or a job and afterwards?
i.e. lost, insecure > confident, in control - use it in your communication strategy & design.

It makes a patient to fell depressed and worried before and it makes him/her to feel confident and hospitalized after.

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.

Patients should be made aware of the solutions that are being provided to solve their issues.

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE
What kind of actions do customers take online? Extract online channels from #7

Quick approach to the online portals or chatbots.

8.2 OFFLINE
What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.

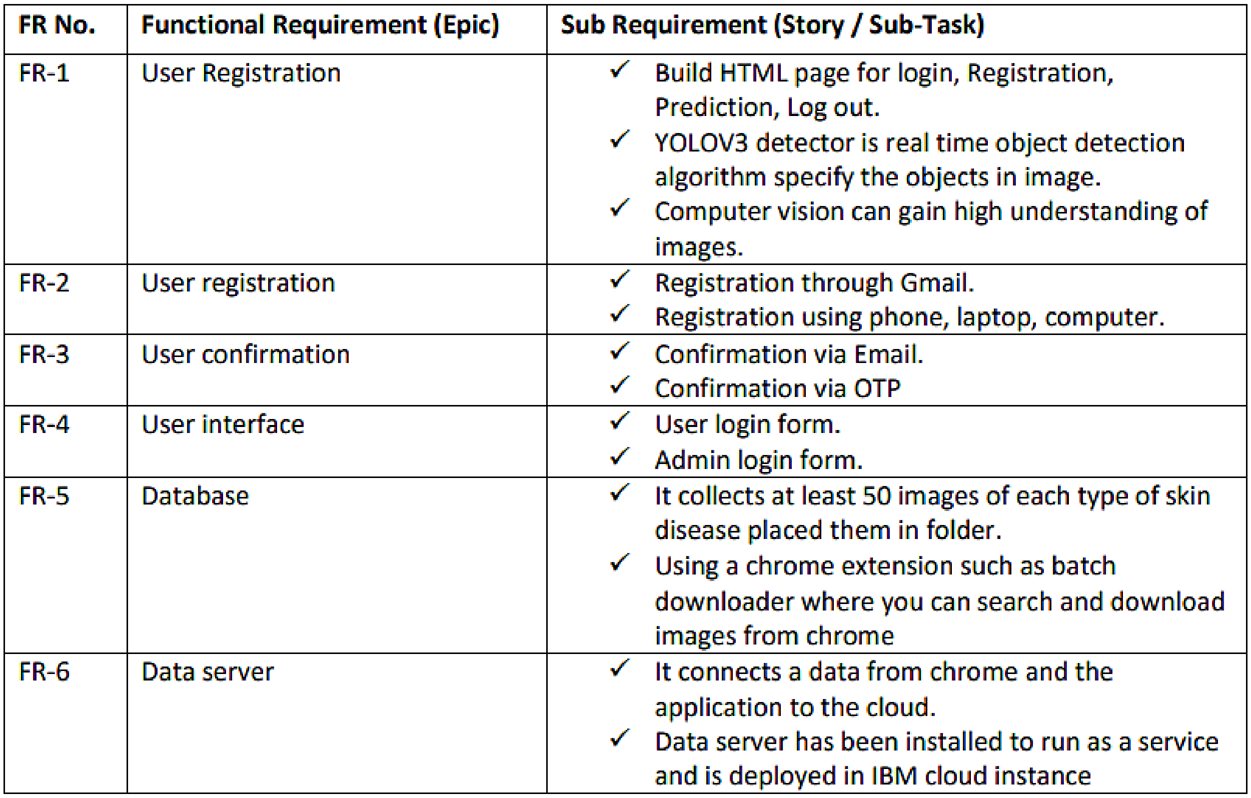
Try to reach the hospital and get clarified on their queries.

Identify strong TR & EM

CHAPTER 4

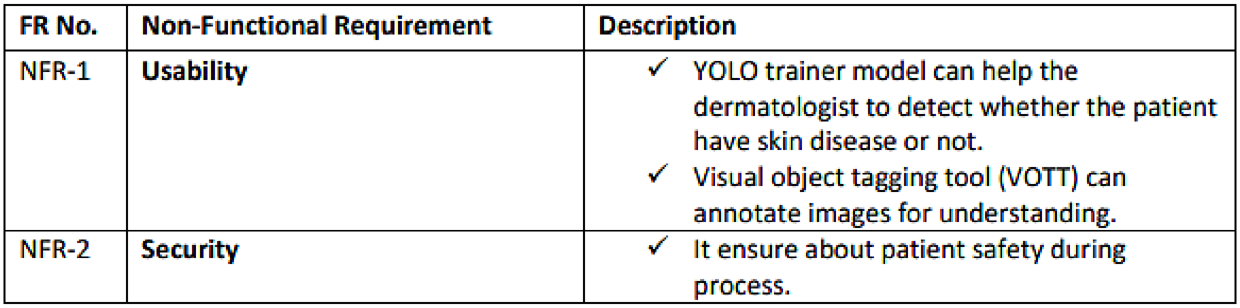
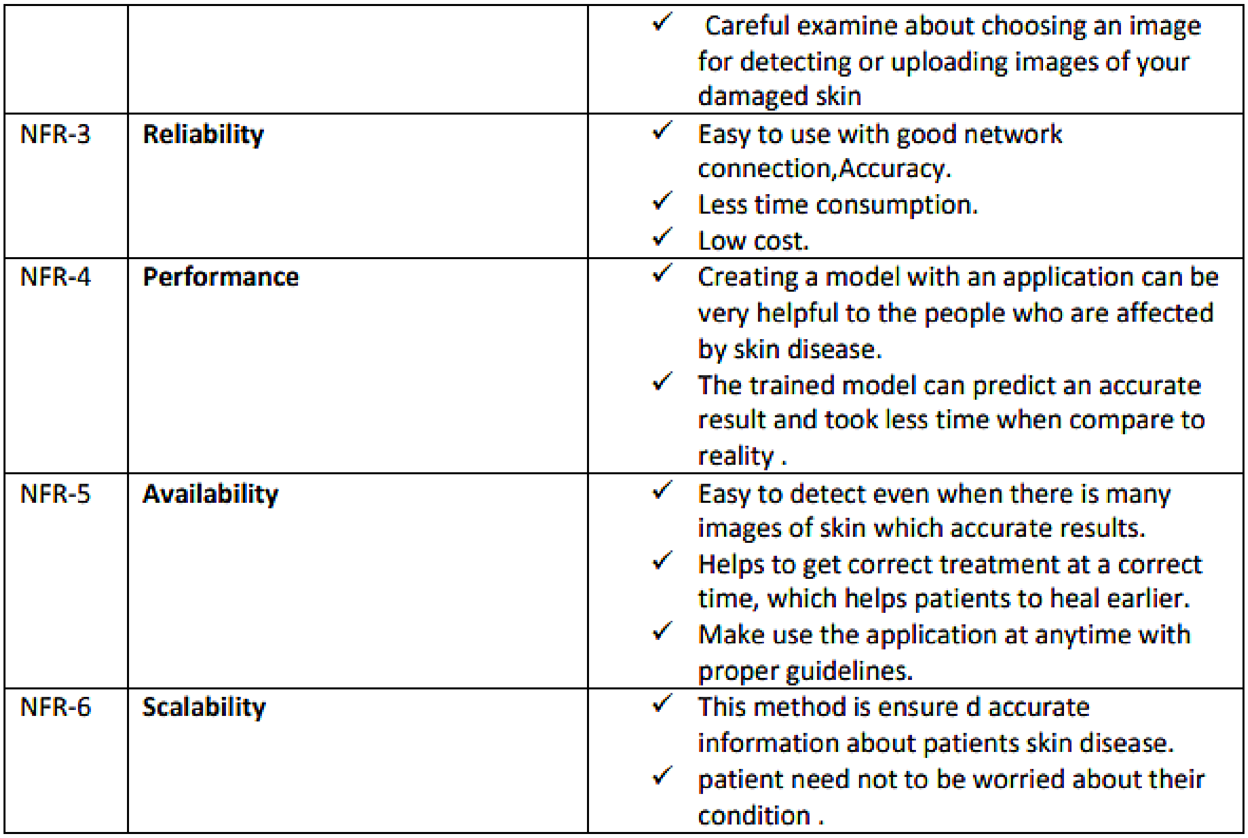
REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS



FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	✓ Build HTML page for login, Registration, Prediction, Log out. ✓ YOLOV3 detector is real time object detection algorithm specify the objects in image. ✓ Computer vision can gain high understanding of images.
FR-2	User registration	✓ Registration through Gmail. ✓ Registration using phone, laptop, computer.
FR-3	User confirmation	✓ Confirmation via Email. ✓ Confirmation via OTP
FR-4	User interface	✓ User login form. ✓ Admin login form.
FR-5	Database	✓ It collects at least 50 images of each type of skin disease placed them in folder. ✓ Using a chrome extension such as batch downloader where you can search and download images from chrome
FR-6	Data server	✓ It connects a data from chrome and the application to the cloud. ✓ Data server has been installed to run as a service and is deployed in IBM cloud instance

4.2 NON-FUNCTIONAL REQUIREMENTS

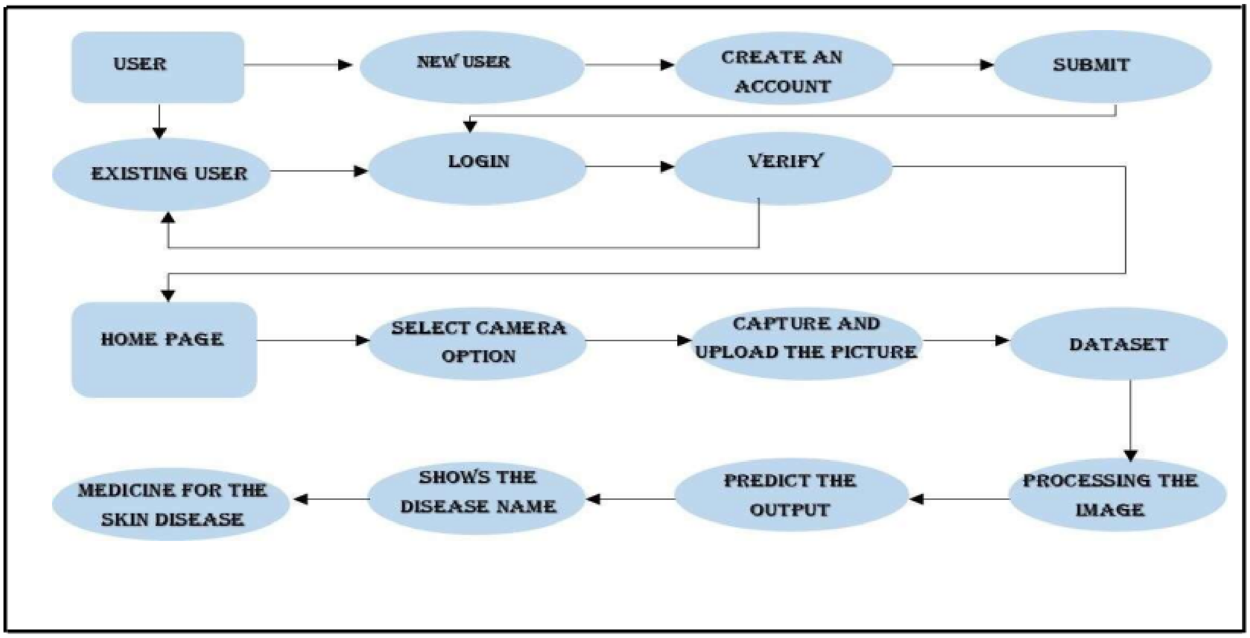
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	✓ YOLO trainer model can help the dermatologist to detect whether the patient have skin disease or not. ✓ Visual object tagging tool (VOTT) can annotate images for understanding.
NFR-2	**Security**	✓ It ensure about patient safety during process. ✓ Careful examine about choosing an image for detecting or uploading images of your damaged skin
NFR-3	**Reliability**	✓ Easy to use with good network connection,Accuracy. ✓ Less time consumption. ✓ Low cost.
NFR-4	**Performance**	✓ Creating a model with an application can be very helpful to the people who are affected by skin disease. ✓ The trained model can predict an accurate result and took less time when compare to reality .
NFR-5	**Availability**	✓ Easy to detect even when there is many images of skin which accurate results. ✓ Helps to get correct treatment at a correct time, which helps patients to heal earlier. ✓ Make use the application at anytime with proper guidelines.
NFR-6	**Scalability**	✓ This method is ensure d accurate information about patients skin disease. ✓ patient need not to be worried about their condition .

CHAPTER 5

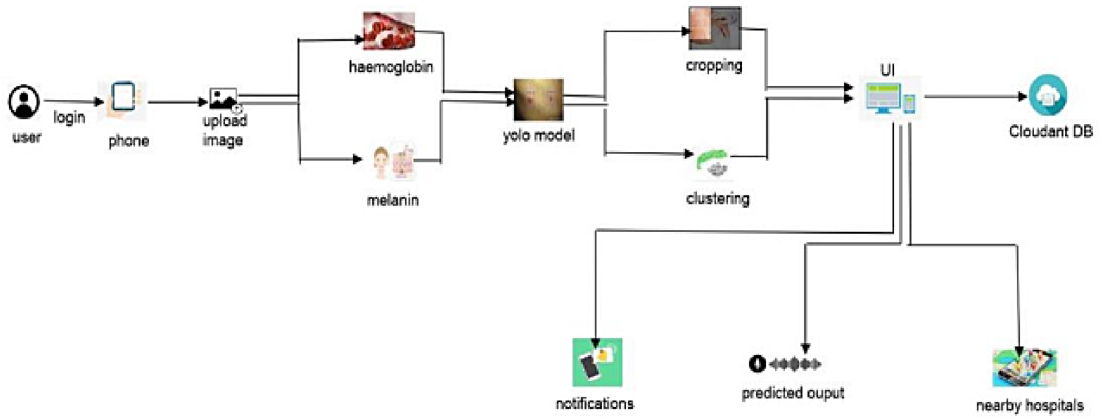
PROJECT DESIGN

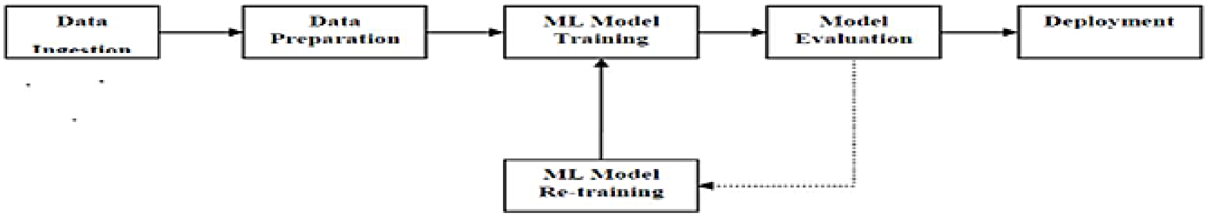
5.1 DATA FLOW DIAGRAMS

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.

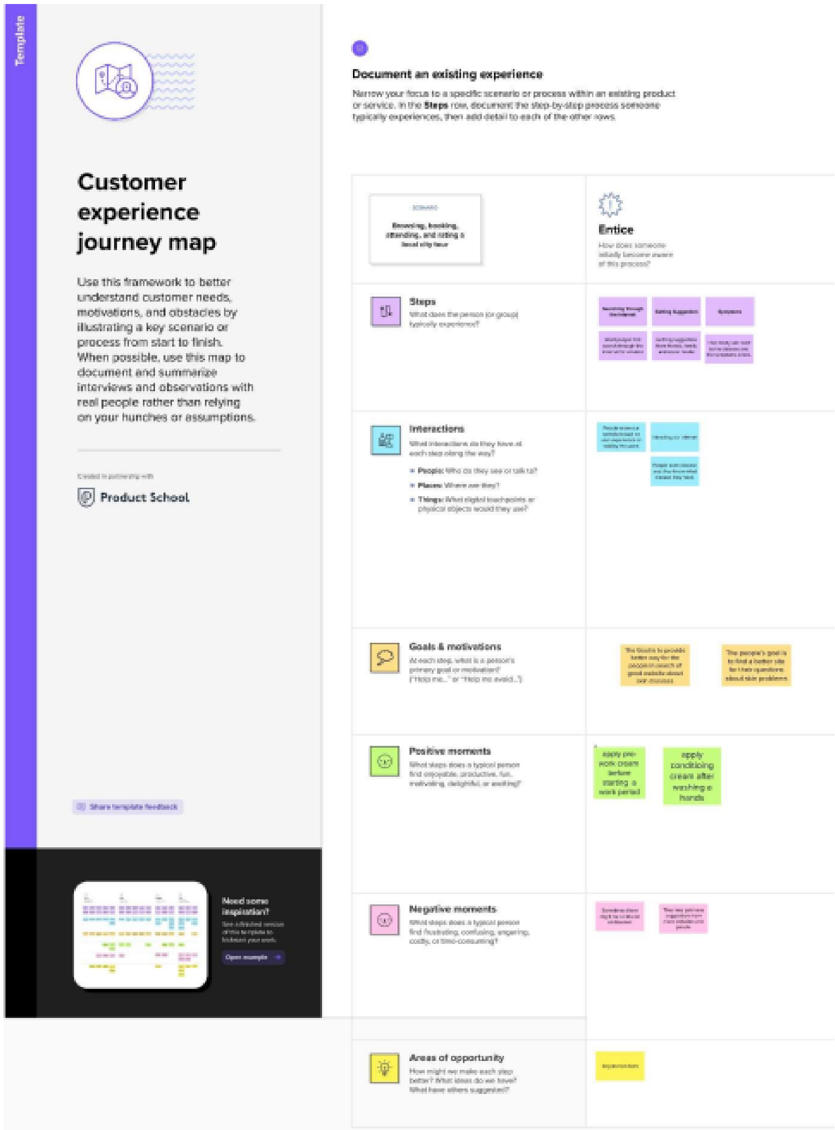


5.2 SOLUTIONS AND TECHNICAL ARCHITECTURE





5.3 USER STORIES



Template

Customer experience journey map

Use this framework to better understand customer needs, motivations, and obstacles by illustrating a key scenario or process from start to finish. When possible, use this map to document and summarize interviews and observations with real people rather than relying on your hunches or assumptions.

Created in partnership with

Product School

Share template feedback

Need some inspiration?

See a finished version of this template to kickstart your work.

Open example →

Document an existing experience

Narrow your focus to a specific scenario or process within an existing product or service. In the **Steps** row, document the step-by-step process someone typically experiences, then add detail to each of the other rows.

SCENARIO

Browsing, booking, attending, and rating a local city tour

Entice

How does someone initially become aware of this process?

Steps

What does the person (or group) typically experience?

[illegible]

[illegible]

Symptoms

[illegible]

[illegible]

[illegible]

Interactions

What interactions do they have at each step along the way?

- **People:** Who do they see or talk to?
- **Places:** Where are they?
- **Things:** What digital touchpoints or physical objects would they use?

[illegible]

[illegible]

[illegible]

Goals & motivations

At each step, what is a person's primary goal or motivation? ("Help me..." or "Help me avoid...")

The Goal is to provide better way for the people in search of good website about skin diseases

The people's goal is to find a better site for their questions about skin problems.

Positive moments

What steps does a typical person find enjoyable, productive, fun, motivating, delightful, or exciting?

apply pre-work cream before starting a work period

apply conditioning cream after washing a hands

Negative moments

What steps does a typical person find frustrating, confusing, angering, costly, or time-consuming?

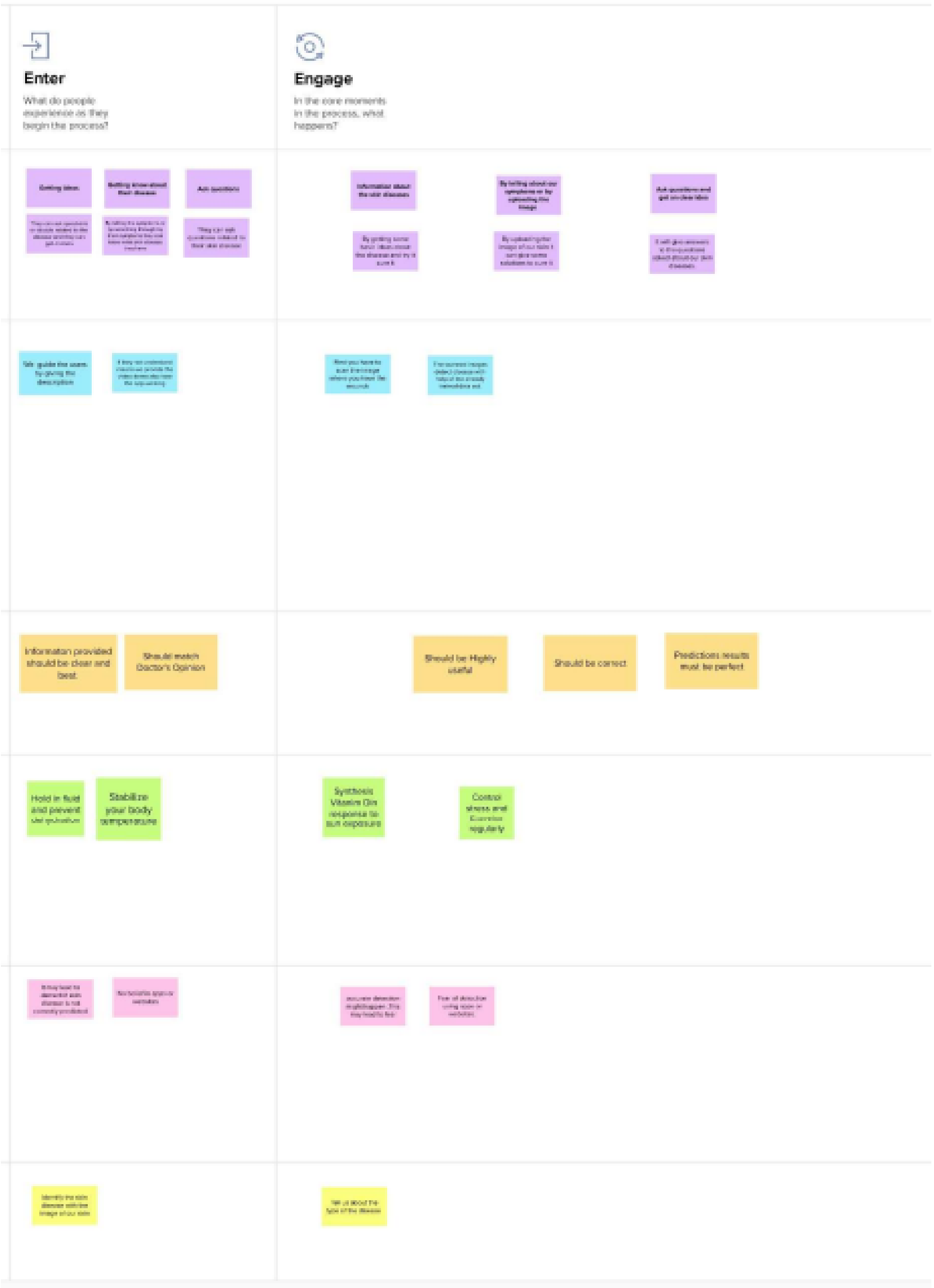
[illegible]

[illegible]

Areas of opportunity

How might we make each step better? What ideas do we have? What have others suggested?

[illegible]



Enter

What do people experience as they begin the process?

Engage

In the core moments in the process, what happens?

Getting ideas

Getting knowledge about their disease

Ask questions

[illegible]

[illegible]

They can ask questions related to their skin disease

Information about the skin diseases

By telling about our symptoms or by uploading the image

Ask questions and get an clear idea

By getting some basic ideas about the disease and try to cure it

By uploading the image of our skin it can give some solutions to cure it

It will give answers to the questions asked about our skin diseases

We guide the users by giving the description

[illegible]

[illegible]

[illegible]

Informaton provided should be clear and best.

Should match Doctor's Opinion

Should be Highly useful

Should be correct

Predictions results must be perfect

Hold in fluid and prevent [illegible]

Stabilize your body temperature

Synthesis Vitamin D in response to sun exposure

Control stress and Exercise regularly

It may lead to [illegible] skin disease is not correctly predicted

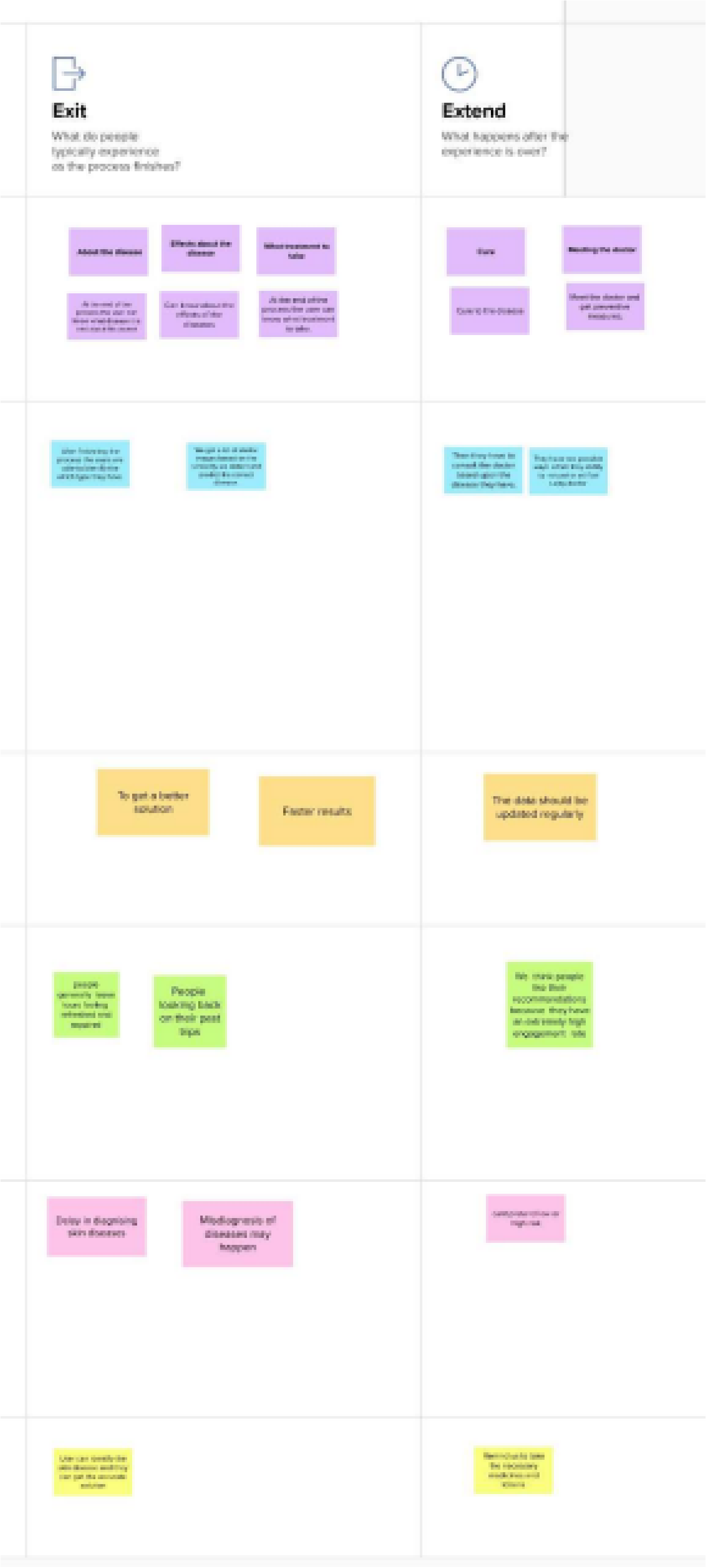
No trust [illegible] apps or websites

[illegible] detection might happen. This may lead to [illegible]

Fear of detection using apps or websites

Identify the skin disease with the image of our skin

Tell us about the type of the disease



Exit

What do people typically experience as the process finishes?

About the disease

Effects about the disease

What treatment to take

At the end of the process the user can [illegible] what disease he [illegible] cure the disease

Can know about the effects of the diseases

At the end of the process the user can know what treatment to take.

After finishing the process the users are [illegible] which type they have

We get a lot of skin images based on the [illegible] and predict the correct disease

To get a better solution

Faster results

people generally leave tours feeling refreshed and relaxed

People looking back on their past trips

Delay in diagnosing skin diseases

Misdiagnosis of diseases may happen

User can identify the skin disease and they can get the accurate solution

Extend

What happens after the experience is over?

Cure

Meeting the doctor

Cure to the disease

Meet the doctor and get preventive measures.

Then they have to consult the doctor based upon the disease they have.

They have [illegible]

The data should be updated regularly

We think people like their recommendations because they have an extremely high engagement rate

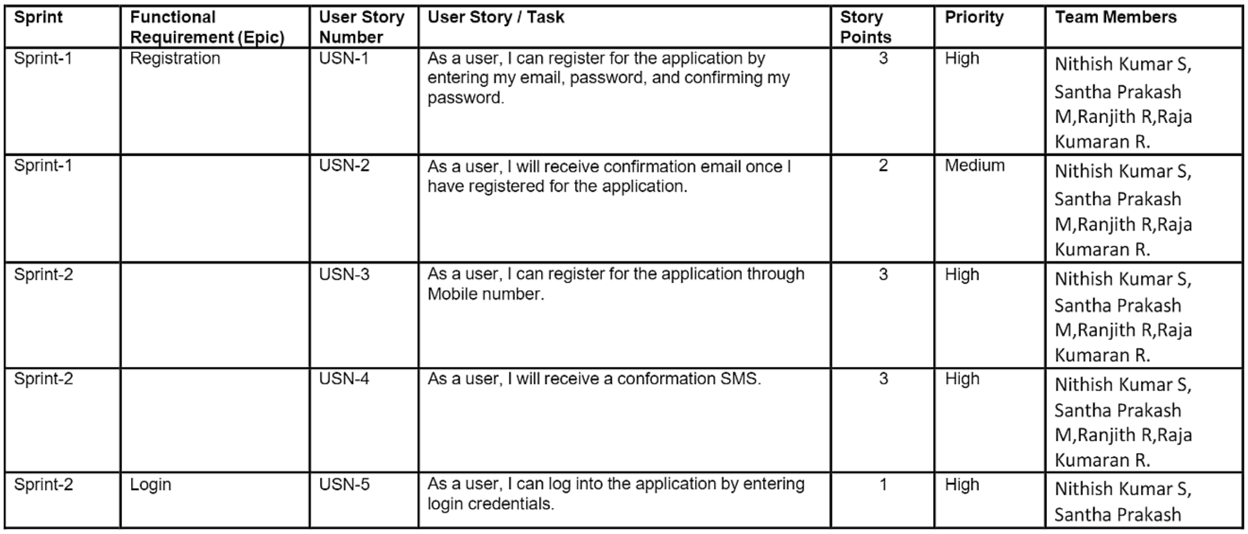
[illegible] high risk

Reminds to take the necessary medicines and [illegible]

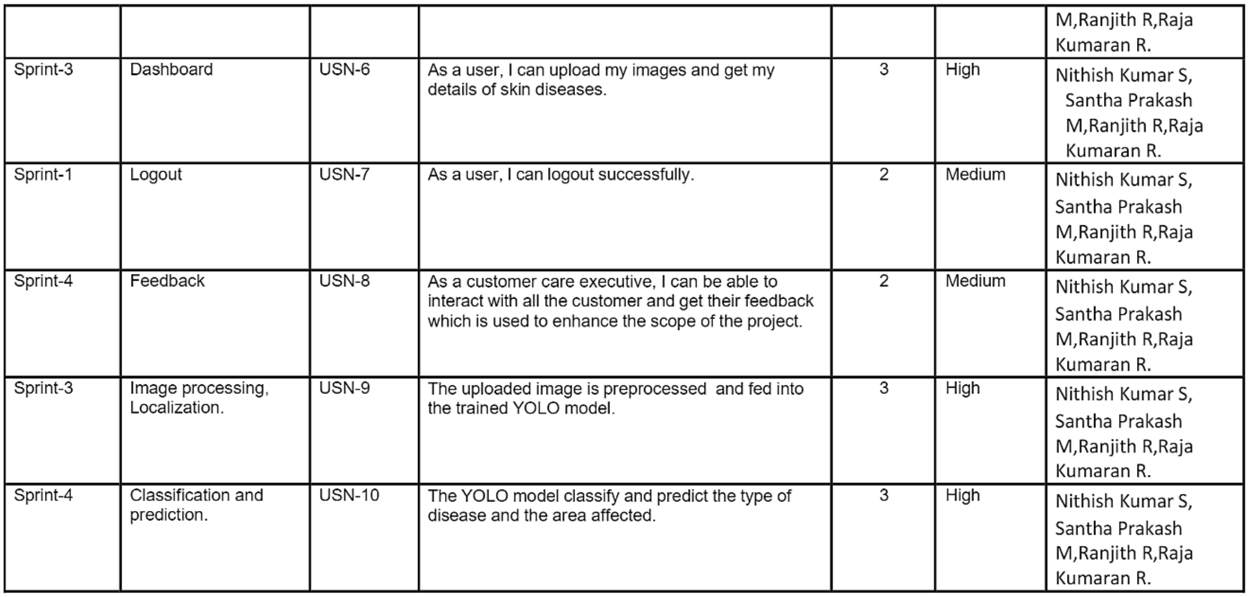
CHAPTER 6

PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING AND ESTIMATION

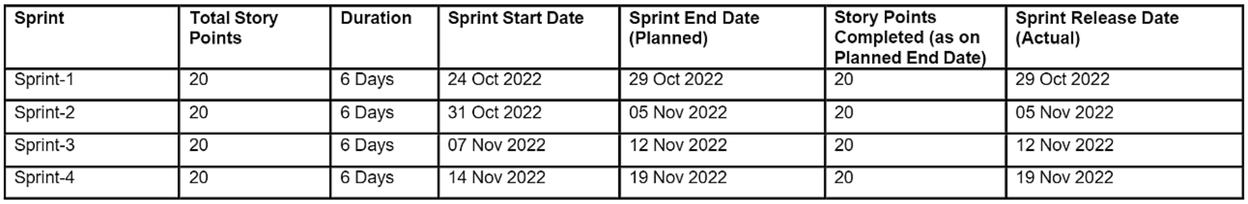


Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	3	High	Nithish Kumar S, Santha Prakash M,Ranjith R,Raja Kumaran R.
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application.	2	Medium	Nithish Kumar S, Santha Prakash M,Ranjith R,Raja Kumaran R.
Sprint-2		USN-3	As a user, I can register for the application through Mobile number.	3	High	Nithish Kumar S, Santha Prakash M,Ranjith R,Raja Kumaran R.
Sprint-2		USN-4	As a user, I will receive a conformation SMS.	3	High	Nithish Kumar S, Santha Prakash M,Ranjith R,Raja Kumaran R.
Sprint-2	Login	USN-5	As a user, I can log into the application by entering login credentials.	1	High	Nithish Kumar S, Santha Prakash



						M,Ranjith R,Raja Kumaran R.
Sprint-3	Dashboard	USN-6	As a user, I can upload my images and get my details of skin diseases.	3	High	Nithish Kumar S, Santha Prakash M,Ranjith R,Raja Kumaran R.
Sprint-1	Logout	USN-7	As a user, I can logout successfully.	2	Medium	Nithish Kumar S, Santha Prakash M,Ranjith R,Raja Kumaran R.
Sprint-4	Feedback	USN-8	As a customer care executive, I can be able to interact with all the customer and get their feedback which is used to enhance the scope of the project.	2	Medium	Nithish Kumar S, Santha Prakash M,Ranjith R,Raja Kumaran R.
Sprint-3	Image processing, Localization.	USN-9	The uploaded image is preprocessed and fed into the trained YOLO model.	3	High	Nithish Kumar S, Santha Prakash M,Ranjith R,Raja Kumaran R.
Sprint-4	Classification and prediction.	USN-10	The YOLO model classify and predict the type of disease and the area affected.	3	High	Nithish Kumar S, Santha Prakash M,Ranjith R,Raja Kumaran R.

6.2 SPRINT DELIVERY SCHEDULE



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	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

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 per Sprint

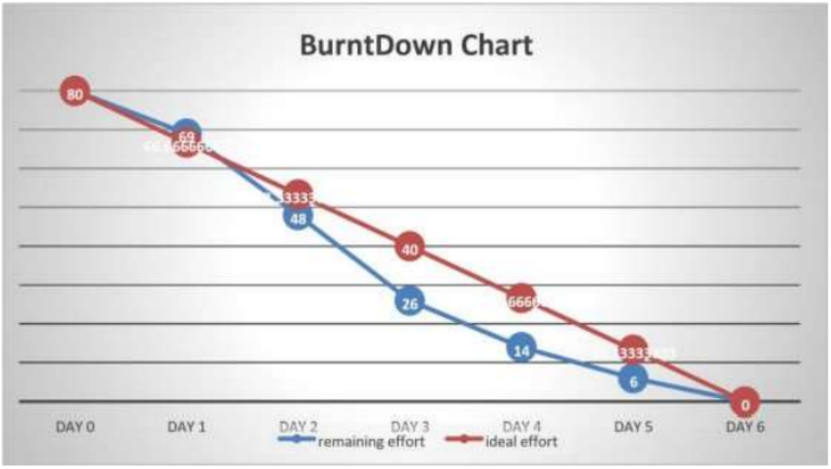
Velocity = Points per Sprint

AV = 20 /6 =4

Therefore, the AVERAGE VELOCITY IS 4 POINTS PER SPRINT.

BURNOUT CHAT :

A burndown chart shows the amount of work that has been completed in an epic or sprint, and the total work remaining. Burndown charts are used to predict your team's likelihood of completing their work in the time available.



CHAPTER 7

CODING AND SOLUTIONING

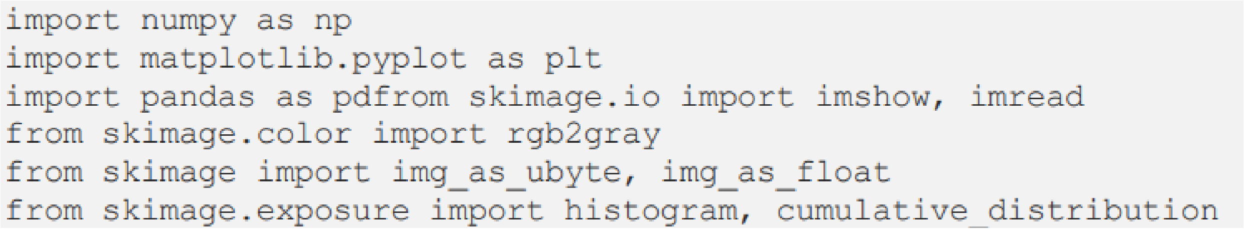
7.1 SOLUTIONING

SPRINT 1

Image Processing

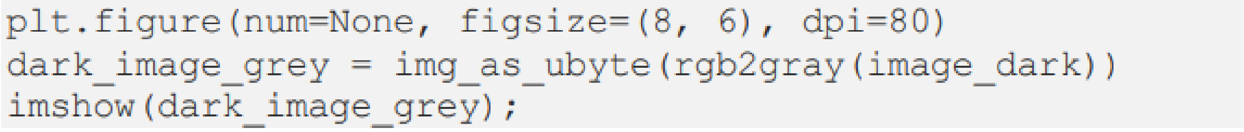
Histogram Manipulation

Import the required libraries.



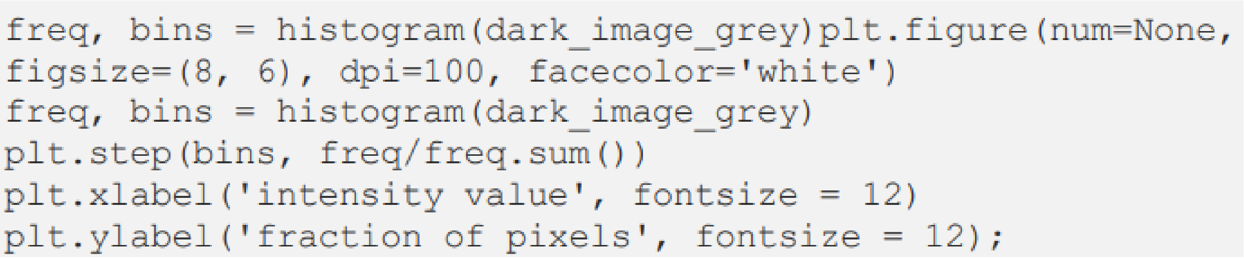
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pdfrom skimage.io import imshow, imread
from skimage.color import rgb2gray
from skimage import img_as_ubyte, img_as_float
from skimage.exposure import histogram, cumulative_distribution
```

Convert the image to greyscale.



```
plt.figure(num=None, figsize=(8, 6), dpi=80)
dark_image_grey = img_as_ubyte(rgb2gray(image_dark))
imshow(dark_image_grey);
```

Extract the image's value histogram.



```
freq, bins = histogram(dark_image_grey)plt.figure(num=None,
figsize=(8, 6), dpi=100, facecolor='white')
freq, bins = histogram(dark_image_grey)
plt.step(bins, freq/freq.sum())
plt.xlabel('intensity value', fontsize = 12)
plt.ylabel('fraction of pixels', fontsize = 12);
```

Intensity Values of Image

It is very clear that this histogram does not resemble a normal distribution. You might be tempted to try and snap this distribution into a normal distribution. However there is a slightly more intuitive way to handle this issue.

Remember that the theoretical Cumulative Distribution Function (CDF) for a normal distribution is a straight line. This being the case, it is better to snap the CDF of our image into a straight line.

Actual CDF of the Image

To do this, we can make use of the interpolate function in NumPy.

interpolation = np.interp(freq, target_freq, target_bins)

Use the interpolation to help us adjust the actual CDF.

```
dark_image_eq =
img_as_ubyte(interpolation[dark_image_grey].astype(int))
```

View the actual image.

```
imshow(dark_image_eq);
```

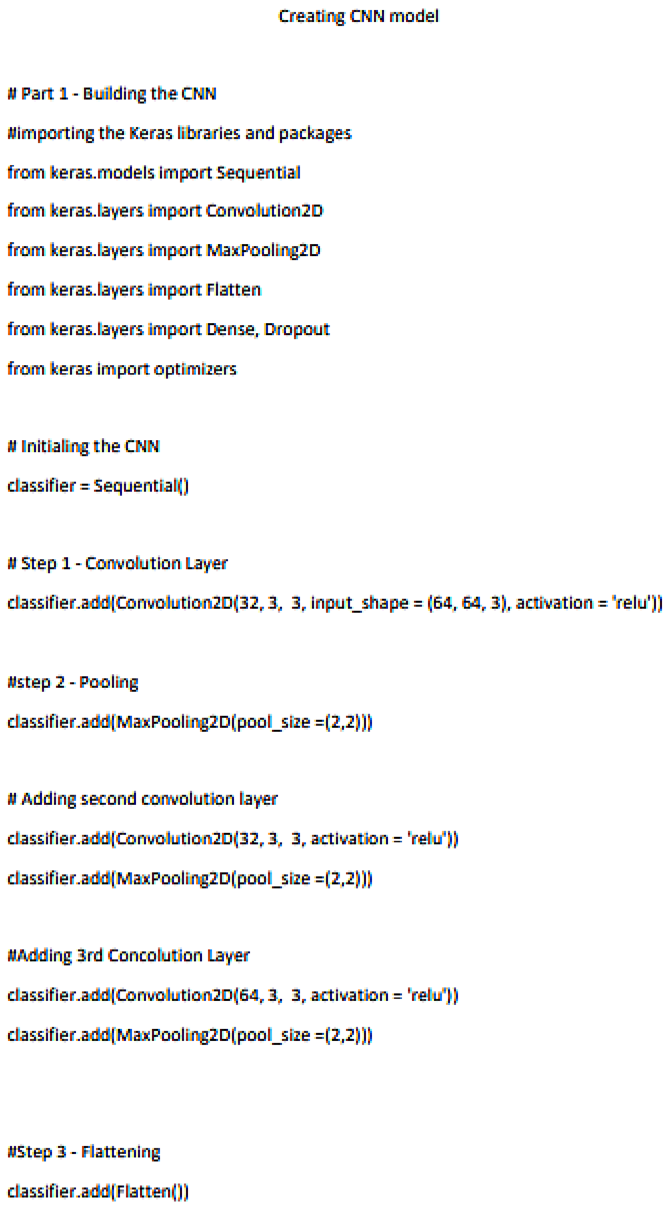
Create a function which will adjust the CDF of any image we feed it.

```
def histogram_adjuster(image):
    dark_image_grey = img_as_ubyte(rgb2gray(image))
    freq, bins = cumulative_distribution(dark_image_grey)
target_bins = np.arange(255)
    target_freq = np.linspace(0, 1, len(target_bins))
interpolation = np.interp(freq, target_freq, target_bins)
    dark_image_eq =
    img_as_ubyte(interpolation[dark_image_grey].astype(int))
    freq_adj, bins_adj = cumulative_distribution(dark_image_eq)


    fig, axes = plt.subplots(1, 2, figsize=(15,7));
    imshow(dark_image_grey, ax = axes[0]);
    imshow(dark_image_eq, ax = axes[1]);

    axes[0].axis('off')
    axes[1].axis('off')
    axes[0].set_title('Unadjusted Image', fontsize = 17)
    axes[1].set_title('Adjusted Image', fontsize = 17)

    fig, axes = plt.subplots(1, 1, figsize=(19,7));
    plt.step(bins, freq, c='blue', label='Actual CDF')
    plt.step(bins_adj, freq_adj, c='purple', label='Adjusted
CDF')
    plt.plot(target_bins,
             target_freq,
             c='red',
             label='Target CDF',
             linestyle = '--')
    plt.legend(prop={'size': 14})
    plt.xlim(0, 255)
    plt.ylim(0, 1)
    plt.xlabel('Intensity values', fontsize = 15)
    plt.ylabel('Cumulative fraction of pixels', fontsize = 15);
```

Adjust the colored image directly.

```
def histogram_adjuster_color(image):
    freq, bins = cumulative_distribution(image)    target_bins = 
np.arange(255)
    target_freq = np.linspace(0, 1, len(target_bins))
interpolation = np.interp(freq, target_freq, target_bins)
    image_eq = img_as_ubyte(interpolation[image].astype(int))
    freq_adj, bins_adj = cumulative_distribution(image_eq)


    fig, axes = plt.subplots(1, 2, figsize=(15,7));
    imshow(image, ax = axes[0]);
    imshow(image_eq, ax = axes[1]);

    axes[0].axis('off')
    axes[1].axis('off')
    axes[0].set_title('Unadjusted Image', fontsize = 17)
    axes[1].set_title('Adjusted Image', fontsize = 17)

    fig, axes = plt.subplots(1, 1, figsize=(19,7));
    plt.step(bins, freq, c='blue', label='Actual CDF')
    plt.step(bins_adj, freq_adj, c='purple', label='Adjusted 
CDF')
    plt.plot(target_bins,
             target_freq,
             c='red',
             label='Target CDF',
             linestyle = '--')

    plt.legend(prop={'size': 15})
    plt.xlim(0, 255)
    plt.ylim(0, 1)
    plt.xlabel('Intensity values', fontsize = 17)
    plt.ylabel('Cumulative fraction of pixels', fontsize = 17);
```

SPRINT 2



Creating CNN model

```
# Part 1 - Building the CNN
#importing the Keras libraries and packages
from keras.models import Sequential
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense, Dropout
from keras import optimizers

# Initialing the CNN
classifier = Sequential()

# Step 1 - Convolution Layer
classifier.add(Convolution2D(32, 3,  3, input_shape = (64, 64, 3), activation = 'relu'))

#step 2 - Pooling
classifier.add(MaxPooling2D(pool_size =(2,2)))

# Adding second convolution layer
classifier.add(Convolution2D(32, 3,  3, activation = 'relu'))
classifier.add(MaxPooling2D(pool_size =(2,2)))

#Adding 3rd Concolution Layer
classifier.add(Convolution2D(64, 3,  3, activation = 'relu'))
classifier.add(MaxPooling2D(pool_size =(2,2)))


#Step 3 - Flattening
classifier.add(Flatten())
```

SPRINT 3

```
#import random
#importcv2
#from keras.preprocessing import image
#import scipy.misc as sm
#from keras.utils import to_categorical
from keras.models import Model
from keras.layers import Dense, GlobalAveragePooling2D
from keras.optimizers import SGD#, Adamfrom
keras.applications.resnet50 importResNet50
from keras.preprocessing.image import
ImageDataGenerator#import numpy asnp
#import os
#from matplotlib importpyplot
#from sklearn.preprocessing importLabelEncoder
# from keras.preprocessing.image import flow_from_directory
#from keras.preprocessing.image import img_to_array
#from sklearn.preprocessing import LabelBinarizer
#from sklearn.model_selection import train_test_split
#import matplotlib.pyplot as plt
#from imutils import paths
#importscipy.misc as sm
#from keras.models import model_from_json


data = ['C:/Users/ankur/.spyder-
py3/autosave/data']labels = []
IMAGE_DIMS = (224,224,3)


print("1")
```

```
"""count=0
ls1=os.listdir('color1')
dic1={}
for idx,i in enumerate(ls1):
    dic1[i]=idx
    ls2=os.listdir('color1/'+i)
    for j in ls2:
  #im1=np.asarray(sm.imread('color/'+i+'/'+j))
  #temp=np.zeros((len(im1),len(im1[0]),len(im1[0][0]) ))
        count=count+1
print(count)
print(dic1)
X=np.zeros((count,224,224,3))
Y=np.zeros((count,1))
vap=0
for idx,i in enumerate(ls1):
    dic1[i]=idx
    ls2=os.listdir('color1/'+i)
    for j in ls2:
        img = image.load_img('color1/'+i+'/'+j, target_size=(224, 224))
        #im1=np.asarray(sm.imread('color1/'+i+'/'+j))
        img = image.img_to_array(img)
        print(img[0])
        print(img.shape)
        #X[vap,:,:,:]=im1
        #Y[vap,0]=idx
        vap=vap+1
"""
```

```
# imagePaths = sorted(list(paths.list_images("color")))#
i=0
# print("2")

# for imagePath in imagePaths:
    # load the image, pre-process it, and store it in the data list #
    img  =  image.load_img(imagePath,target_size=(224,224))  #
    img = img_to_array(img)
    # data.append(img)
    # """im0=np.asarray(image)#
    data[i,:,:,:]=im0"""
    # extract set of class labels from the image path and update the#
    labels list
    # l = label = imagePath.split(os.path.sep)[-2]#
    labels.append(l)



# print("3")
# data = np.array(data, dtype="float") / 255.0

# ltb=labels = np.array(labels)#
print(labels[16])
# lb = LabelBinarizer()
# labels = lb.fit_transform(labels)

"""
```

```
train_labels = os.listdir("color")
le = LabelEncoder()
le.fit([tl for tl in train_labels])
le = LabelEncoder()
le_labels = le.fit_transform(ltb)
"""
# (trainX, testX, trainY, testY) = train_test_split(data,#
            labels, test_size=0.3, random_state=42)


# print("4")
# print(trainX.shape)
""""
ind_train = random.sample(list(range(trainX.shape[0])), 20)
trainX = trainX[ind_train]
trainY = trainY[ind_train]
# test data
ind_test = random.sample(list(range(testX.shape[0])),
5)testX = testX[ind_test]
testY = testY[ind_test]
def resize_data(data):
    data_upscaled = np.zeros((data.shape[0], 320, 320, 3))
    for i, imgin enumerate(data):
        large_img = cv2.resize(img, dsize=(320, 320),
        interpolation=cv2.INTER_CUBIC)data_upscaled[i] = large_img
    return data_upscaled
# resize train and test
datax_train_resized =
resize_data(trainX)x_test_resized =
resize_data(testX) """


# y_train_hot_encoded = to_categorical(trainY)


# y_test_hot_encoded = to_categorical(testY)


"""for i in range(0,len(trainY)):
    print(y_train_hot_encoded[i])
    print("\n")
"""

aug = ImageDataGenerator(rotation_range=25, width_shift_range=0.1,
    height_shift_range=0.1, shear_range=0.2, zoom_range=0.2,
    horizontal_flip=True, fill_mode="nearest")



train_generator=aug.flow_from_directory(
[10:35, 11/8/2022] Irin: directory=r"C:/Users/ankur/.spyder-py3/autosave/data/train",
        target_size=(224,224),
        color_mode="rgb",
        batch_size=64,
        class_mode="categorical",
        shuffle=True,
        seed=None
    )


valid_generator=aug.flow_from_directory(
        directory=r"C:/Users/ankur/.spyder-py3/autosave/data/test",
        target_size=(224,224),
        color_mode="rgb",
        batch_size=64,
        class_mode="categorical",
        shuffle=True,
        seed=None
    )





def model(base_model):


    print("5")
    # get layers and add average pooling layerx
    = base_model.output
    x= GlobalAveragePooling2D()(x)


    #add fully-connected layer
    x= Dense(512, activation='relu')(x)


    # add output layer
    predictions = Dense(7, activation='softmax')(x)
```

```
model = Model(inputs=base_model.input, outputs=predictions)#
fname = "weights.hdf5"
#model.load_weights(fname)
# freeze pre-trained model area's layerfor
layer inbase_model.layers:
    layer.trainable = False


#update the weight that are added
# model.compile(optimizer='rmsprop', loss='categorical_crossentropy')#
model.fit(x_train, y_train,epochs=4)


# choose the layers which are updated by training
layer_num = len(model.layers)
print(layer_num," numberof layers")
for layer in model.layers[:int(layer_num * 0.7)]:
        layer.trainable = False


for layer in model.layers[int(layer_num * 0.7):]:
        layer.trainable = True



#update the weights
model.compile(optimizer=SGD(lr=1e-4,decay=1e-6, momentum=0.9),
loss='categorical_crossentropy', metrics=['accuracy'])
"""history = model.fit_generator(
aug.flow(x_train, y_train,),
validation_data=(testX, testY),
steps_per_epoch=len(trainX),
epochs=5,verbose=1)"""
STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
history=model.fit_generator(generator=train_generator,
                                         steps_per_epoch=STEP_SIZE_TRAI
                                         N,validation_data=valid_generator,
                                         validation_steps=STEP_SIZE_VALI
                                         D, # use_multiprocessing=True,
                                         # workers=3, #
                                         verbose=2,
                                         epochs=100
                                  )

#print(model.evaluate_generator(generator=valid_generator))

model_json = model.to_json()
with open("C:/Users/ankur/.spyder-py3/autosave/model.json", "w") as json_file:
        json_file.write(model_json)
# serialize weights to HDF5
model.save_weights("model.h5")
print("Saved model to disk")
fname="C:/Users/ankur/.spyder-py3/autosave/weights1.hdf5"
model.save_weights(fname,overwrite=True)

#prediction
#img =
image.load_img(r'C:\Users\WASD\Desktop\hoga\color\Pepper,bell_Bacterial_spot\29.jpg',target_size=(224,224))
          #img = image.img_to_array(img)
          #img=np.expand_dims(img,axis=0)
          #predictedclass = model.predict(img)#
          print(train_generator.class_indices)#
          predictedclass


          #for i in train_generator.class_indices:
          #         if train_generator.class_indices[i] == np.argmax(predictedclass):#
          print(i)
          #                  break




          # history = model.fit(x_train, y_train, epochs=7,batch_size=10)i#
          pyplot.plot(history.history['loss'])
          #pyplot.plot(history.history['val_loss'])
          # pyplot.title('model train vs validation loss')#
          pyplot.ylabel('loss')
          # pyplot.xlabel('epoch')
          # pyplot.legend(['train', 'validation'], loc='upper right')#
          pyplot.show# print(model.summary())

          return history
```

SPRINT 4

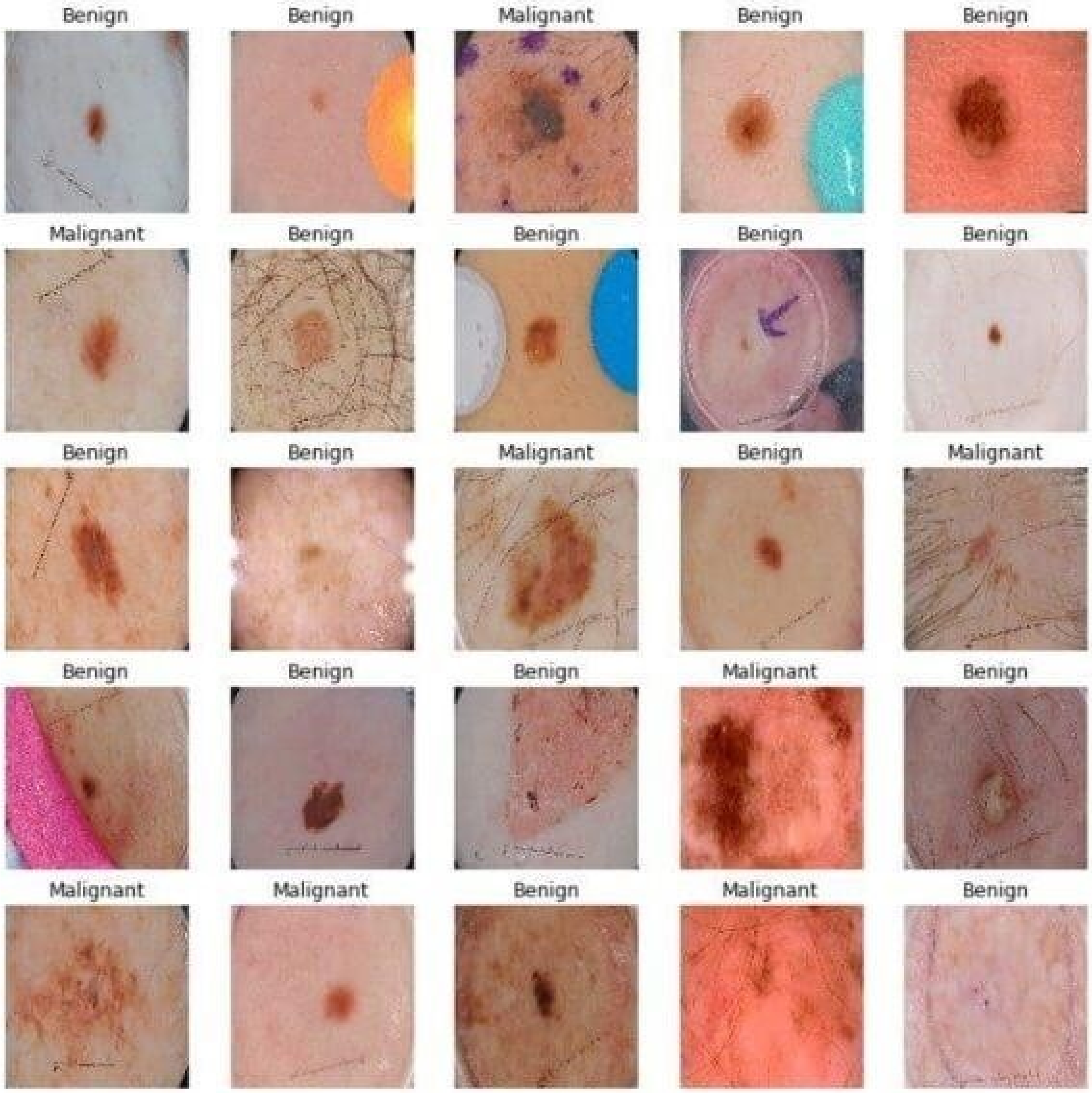
```
import tensorflow as tf import
tensorflow_hub as hub
import matplotlib.pyplot as pltimport numpy
as np
import pandas as pd import
seaborn as sns
from tensorflow.keras.utils import get_file
from sklearn.metrics import roc_curve, auc, confusion_matrix from imblearn.metrics import
sensitivity_score, specificity_score

import os import glob
import zipfileimport
random

# to get consistent resultsafter multiple runstf.random.set_seed(7)
np.random.seed(7)
random.seed(7)

# 0 for benign, 1 for malignant class_names =["benign",
"malignant"]
```

Preparing the Dataset

```
def download_and_extract_dataset():
  # dataset from https://github.com/udacity/dermatologist-ai# 5.3GB
  train_url = "https://s3-us-west-1.amazonaws.com/udacity-dlnfd/datasets/skin-
cancer/train.zip"
  # 824.5MB
  valid_url = "https://s3-us-west-1.amazonaws.com/udacity-dlnfd/datasets/skin-
cancer/valid.zip"


  # 5.1GB
  test_url = "https://s3-us-west-1.amazonaws.com/udacity-dlnfd/datasets/skin-
cancer/test.zip"
  for i, download_link in enumerate([valid_url, train_url, test_url]):temp_file = f"temp{i}.zip"
    data_dir = get_file(origin=download_link,
fname=os.path.join(os.getcwd(), temp_file))
    print("Extracting", download_link)
    with zipfile.ZipFile(data_dir, "r") as z:z.extractall("data")
    # remove the temp file
    os.remove(temp_file)

# comment the below line if you already downloaded the datasetdownload_and_extract_dataset()


# preparing data
# generate CSV metadata file toread img paths and labels from itdef generate_csv(folder,
label2int):
    folder_name = os.path.basename(folder)labels
    =list(label2int)
    # generate CSV file
    df = pd.DataFrame(columns=["filepath", "label"])i = 0
    for label in labels:
        print("Reading", os.path.join(folder, label, "*"))
        for filepath in glob.glob(os.path.join(folder, label, "*")):df.loc[i] = [filepath,
            label2int[label]]
```

```
                    i += 1
        output_file = f"{folder_name}.csv"print("Saving",
        output_file) df.to_csv(output_file)


# generate CSV files for all data portions, labelingnevus andseborrheic keratosis
# as 0 (benign), and melanoma as 1 (malignant)
# you shouldreplace "data" path to your extracted datasetpath
# don't replace if you used download_and_extract_dataset() functiongenerate_csv("data/train",

{"nevus": 0, "seborrheic_keratosis": 0,

"melanoma": 1})

generate_csv("data/valid", {"nevus": 0, "seborrheic_keratosis": 0,
"melanoma": 1})

generate_csv("data/test", {"nevus": 0, "seborrheic_keratosis": 0,
"melanoma": 1})

# loading data train_metadata_filename =
"train.csv"valid_metadata_filename = "valid.csv"# load
CSV files as DataFrames
df_train = pd.read_csv(train_metadata_filename)df_valid =
pd.read_csv(valid_metadata_filename)n_training_samples =
len(df_train) n_validation_samples = len(df_valid)
print("Number of training samples:", n_training_samples) print("Number of validation
samples:", n_validation_samples)
train_ds = tf.data.Dataset.from_tensor_slices((df_train["filepath"],df_train["label"]))
valid_ds = tf.data.Dataset.from_tensor_slices((df_valid["filepath"],df_valid["label"]))
Number of training samples: 2000Number of
validation samples: 150
# preprocess data def
decode_img(img):
    # convert the compressed stringto a 3D uint8 tensorimg =
    tf.image.decode_jpeg(img, channels=3)
    # Use `convert_image_dtype` to convert to floats in the [0,1] range.img =
    tf.image.convert_image_dtype(img, tf.float32)
    # resize the image to the desiredsize.
    return tf.image.resize(img, [299, 299])


def process_path(filepath, label):
    # load the rawdata from the file as a stringimg =
    tf.io.read_file(filepath)
    img = decode_img(img)return
    img, label


valid_ds    =    valid_ds.map(process_path)    train_ds    =
train_ds.map(process_path)# test_ds = test_ds
for image, label in train_ds.take(1): print("Image shape:",
        image.shape)print("Label:", label.numpy())
Image shape: (299, 299, 3)
Label: 0

# training parametersbatch_size
= 64 optimizer = "rmsprop"

def prepare_for_training(ds, cache=True, batch_size=64,
shuffle_buffer_size=1000):
    if cache:
        if isinstance(cache, str):ds =
            ds.cache(cache)
        else:
            ds    =    ds.cache()#
    shufflethe dataset
    ds = ds.shuffle(buffer_size=shuffle_buffer_size)# Repeat forever
    ds = ds.repeat()
    # split to batches
    ds = ds.batch(batch_size)
    # `prefetch` lets the dataset fetch batches in the background whilethe model
    # is training.
    ds = ds.prefetch(buffer_size=tf.data.experimental.AUTOTUNE)return ds

valid_ds = prepare_for_training(valid_ds, batch_size=batch_size,cache="valid-cached-data")
train_ds = prepare_for_training(train_ds, batch_size=batch_size,cache="train-cached-data")
batch = next(iter(valid_ds))


def
    show_batch(batch):plt.figure(figsize=(12,
    12))for n in range(25):
            ax = plt.subplot(5,5,n+1)
            plt.imshow(batch[0][n])
            plt.title(class_names[batch[1][n].numpy()].title())plt.axis('off')

show_batch(batch)
```

Output:



```
# buildingthe model
# InceptionV3 model & pre-trained weights
module_url ="https://tfhub.dev/google/tf2-
preview/inception_v3/feature_vector/4"m =
tf.keras.Sequential([
        hub.KerasLayer(module_url, output_shape=[2048], trainable=False),tf.keras.layers.Dense(1,
        activation="sigmoid")
])
m.build([None, 299, 299, 3])
m.compile(loss="binary_crossentropy", optimizer=optimizer,metrics=["accuracy"])
m.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                    Output Shape                Param #
=================================================================
keras_layer (KerasLayer)         multiple                    21802784
_________________________________________________________________
dense (Dense)                   multiple                    2049
=================================================================
Totalparams: 21,804,833
Trainable params: 2,049
Non-trainable params: 21,802,784
_________________________________________________________________
```

Training the Model

```
model_name = f"benign-vs-malignant_{batch_size}_{optimizer}"
tensorboard = tf.keras.callbacks.TensorBoard(log_dir=os.path.join("logs",model_name))
# saves model checkpont wheneverwe reach betterweights
modelcheckpoint = tf.keras.callbacks.ModelCheckpoint(model_name +"_{val_loss:.3f}.h5",
save_best_only=True, verbose=1)
```

```
history = m.fit(train_ds, validation_data=valid_ds,
                    validation_steps=n_validation_samples // batch_size,
verbose=1, epochs=100,
                    callbacks=[tensorboard, modelcheckpoint])
```

Here is a part of the outputduring training:

```
Train for 31 steps, validate for 2 stepsEpoch 1/100
30/31 [============================>.] - ETA: 9s - loss: 0.4609 -
accuracy: 0.7760

Epoch 00001: val_loss improved from inf to 0.49703, saving model tobenign-vs-
malignant_64_rmsprop_0.497.h5
31/31 [==============================] - 282s 9s/step - loss: 0.4646 -
accuracy: 0.7722 - val_loss: 0.4970 - val_accuracy: 0.8125
<..SNIPED..>
Epoch 27/100
30/31 [============================>.] - ETA: 0s - loss: 0.2982 -
accuracy: 0.8708
Epoch 00027: val_loss improved from 0.40253 to 0.38991, saving model tobenign-vs-
malignant_64_rmsprop_0.390.h5
31/31 [==============================] - 21s 691ms/step - loss: 0.3025
- accuracy: 0.8684 - val_loss: 0.3899 - val_accuracy: 0.8359
<..SNIPED..>
Epoch 41/100
30/31 [============================>.] - ETA: 0s - loss: 0.2800 -
accuracy: 0.8802
Epoch 00041: val_loss did not improve from 0.38991
31/31 [==============================] - 21s 690ms/step - loss: 0.2829
- accuracy: 0.8790 - val_loss: 0.3948 - val_accuracy: 0.8281Epoch 42/100
30/31 [============================>.] - ETA: 0s - loss: 0.2680 -
```

```
accuracy: 0.8859

Epoch 00042: val_loss did not improve from 0.38991

31/31 [==============================] - 21s 693ms/step - loss: 0.2722
- accuracy: 0.8831 - val_loss: 0.4572 - val_accuracy: 0.8047
```

Model Evaluation

```
# evaluation
# load testing set test_metadata_filename =
"test.csv"
df_test = pd.read_csv(test_metadata_filename)n_testing_samples
= len(df_test)
print("Numberof testing samples:", n_testing_samples)
test_ds = tf.data.Dataset.from_tensor_slices((df_test["filepath"],df_test["label"]))


def prepare_for_testing(ds, cache=True, shuffle_buffer_size=1000):

    if cache:
        if isinstance(cache, str):ds =
            ds.cache(cache)
        else:
            ds = ds.cache()
    ds = ds.shuffle(buffer_size=shuffle_buffer_size)return ds


test_ds = test_ds.map(process_path)
test_ds = prepare_for_testing(test_ds, cache="test-cached-data")
```

The above code loads our test data and prepares it for testing:

(299, 299,3)

600

images of the shape

set from tf.datainto a NumPy array:

```
# convert testing set to numpy array to fit in memory (don't do thatwhentesting
# set is too large)
y_test = np.zeros((n_testing_samples,))
X_test = np.zeros((n_testing_samples, 299, 299, 3))
for i, (img, label) in enumerate(test_ds.take(n_testing_samples)):# print(img.shape, label.shape)
    X_test[i] =    img y_test[i] =
    label.numpy()

print("y_test.shape:",    y_test.shape)# load the weights
with the least loss
m.load_weights("benign-vs-malignant_64_rmsprop_0.390.h5")print("Evaluating the model...")
loss, accuracy = m.evaluate(X_test, y_test, verbose=0)print("Loss:", loss, "
Accuracy:", accuracy)
```

Output:

```
Evaluating the model...
Loss:0.4476394319534302                    Accuracy: 0.8
```

The below functiondoes that:

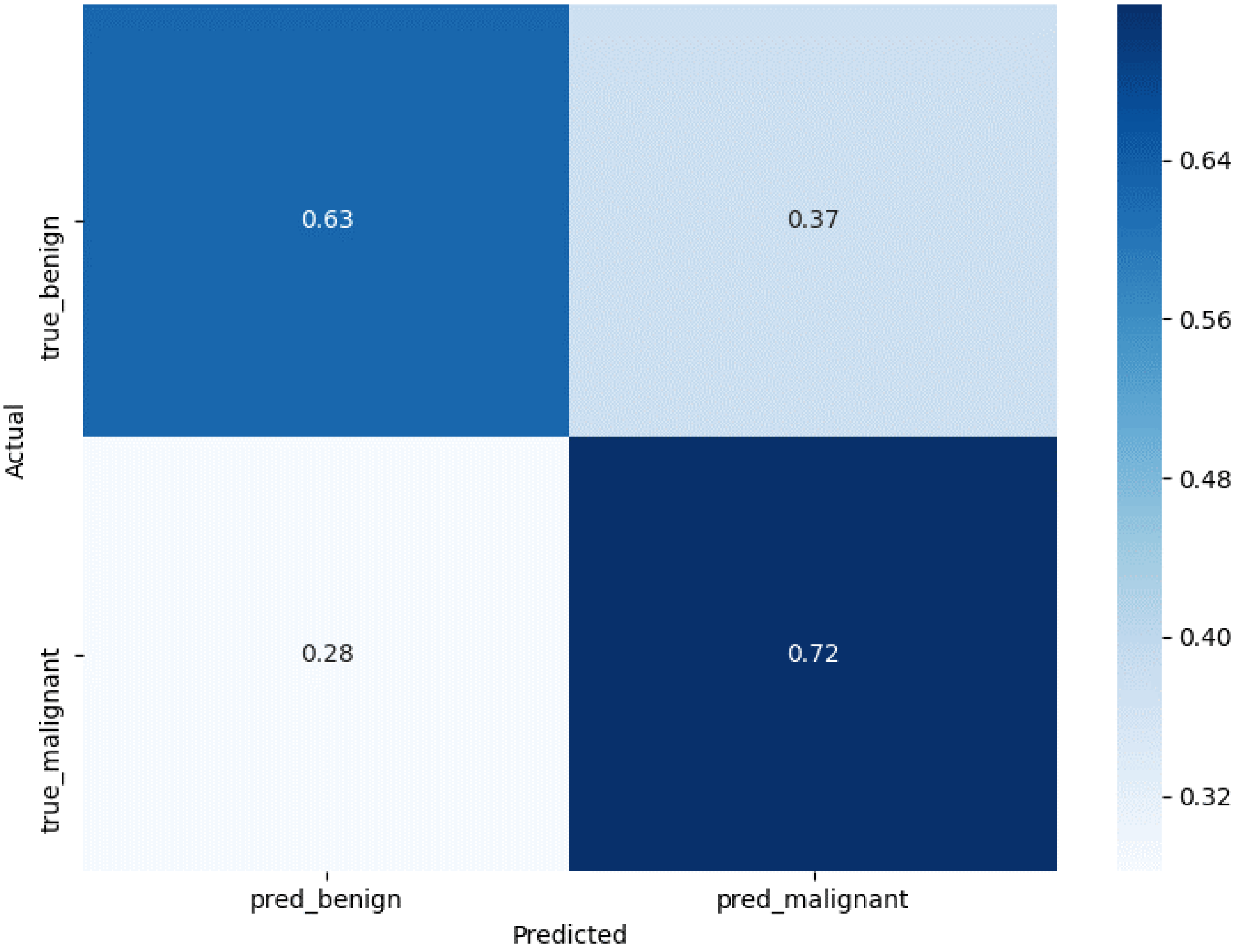
```
def get_predictions(threshold=None):"""
```

```
    Returns predictions for binary classification given `threshold`
    For instance, if threshold is 0.3, then it'll output1 (malignant)for that sample if
    the probability of 1 is 30% or more (insteadof 50%)"""
    y_pred = m.predict(X_test)if not
    threshold:
        threshold = 0.5
    result = np.zeros((n_testing_samples,))for i in
    range(n_testing_samples):
        # test melanomaprobability if y_pred[i][0]
        >= threshold:
            result[i] = 1
        # else, it's 0 (benign)return result

threshold = 0.23
# get predictions with 23% threshold
# which means if the model is 23% sure or more that is malignant,# it's assigned as malignant,
otherwise it's benign
y_pred = get_predictions(threshold)
```

Now let's draw our confusion matrix and interpretit:

```
def plot_confusion_matrix(y_test, y_pred):cmn =
    confusion_matrix(y_test, y_pred)
    # Normalise
    cmn = cmn.astype('float') / cmn.sum(axis=1)[:, np.newaxis]

    # print
    itprint(cmn)
    fig, ax = plt.subplots(figsize=(10,10))sns.heatmap(cmn,
    annot=True, fmt='.2f',
                              xticklabels=[f"pred_{c}" for c in class_names], yticklabels=[f"true_{c}"
```

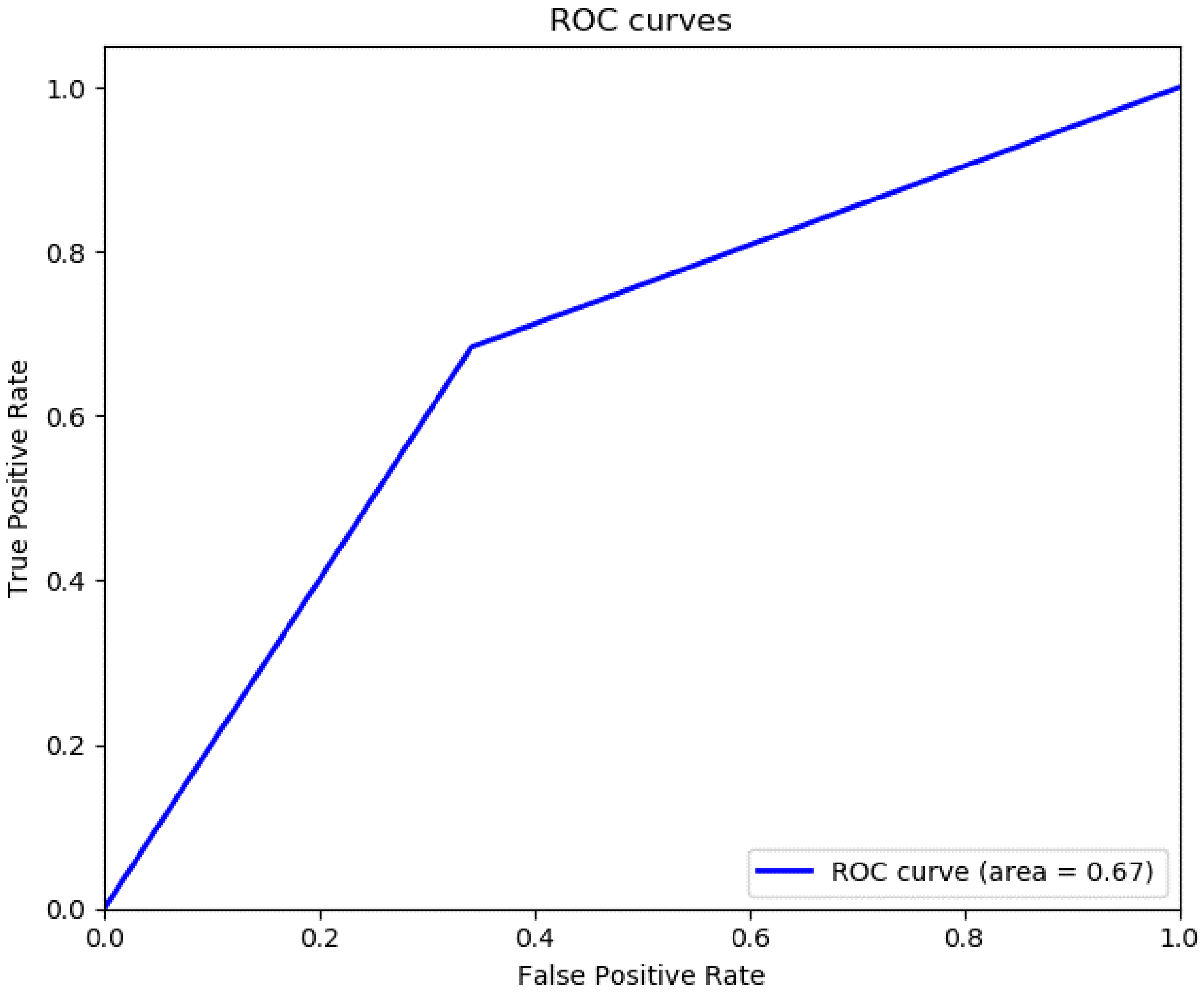
```
                    for c in class_names],cmap="Blues"
                    )
  plt.ylabel('Actual')
  plt.xlabel('Predicted')
  # plot the resulting confusionmatrixplt.show()

plot_confusion_matrix(y_test, y_pred)
```



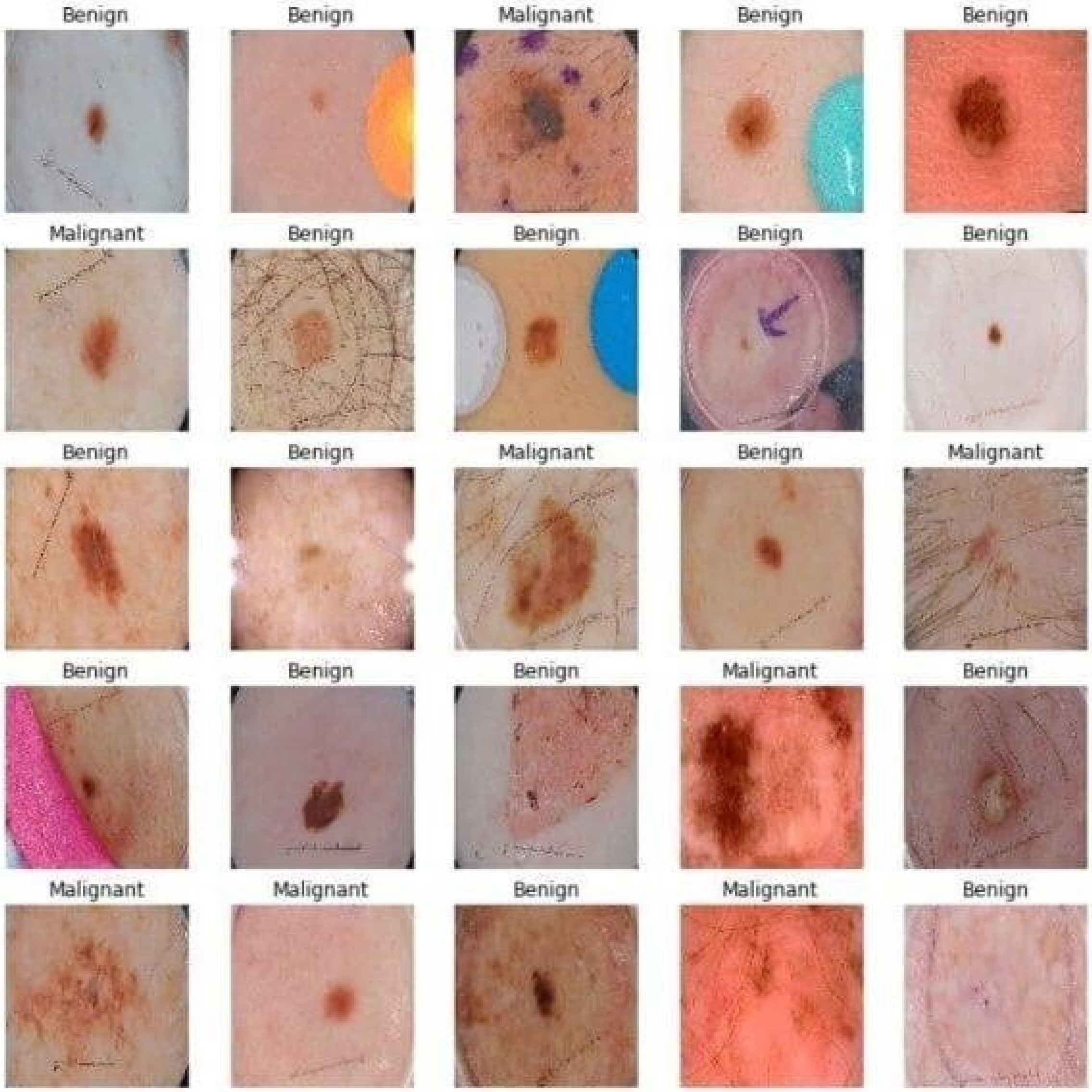
```
def plot_roc_auc(y_true, y_pred):"""
    This function plots the ROC curves and provides the scores."""
    # prepare for figureplt.figure()
    fpr, tpr, _ = roc_curve(y_true, y_pred)# obtain ROC AUC
    roc_auc = auc(fpr, tpr)# print score
```

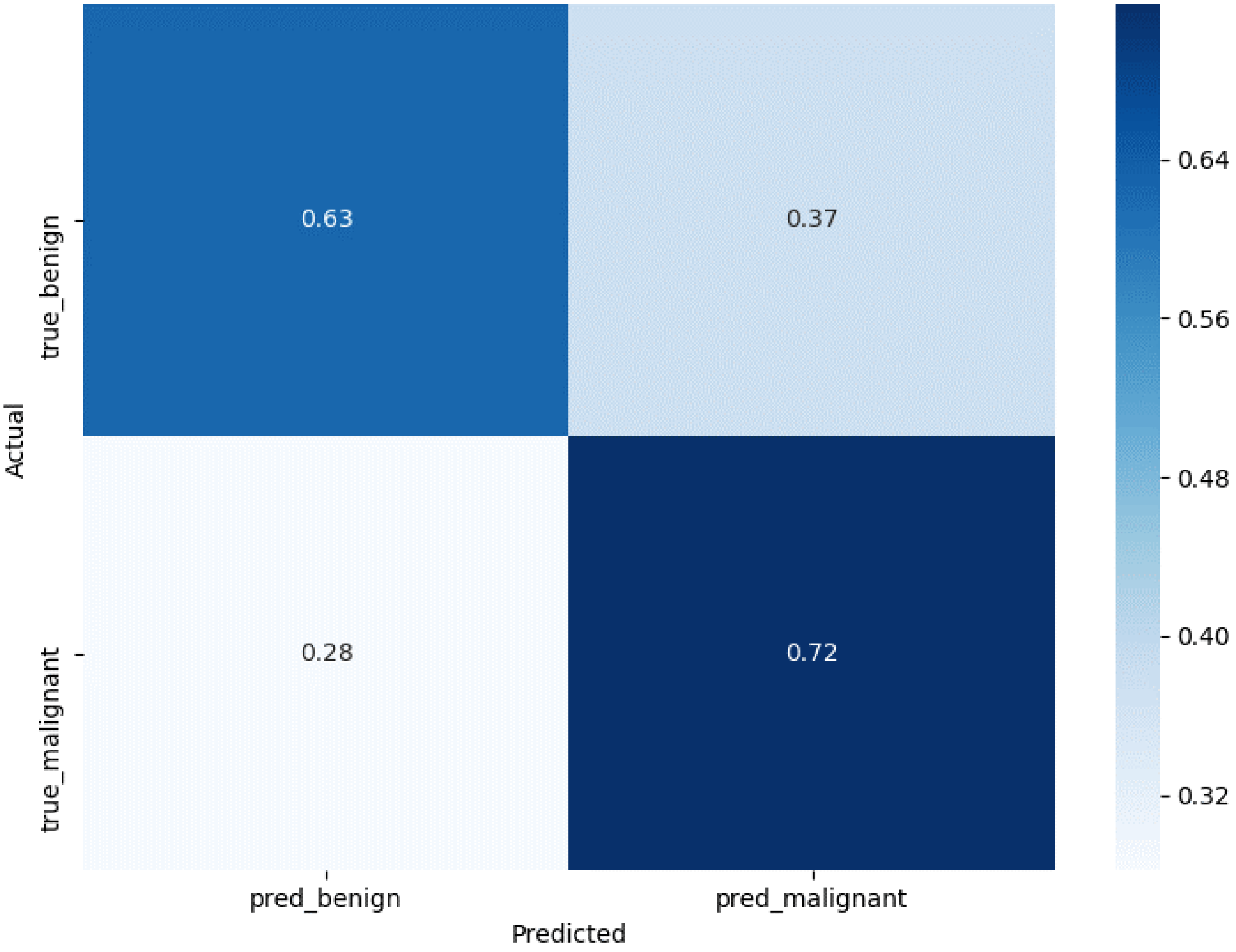
```
        print(f"ROC AUC: {roc_auc:.3f}")# plot ROC
        curve
        plt.plot(fpr, tpr, color="blue", lw=2,
                            label='ROC curve (area = {f:.2f})'.format(d=1,
f=roc_auc))
        plt.xlim([0.0, 1.0])
        plt.ylim([0.0, 1.05]) plt.xlabel('False Positive Rate')
        plt.ylabel('True Positive Rate') plt.title('ROC
        curves') plt.legend(loc="lower right") plt.show()


plot_roc_auc(y_test, y_pred)
```

Output:

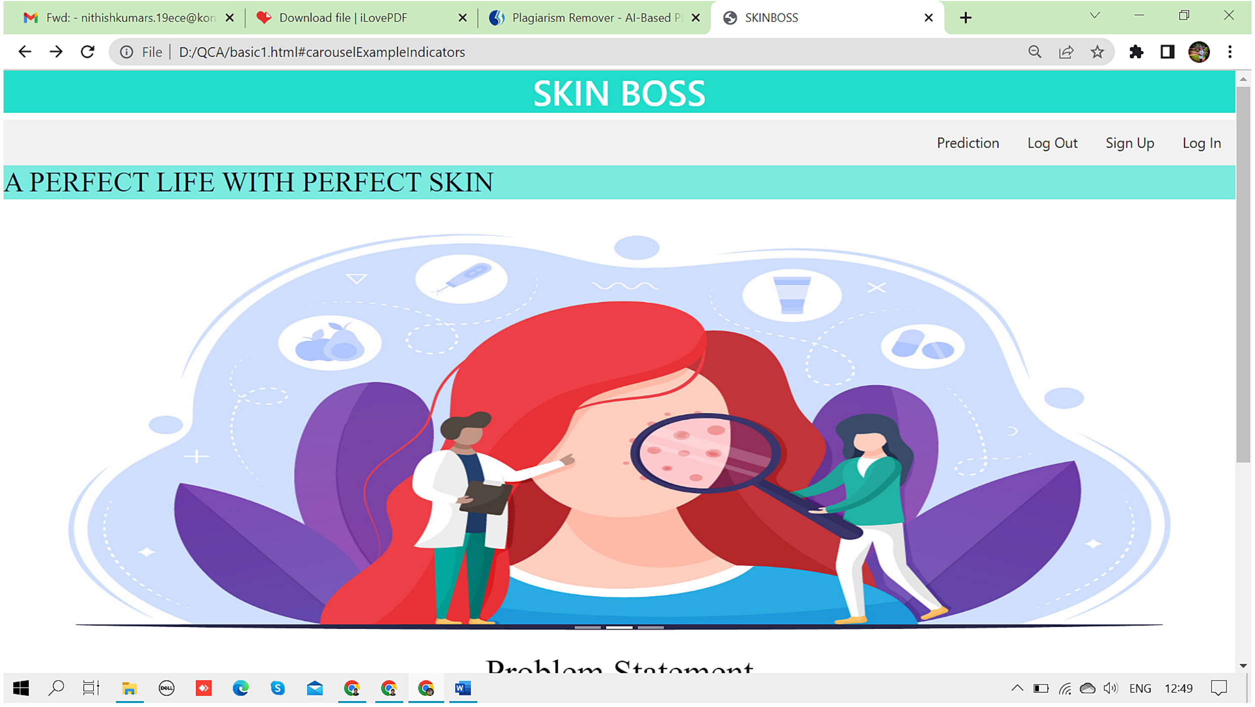

ROC curves
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.67)

ROC AUC: 0.671


Benign
Benign
Malignant
Benign
Benign
Malignant
Benign
Benign
Benign
Benign
Benign
Benign
Malignant
Benign
Malignant
Benign
Benign
Benign
Malignant
Benign
Malignant
Malignant
Benign
Malignant
Benign


0.63
0.37
0.28
0.72
true_benign
true_malignant
Actual
pred_benign
pred_malignant
Predicted
0.64
0.56
0.48
0.40
0.32

APPLICATION BUILDINGS


Fwd: - nithishkumars.19ece@kon
Download file | iLovePDF
Plagiarism Remover - AI-Based P
SKINBOSS
File | D:/QCA/basic1.html#carouselExampleIndicators
ENG
12:49

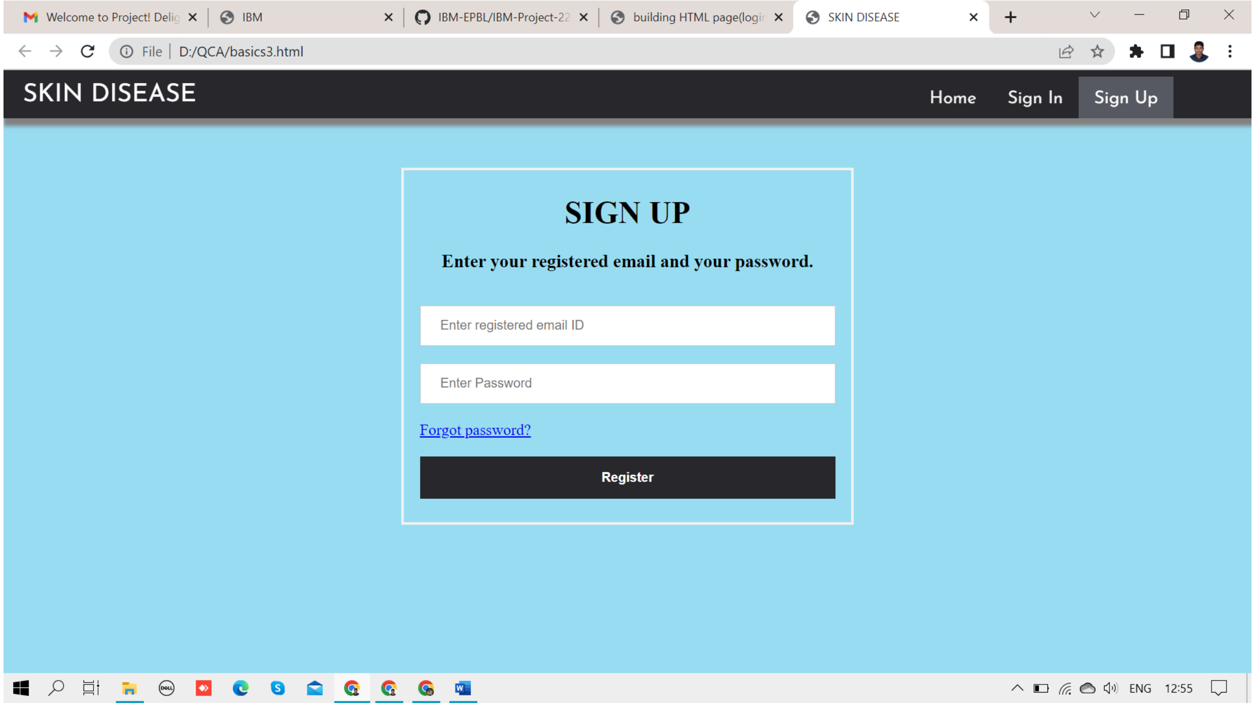
SKIN BOSS

Prediction Log Out Sign Up Log In

A PERFECT LIFE WITH PERFECT SKIN

Problem Statement

SIGNUP PAGE:



SKIN DISEASE

Home Sign In Sign Up

SIGN UP

Enter your registered email and your password.

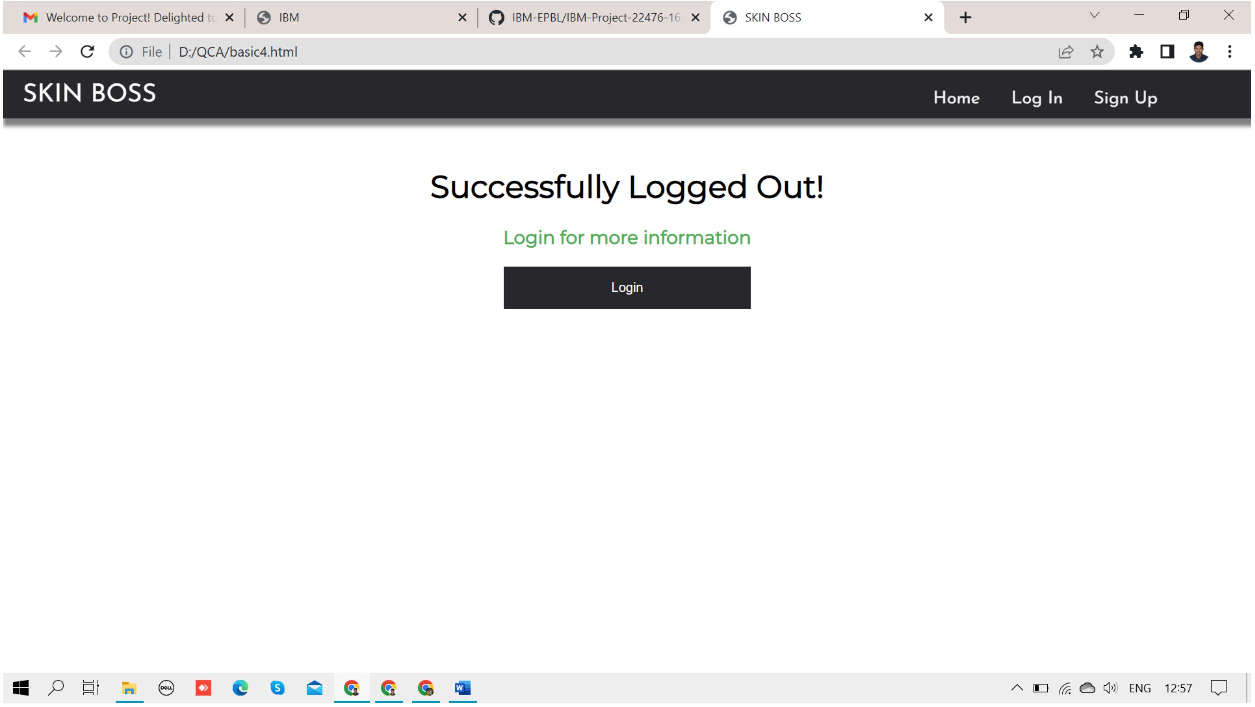
Enter registered email ID

Enter Password

Forgot password?

Register

LOGOUT PAGE:



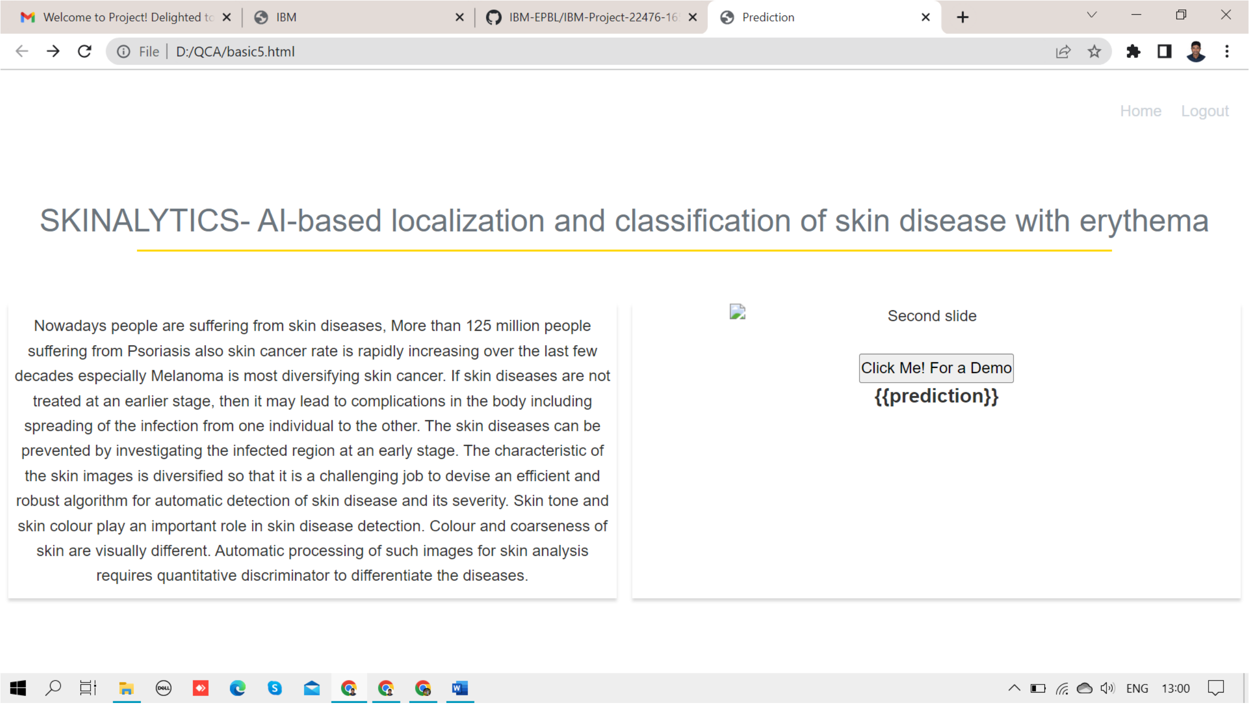
Welcome to Project! Delighted to
IBM
IBM-EPBL/IBM-Project-22476-16
SKIN BOSS
File | D:/QCA/basic4.html
SKIN BOSS
Home
Log In
Sign Up
ENG
12:57

Successfully Logged Out!

Login for more information

Login

HOME PAGE:


Welcome to Project! Delighted to
IBM
IBM-EPBL/IBM-Project-22476-16
Prediction
File | D:/QCA/basic5.html
Home
Logout
ENG
13:00

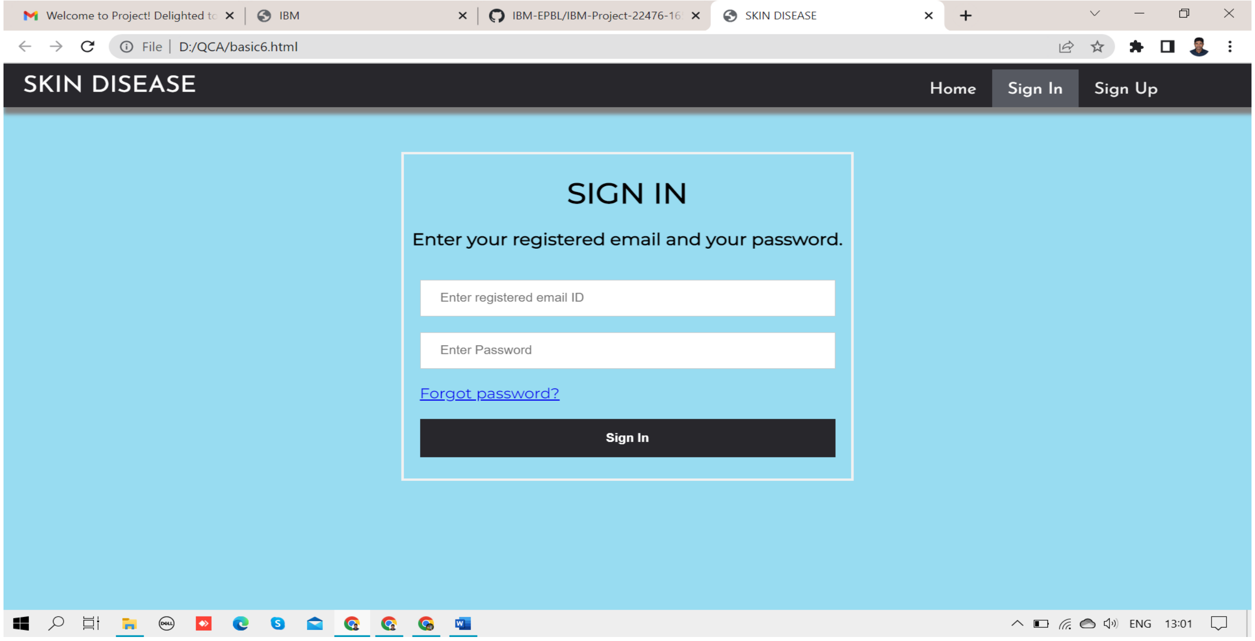
SKINALYTICS- AI-based localization and classification of skin disease with erythema

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.

Second slide

{{prediction}}

SIGNIN PAGE


Welcome to Project! Delighted to
IBM
IBM-EPBL/IBM-Project-22476-16
SKIN DISEASE
File | D:/QCA/basic6.html
SKIN DISEASE
Home
Sign In
Sign Up
ENG
13:01

SIGN IN

Enter your registered email and your password.

Enter registered email ID

Enter Password

Forgot password?

CONCLUSION:

We have shown that sufficient accuracy can be achieved without large datasets and high-quality images. Furthermore, we have shown that current state-of-the-art CNN models outperform models produced by previous studies through appropriate data pre-processing, self-supervised learning, transfer learning, and special CNN architectural techniques. rice field. In addition, precise segmentation provides insight into disease location. This is useful when pre-processing data for classification, as it allows the CNN model to focus on regions of interest. Finally, unlike previous studies, our method provides a solution for classifying multiple diseases in a single image. With higher quality and larger amounts of data, state-of-the-art models will enable the use of CAD in the field of dermatology.