

AI-based localization and classification of skin disease with erythema

A Project report

Submitted by

SAMVARTHINI .C	(813819104081)
ROSHAN AKTHAR.S	(813819104078)
SABARI.P	(813819104079)
SUVEDHA .M	(813819104107)

Project Report Format

1. INTRODUCTION

1.1 Project Overview

1.2 Purpose

2. LITERATURE SURVEY

2.1 Existing problem

2.2 References

2.3 Problem Statement Definition

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

3.2 Ideation & Brainstorming

3.3 Proposed Solution

3.4 Problem Solution fit

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

4.2 Non-Functional requirements

5. PROJECT DESIGN

5.1 Data Flow Diagrams

5.2 Solution & Technical Architecture

5.3 User Stories

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

6.2 Sprint Delivery Schedule

6.3 Reports from JIRA

7. CODING & SOLUTIONING (Explain the features added in the project along with code)

7.1 Feature 1

7.2 Feature 2

7.3 Database Schema (if Applicable)

8. TESTING

8.1 Test Cases

8.2 User Acceptance Testing

9. RESULTS

9.1 Performance Metrics

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

Source Code

GitHub & Project Demo Link

1.INTRODUCTION:

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

1.1 Project Overview

To overcome the above problem we are building a model which is used for the prevention and early detection of skin cancer, psoriasis. Basically, skin disease diagnosis depends on the different characteristics like colour, shape, texture etc. Here the person can capture the images of skin and then the image will be sent the trained model. The model analyses the image and detect whether the person is having skin disease or not.

1.2 PURPOSE

A skin disease detection and classification system is a system used for **detecting whether a disease is present or not, and then classifying the type of disease, if present**. The classification is based on decisions taken using the features extracted through the feature extraction methods

2.LITERATURE SURVEY

2.1 EXISTING PROBLEM

Now a day's people are suffering from skin diseases, More than 125 million people suffering from Psoriasis also skin cancer rate is rapidly increasing over the last few decades especially Melanoma is most diversifying skin cancer. If skin diseases are not treated at an earlier stage, then it may lead to complications in the body including spreading of the infection from one individual to the other. The skin diseases can be prevented by investigating the infected region at an early stage. The characteristic of the skin images is diversified so that it is a challenging job to devise an efficient and robust algorithm for automatic detection of skin

disease and its severity. Skin tone and skin colour play an important role in skin disease detection. Colour and coarseness of skin are visually different. Automatic processing of such images for skin analysis requires quantitative discriminator to differentiate the diseases.

To overcome the above problem we are building a model which is used for the prevention and early detection of skin cancer, psoriasis. Basically, skin disease diagnosis depends on the different characteristics like colour, shape, texture etc. Here the person can capture the images of skin and then the image will be sent the trained model. The model analyses the image and detect whether the person is having skin disease or not.

2.2 REFERENCES

- [1] Arifin, S., Kibria, G., Firoze, A., Amini, A., & Yan, H. (2012) “Dermatological Disease Diagnosis Using Color-Skin Images.” Xian: International Conference on Machine Learning and Cybernetics. W.-K. Chen, Linear Networks and Systems (Book style). Belmont, CA: Wadsworth, 1993, pp. 123–135.
- [2] Nawal Soliman ALKolifi ALEnezi “A method of skin disease detection using image processing and machine learning” at 16TH International learning & Technology conference 2019
- [3] Pravin S. Ambed, A S Shirsat “A image analysis system to detect skin diseases” at IOSR Journal of VLSI and signal processing Volume 6 Issue 5 Ver 1 e-ISSN 2013-4200
- [4] Li-Sheng Wei, Quan Gan and Tao ji “Skin Disease Recognition Method Based on Image Color and texture features”Y. Computational and Mathematical Methods in Medicine Volume 2018, Article ID 8145713, 10 pages <https://doi.org/10.1155/2018/8145713>
- [5] R. Sumithra, M. Suhilb, and D. S. Guruc, “Segmentation and classification of skin lesions for disease diagnosis,” Procedia Computer Science, vol. 45, pp. 76–85, 2015.

2.3 PROBLEM STATEMENT AND DEFINITION

Now day’s skin diseases become more common problem in human life. Most of these diseases are dangerous and harmful, particularly if not treated at an initial stage. People do not treat skin diseases seriously. Sometimes, most of the people treat these infections of the skin using their own household methods. However, if these household treatments are not suitable for that particular skin problem then it would affect the skin. Also they may not be

aware of severe problem of skin diseases. Skin diseases have tendency to pass from one person to another person easily. Hence it is very important to control it at earlier stage to prevent it from spreading in people. The damage done to the skin due to skin diseases also could damage the self-confidence, mental confidence as well as wellbeing of people. Therefore the skin diseases are become a huge problem among people. It has become an important thing to treat these skin diseases properly at the earlier stages itself to prevent serious damage to skin. This system would help to solve this problem to a great extent. Since it system would allow users to determine the skin diseases to provide treatments or advice to patient by making use of images of skin infected with the disease and by obtaining information from the patient.

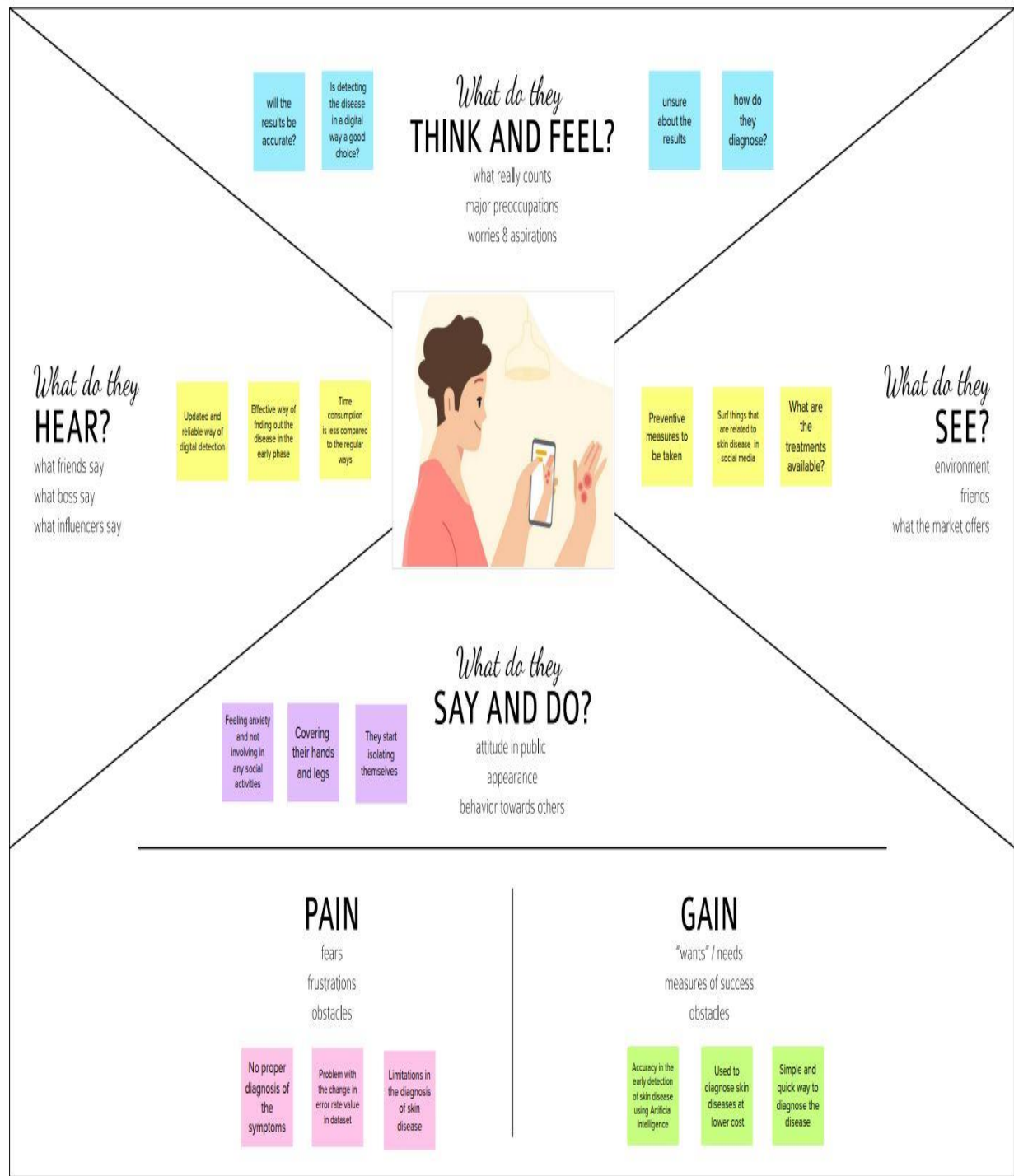
3. IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS

An **empathy map** is a collaborative visualization used to articulate what we know about a particular type of user. It externalizes knowledge about users in order to

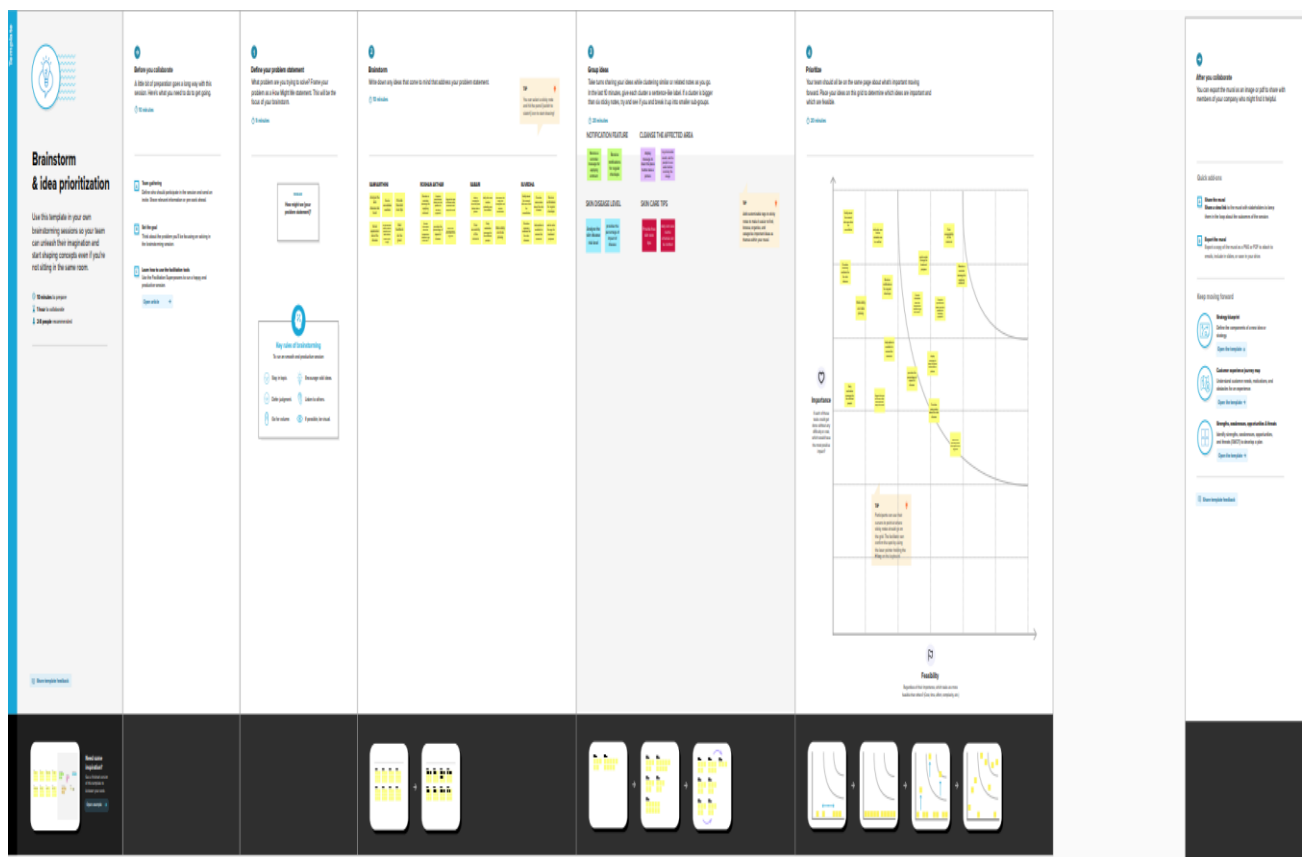
- 1) create a shared understanding of user needs, and
- 2) aid in decision making.

AI-BASED LOCALIZATION AND CLASSIFICATION OF SKIN DISEASE WITH ERYTHEMA



3.2 IDEATION & BRAINSTORMING

Ideation is often closely related to the practice of brainstorming, a specific technique that is utilized to generate new ideas. A principal difference between ideation and brainstorming is that **ideation is commonly more thought of as being an individual pursuit, while brainstorming is almost always a group activity**



3.3 PROPOSED SOLUTION

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	People suffering from Psoriasis also skin cancer rate is rapidly increasing over the last few decades. If skin diseases are not treated at an earlier stage, then it may lead to complications in the body including spreading of the infection. .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.
2.	Idea / Solution description	We are building a model which is used for the prevention and early detection of skin cancer, psoriasis. Basically, skin disease diagnosis depends on the different characteristics like colour, shape, texture etc. Here the person can capture the images of skin and then the image will be sent the trained model. The model analyses the image and detect whether the person is having skin disease or not.
3.	Novelty / Uniqueness	Easy access to all patients with social security during the test period. Diagnosing not just suspicious skin lesions for cancer risk, but hundreds of dermatology conditions.
4.	Social Impact / Customer Satisfaction	Effective way of finding out the disease in the early phase. Updated and reliable way of digital detection.
5.	Business Model (Revenue Model)	Use to identify potential for skin cancer recommend nearby dermatologists who can diagnose the patient properly. Can make revenue upon each successful booking placed by a patient. Users must subscribe with a monthly charge to use the application.
6.	Scalability of the Solution	Accuracy in the early detection of skin disease using Artificial Intelligence. Time consumption is less compared to the regular ways.

3.4 PROBLEM SOLUTION FIT



4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Processing image	Uses deep learning for yolo model and convolutional neural network Recognizes objects in an image
FR-4	Access cloud services	Store the details in the cloudant database Retrieve the needed information for user

4.2 NON-FUNCTIONAL REQUIREMENT

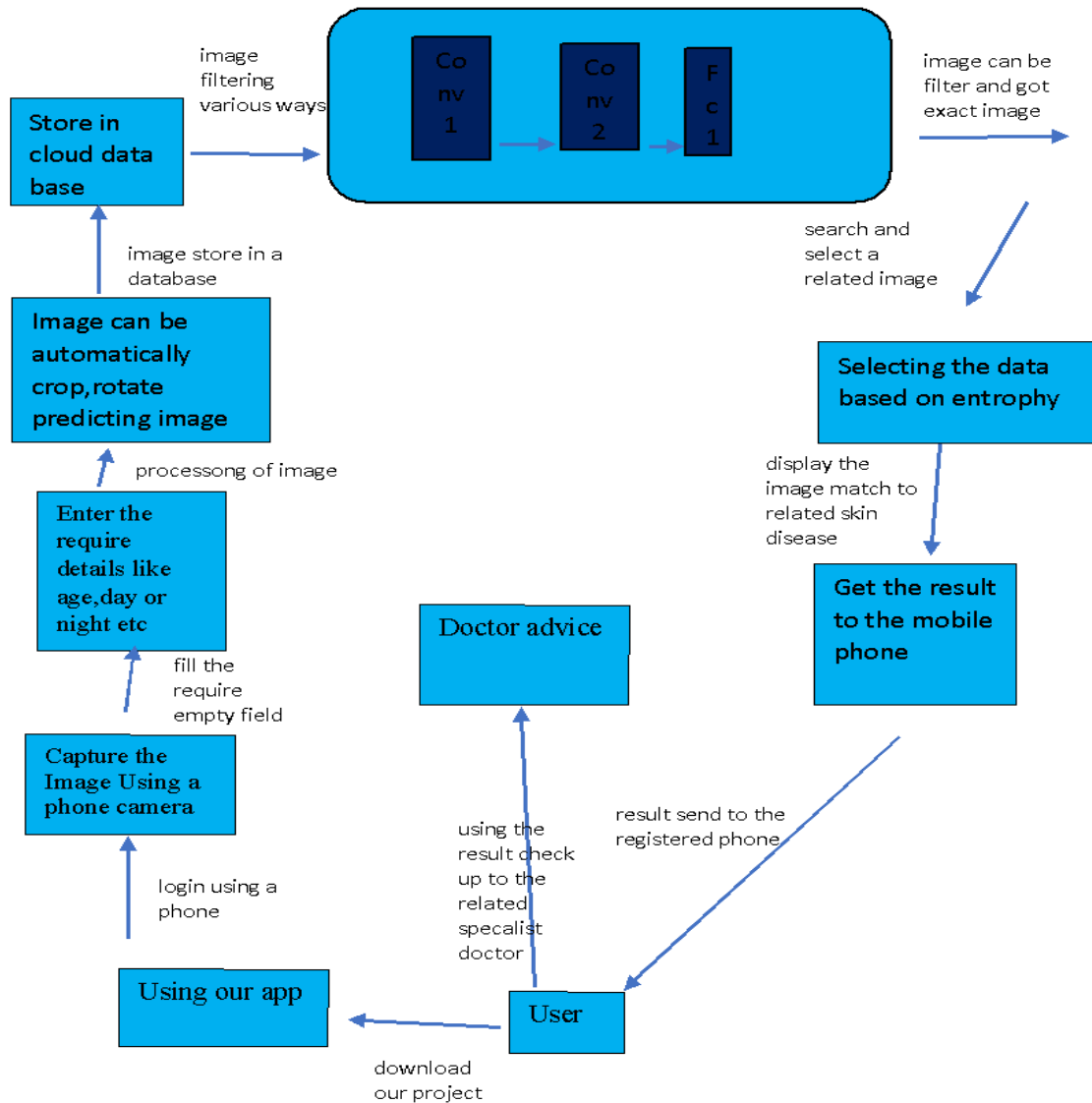
FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Application can be used by anyone who has knowledge about internet and computer
NFR-2	Security	For security knowledge factor authentication is enabled for user safety
NFR-3	Reliability	Highly reliable since it uses trusted cloud services like IBM
NFR-4	Performance	Performance is better when compared with other applications and market products
NFR-5	Availability	Available on all devices and can be downloaded from play store
NFR-6	Scalability	Using cloud services makes the scalability a higher one

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

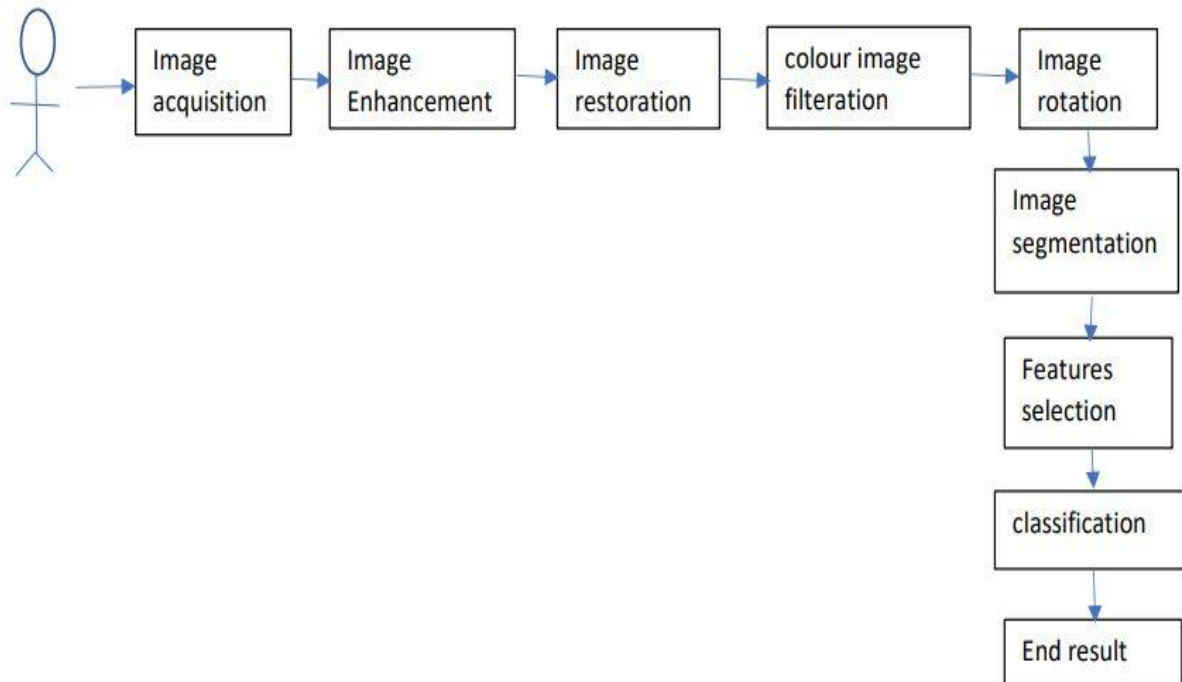
graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

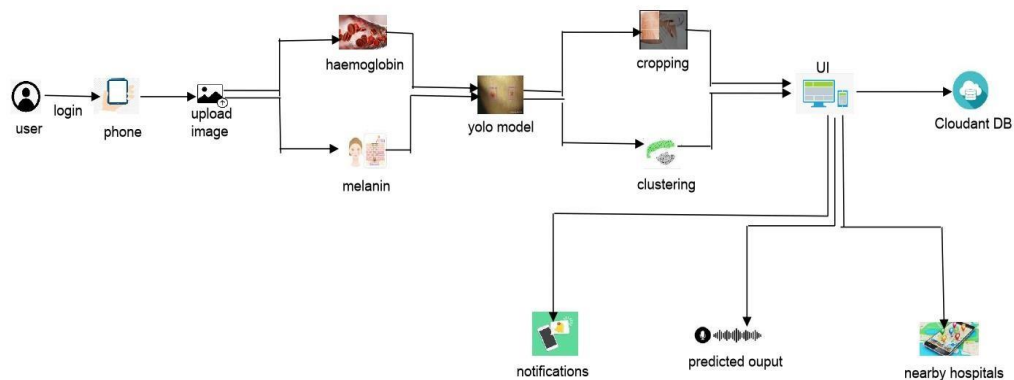


5.2 Solution & Technical Architecture

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to: • Find the best tech solution to solve existing business problems.

- Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed, and delivered





5.3 User Stories

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.	5	High	samvarthini
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application.	1	High	suvetha
Sprint-1		USN-3	As a user, I can take a photo and upload in a Application.	5	High	sabari
Sprint-1		USN-4	As a user, I can receive a result of image I uploaded.	2	Medium	Roshan akthar
Sprint-1		USN-5	As a user, I can consult a doctor according to the result I receive.	7	High	suvetha
Sprint-2	Dashboard	USN-6	As a user, I can enter the age and professional	2	Medium	Sabari

Sprint-2		USN-7	As a user, I can upload a photo also from a gallery and drive	3	Low	Roshan akthar
Sprint-2		USN-8	As a user, I can re upload a photo whether it show a some error	15	Medium	samvarthini
Sprint-3		USN-9	As a user, I can receive a message notification for enabled a notification button	15	Low	suvetha
Sprint-3		USN-10	As a user, I can download the result of application for the uploaded image by using a download button	5	High	samvarthini
Sprint-4	Admin	USN-11	As a admin,I can maintain data of the user safely	2	High	Roshan akthar
Sprint-4		USN-12	As a admin,i can manage a image and updation of disease for train a app regularly	18	Medium	sabari

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

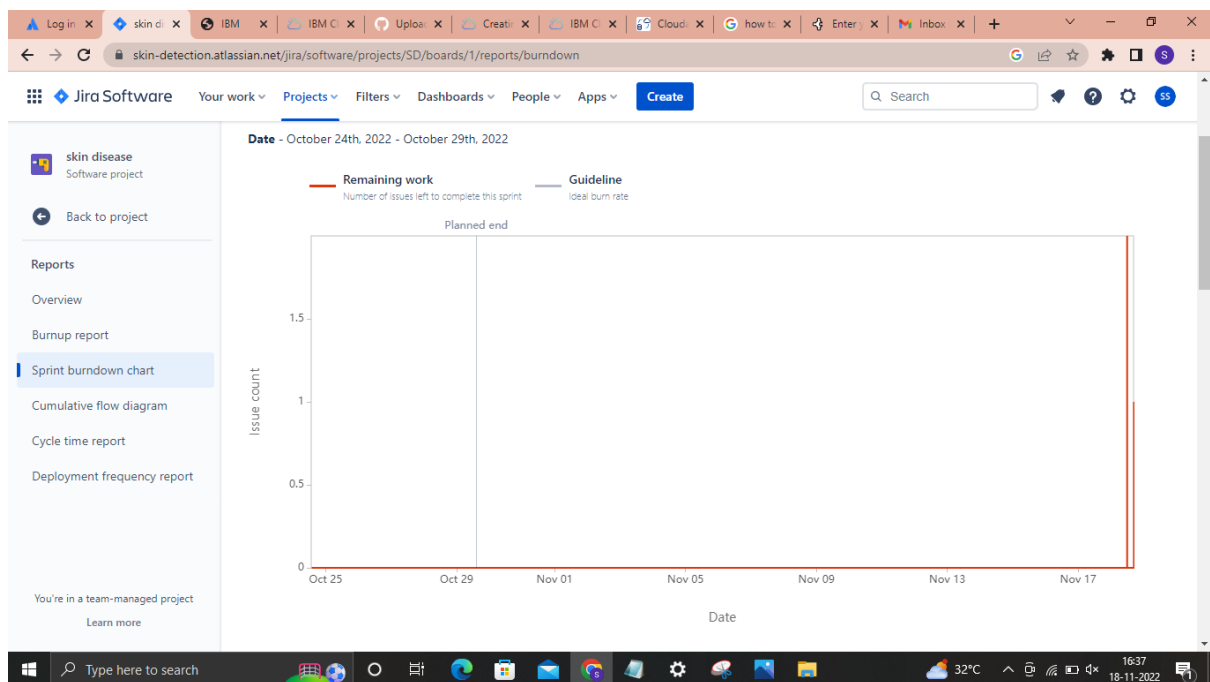
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	5	High	samvarthini
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application.	1	High	suvetha
Sprint-1		USN-3	As a user, I can take a photo and upload in a Application.	5	High	sabari
Sprint-1		USN-4	As a user, I can receive a result of image I uploaded.	2	Medium	Roshan akthar
Sprint-1		USN-5	As a user, I can consult a doctor according to the result I receive.	7	High	suvetha
Sprint-2	Dashboard	USN-6	As a user, I can enter the age and professional	2	Medium	Sabari
Sprint-2		USN-7	As a user, I can upload a photo also from a gallery and drive	3	Low	Roshan akthar
Sprint-2		USN-8	As a user, I can re upload a photo whether it show a some error	15	Medium	samvarthini
Sprint-3		USN-9	As a user, I can receive a message notification for enabled a notification button	15	Low	suvetha
Sprint-3		USN-10	As a user, I can download the result of application for the uploaded image by using a download button	5	High	samvarthini
Sprint-4	Admin	USN-11	As a admin,I can maintain data of the user safely	2	High	Roshan akthar
Sprint-4		USN-12	As a admin,i can manage a image and updation of disease for train a app regularly	18	Medium	sabari

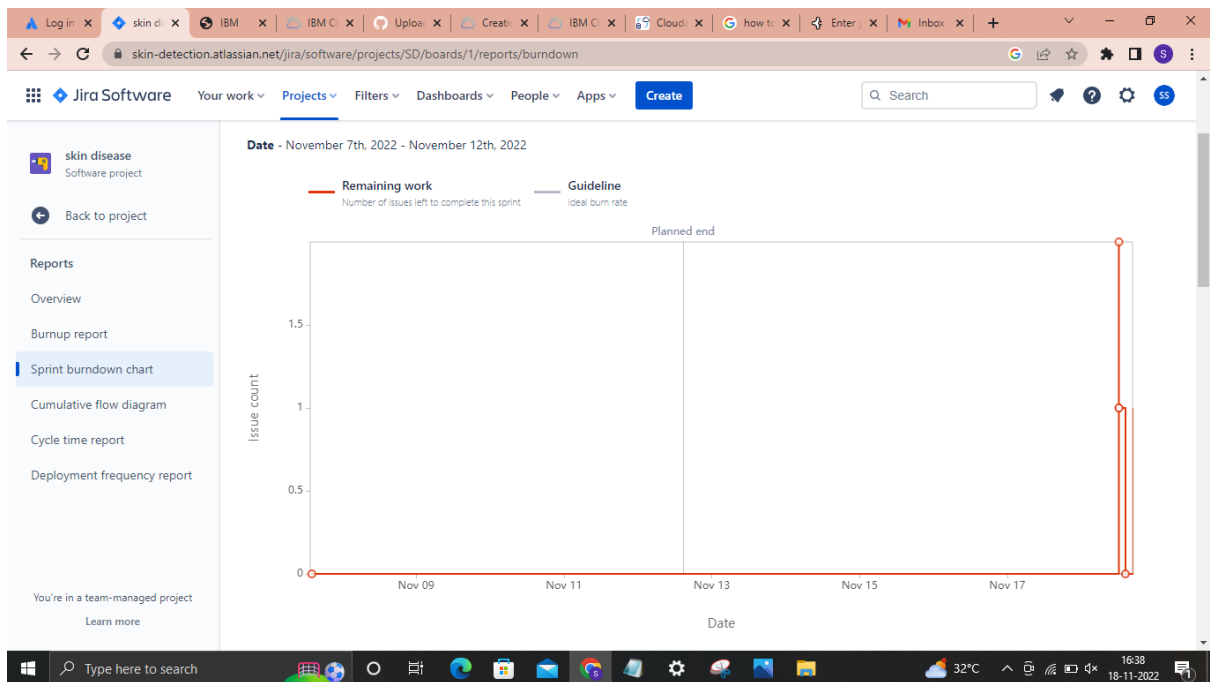
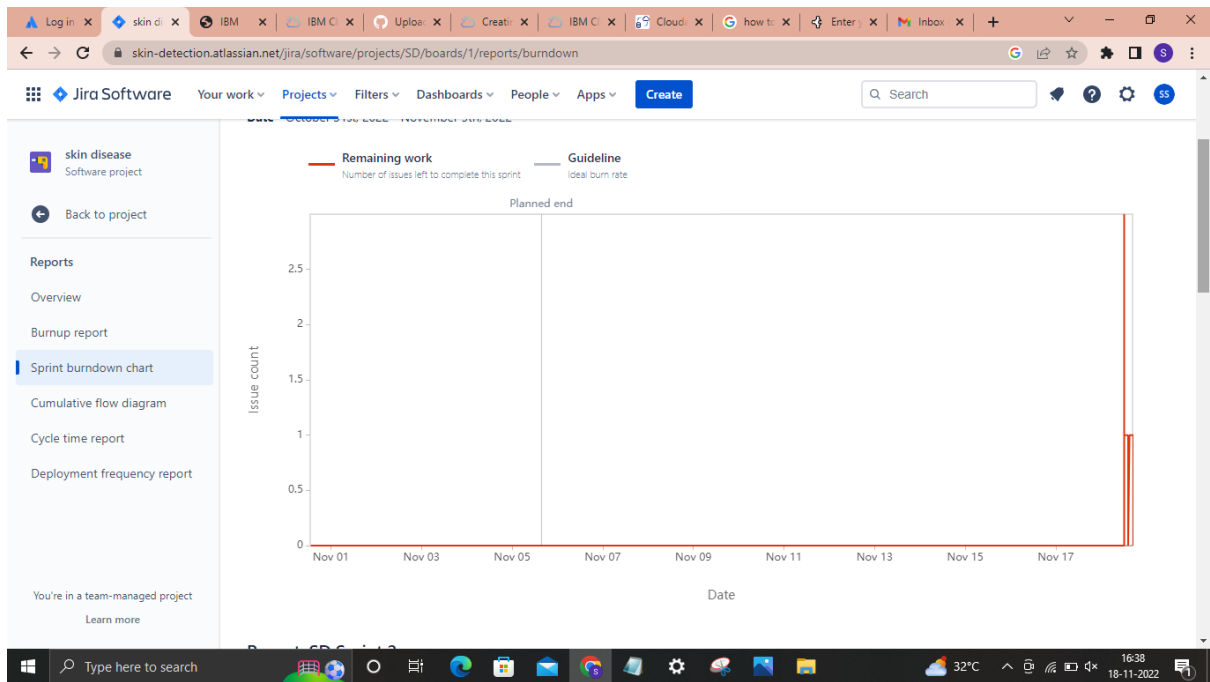
6.2 Sprint Delivery Schedule

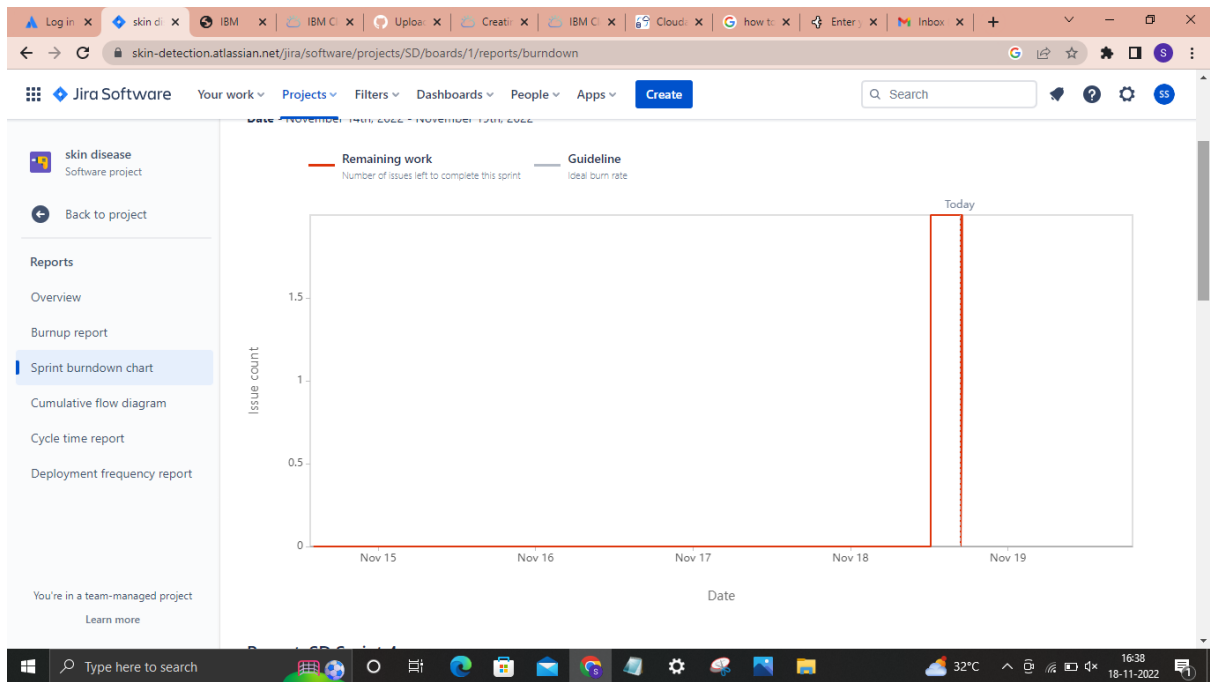
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	29 Nov2022
Sprint-3	20	6 Days	07 Nov 2022	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

Backlog-sprint1







7. CODING & SOLUTIONING (Explain the features added in the project along with code)

7.1 Feature 1

Before going to upload a image.A user can enter the age for to get a accurate result of disease and select a image to upload

The screenshot displays the RASKIN web application's Sign Up page. The header is dark blue with the RASKIN logo on the left and navigation links (Home, Sign In, Sign Up) on the right. The main content area has a light blue background. A white box in the center contains the 'SIGN UP' form. The form includes a title 'SIGN UP', a subtitle 'Enter your name email and password.', and three input fields: a text field for the name (containing 'Rakesh'), an email field (containing 'xyz@gmail.com'), and a password field (containing three dots). Below the input fields is a dark blue 'Register' button. At the bottom of the form, there is a link: 'Already have an account? [Sign In](#)'. The footer is dark blue with the RASKIN logo on the left and 'Home Logout' links on the right.

RASKIN- 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.



[Click Me! For a Demo](#)



7.2 Feature 2

we have a contact page by user this page the use can put a queries to the admin and send feedback. In case any doubt the use can contact the admin by using this page

RASKIN

Sign InSign Up

A PERFECT LIFE WITH PERFECT SKIN



ABOUT PROJECT

Problem:

Nowadays people are suffering from skin diseases. More than 150 million people suffering from Psoriasis who 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 contours of skin are visually different. Automatic processing of such images for skin analysis requires quantitative discriminator to differentiate the diseases.

Solution:



To overcome this problem we are building a model which is used for the prevention and early detection of skin cancer, psoriasis. Basically, skin disease diagnosis depends on the different characteristics like colour, shape, texture etc. Here the person can capture the images of skin and then the image will be sent the trained model. The model analyses the image and detect whether the person is having skin disease or not

My Drive - Google Drive | (no subject) - suve611@gmail.com | IBM | IBM-16815-1662569832 | RASKIN

C:/Users/ELCOT/Desktop/yolo_structure/templates/contact.html

Google | Gmail | YouTube | Maps

Home | Sign In | Sign Up

Contact Info
 cse221107@saranathan.ac.in
 8825600826

Send a Message

First NameLast Name

Email AddressMobile Number

Write your message here...

Send

Type here to search | 15:09 19-11-2022

7.3 Database Schema (if Applicable)

← → ↻

2afd311a-94e6-4743-a958-641316eb4d7e-bluemix.cloudant.com/dashboard.html

🔗 ☆ ⚙️ 📄 👤 ⋮

↔️ Databases

Database name ▾

Create Database

{ } JSON

📄

🔔

Your Databases

Name	Size	# of Docs	Partitioned	Actions
my_database	0 bytes	0	No	<div>↔️ 🔒 📄</div>

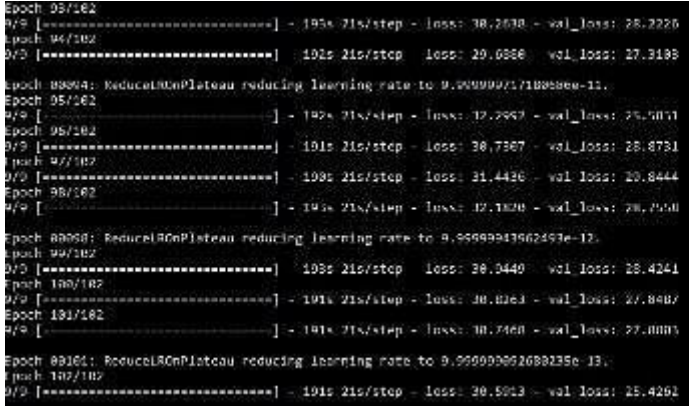
Log Out

Showing 1-1 of 1 databases. Databases per page 20 ▾

« 1 »

8. TESTING

8.1 Test Cases

S. No.	Parameter	Values	Screenshot
1	Model Summary	<p>Found 5 input labels: ['ErythemaAbigne', 'ErythemaMarginatum', 'ErythemaMultiforme', 'ErythemaInfectiosum', 'ErythemaMigrans'] ...</p> <p>Found 14 input images: ['ab2.jpg', 'ab3.jpg', 'ErythemaMarginatum4.jpg', 'Erythemamultiforme11.jpg', 'Erythemamultiforme12.jpg'] ...</p> <p>(416, 416, 3)</p> <p>Found 0 boxes for img</p> <p>Time spent: 11.708sec</p> <p>-34.44164674096789</p> <p>90.90276929854582</p> <p>(416, 416, 3)</p> <p>Found 0 boxes for img</p> <p>Time spent: 4.049sec</p> <p>-7.573041683860296</p> <p>24.676842322523612</p> <p>(416, 416, 3)</p> <p>Found 1 boxes for img</p> <p>ErythemaInfectiosum</p> <p>0.28 (81, 0)</p> <p>(145, 187)</p> <p>Time spent: 4.286sec</p> <p>54.23432382513661</p> <p>45.666406395306694</p> <p>(416, 416, 3)</p>	 

		<p>Found 1 boxes for img ErythemaMigrans 0.36 (58, 61) (170, 163) Time spent: 4.078sec 36.53257619211057 100.37478430619085</p> <p>(416, 416, 3) Found 0 boxes for img Time spent: 4.163sec 22.11471165806249 46.31868468927695 Processed 14 images in 77.3sec - 0.2FPS</p>	
2	Accuracy	<p>Training Accuracy: 70%</p> <p>Validation Accuracy: 75%</p>	<pre> Epoch 93/102 9/9 [=====] - 153s 21s/step - loss: 30.2638 - val_loss: 25.2226 Epoch 94/102 9/9 [=====] - 152s 21s/step - loss: 29.6888 - val_loss: 27.3188 Epoch 95/102: ReduceLROnPlateau reducing learning rate to 9.99999991188888e-11. Epoch 95/102 9/9 [=====] - 152s 21s/step - loss: 31.2982 - val_loss: 25.5031 Epoch 96/102 9/9 [=====] - 151s 21s/step - loss: 30.7307 - val_loss: 25.8731 Epoch 97/102 9/9 [=====] - 150s 21s/step - loss: 31.4436 - val_loss: 26.8444 Epoch 98/102 9/9 [=====] - 153s 21s/step - loss: 32.1838 - val_loss: 26.7558 Epoch 99/102: ReduceLROnPlateau reducing learning rate to 9.999999948962493e-12. Epoch 99/102 9/9 [=====] - 150s 21s/step - loss: 30.3449 - val_loss: 25.4241 Epoch 100/102 9/9 [=====] - 151s 21s/step - loss: 30.8263 - val_loss: 27.8487 Epoch 101/102 9/9 [=====] - 151s 21s/step - loss: 30.7468 - val_loss: 27.8083 Epoch 102/102: ReduceLROnPlateau reducing learning rate to 9.999999992688135e-13. Epoch 102/102 9/9 [=====] - 151s 21s/step - loss: 30.5913 - val_loss: 25.4262 </pre>
3.	Confidence Score (Only Yolo Projects)	<p>Class Detected - 5</p> <p>Confidence Score - Average (61.01%)</p>	<pre> Epoch 95/102 9/9 [=====] - 155s 21s/step - loss: 30.2638 - val_loss: 25.2226 Epoch 96/102 9/9 [=====] - 152s 21s/step - loss: 29.6888 - val_loss: 27.3188 Epoch 97/102: ReduceLROnPlateau reducing learning rate to 9.99999991188888e-11. Epoch 97/102 9/9 [=====] - 152s 21s/step - loss: 32.2982 - val_loss: 25.5031 Epoch 98/102 9/9 [=====] - 151s 21s/step - loss: 30.7307 - val_loss: 25.8731 Epoch 99/102 9/9 [=====] - 150s 21s/step - loss: 31.4436 - val_loss: 26.8444 Epoch 100/102 9/9 [=====] - 153s 21s/step - loss: 32.1838 - val_loss: 26.7558 Epoch 101/102: ReduceLROnPlateau reducing learning rate to 9.999999948962493e-12. Epoch 101/102 9/9 [=====] - 150s 21s/step - loss: 30.3449 - val_loss: 25.4241 Epoch 102/102 9/9 [=====] - 151s 21s/step - loss: 30.8263 - val_loss: 27.8487 Epoch 103/102 9/9 [=====] - 151s 21s/step - loss: 30.7468 - val_loss: 27.8083 Epoch 104/102: ReduceLROnPlateau reducing learning rate to 9.999999992688135e-13. Epoch 104/102 9/9 [=====] - 151s 21s/step - loss: 30.5913 - val_loss: 25.4262 </pre>

8.2 User Acceptance Testing

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the AI Based Localization and classification of skin disease with erythema project at the time of the release to User Acceptance Testing (UAT).

Section	Total Cases	Not Tested	Fail	Pass
Registration	2	0	0	2
Registration Confirmation mail	1	0	0	1
Login (correct credentials)	1	0	0	1

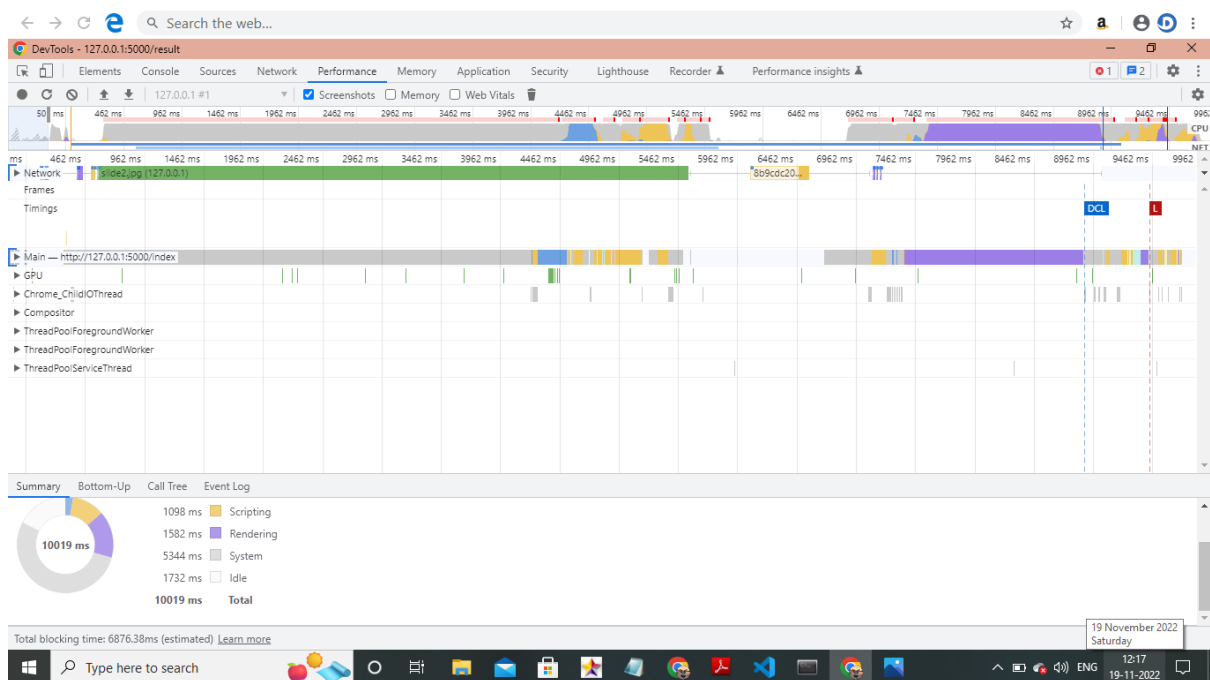
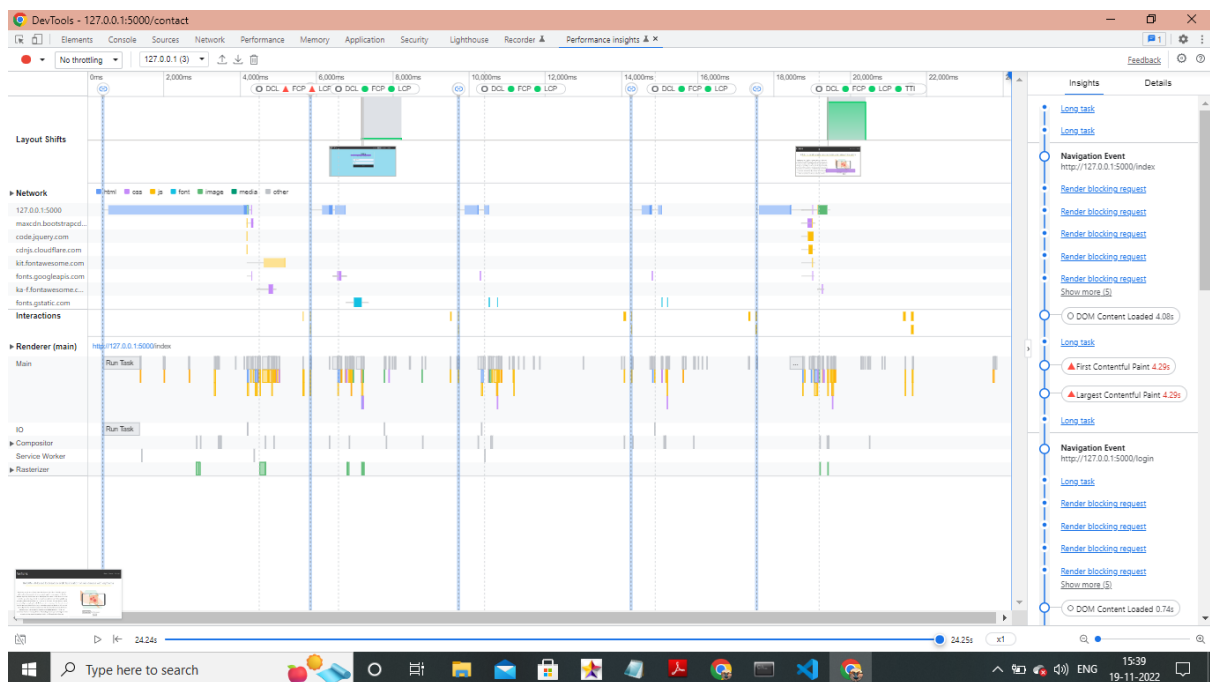
This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	4	3	0	0	7
Duplicate	0	0	0	0	0
External	0	2	0	0	2
Fixed	5	2	1	0	8
Not Reproduced	0	0	1	0	1
Skipped	0	0	0	0	0
Won't Fix	0	1	0	0	1
Totals	9	8	2	0	19

9. RESULTS

9.1 Performance Metrics

Final binary mask which is provided on image to get region of interest in image. Final binary mask is created after the number of iteration provide in localized segmentation. There are number of iteration provide in segmentation to get more accurate result. After applying no iteration result of localized segmentation



10. ADVANTAGES & DISADVANTAGES

ADVANTAGES

- * The user can quickly identified a skin disease in the disease stared stage
- * The user can consult a specialist doctor by using this project and get self motivated
- * we can cure a disease in early stages of disease

DISADVANTAGES

- * some user can fear about the disease when they get a negative result
- *The person are in a deep depression after knowing the disease
- *they don't concentrate on the other works

11. CONCLUSION

In this system we presented the automated system to find out the affected area from dermatological disease. As per the area results we decided stages of diseases that is initial stage, medium stage or final stage of disease. Localized Segmentation of the image has been done successfully. An algorithm has been developed to identify the area of infection from segmented image. In addition to work done above we can add some more advancement such as more feature extraction from image. Using the data to train artificial neural network by using such techniques we will obtain final results in next stage.

12.FUTURE SCOPE

- * In feature we can upgrade a daily notification for the use get self motivated
- * And also update a doctor consultant by using a video or audio conference through our project
- *we convert our project into mobile phone app for easy usage

13.APPENDIX

Source Code

app.py

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

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

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

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

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

```

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

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

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

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

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

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

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

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

FLAGS = None

from cloudant.client import Cloudant

# Authenticate using an IAM API key
client=Cloudant.iam('2afd311a-94e6-4743-a958-641316eb4d7e-
    bluemix','MDGr8AgVHb0DdqM4hcpakD4bzz7k-
    2CyDs_UcCs2ZHMW',connect=True)

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

app=Flask(__name__)

#default home page or route
@app.route('/',methods=["GET","POST"])
def index():
    return render_template('index.html')

@app.route('/index',methods=["GET","POST"])
def home():
    return render_template("index.html")

```

```

@app.route('/contact',methods=["GET","POST"])
def contact():
    return render_template('contact.html')

#registration page
@app.route('/register',methods=["GET","POST"])
def register():
    return render_template('register.html')

@app.route('/afterreg', methods=["GET","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',methods=["GET","POST"])
def login():
    return render_template('login.html')

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

```

```
passwd = request.form['passwd']
print(user,passwd)
```

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

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

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

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

```
@app.route('/logout',methods=["GET","POST"])
def logout():
    return render_template('logout.html')
```

```
@app.route('/prediction',methods=["GET","POST"])
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)

```

Login.html

```

<body style="font-family:Montserrat;">

<div class="header">
<div style="width:50%;float:left;font-size:2vw;text-align:left;color:white; padding-
top:1%">RASKIN</div>
<div class="topnav-right" style="padding-top:0.5%;">

<a href="index">Home</a>

```

```
<a class="active" href="login">Sign In</a>
<a href="register">Sign Up</a>
<a href="contact">Contact</a>
```

```
</div>
</div>
```

```
<div id="SignIn" class="SignIn">
```

```
    <form action="prediction" method="post">
    <div class="imgcontainer">
        <h2>SIGN IN</h2>
        <h6 class="information-text">Enter your registered email and your password.</h6>
    </div>
```

```
        <div class="container">
```

```
            <input type="email" placeholder="Enter registered email ID" name="_id"
            required><br>
```

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

```
        <button type="submit">Sign In</button><br>
```

```
    </div>
</form>
```

```
</div>
</div>
```

```
</body>
```

register.html

```
<body style="font-family:Montserrat;">
```

```
<div class="header">
```

```
<div style="width:50%;float:left;font-size:2vw;text-align:left;color:white; padding-
top:1%">RASKIN</div>
```

```
    <div class="topnav-right" >
```

```
        <a href="index">Home</a>
```

```
        <a href="login">Sign In</a>
```

```
        <a class="active" href="register">Sign Up</a>
```

Contact

</div>

</div>

<div id="SignIn" class="SignIn">

<form action="/prediction" method="post">

<div class="imgcontainer">

<h2>SIGN UP</h2>

<h6 class="information-text">Enter your name email and password.</h6>

</div>

<div class="container">

<input type="text" placeholder="Enter Name" name="name" required>

<input type="email" placeholder="Enter Email ID" name="_id" required>

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

<button type="submit">Register</button>

</div>

<div class="container" style="background-color:#f1f1f1">

<div class="psw">Already have an account? Sign In

</div>

</div>

</form>

</div>

</body>

index.html

<body>

<header id="head" class="header">

<section id="navbar">

<h1 class="nav-heading">RASKIN</h1>

<div class="nav--items">

Sign In

Sign Up

Contact

</div>

</section>

```

<div class="top">
  <h2 class="title text-muted">
    <p style="font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;">A
    PERFECT LIFE WITH PERFECT SKIN</p>
  </h2>

</div>
<section id="slider">
  <div id="carouselExampleIndicators" class="carousel" data-ride="carousel">
    <ol class="carousel-indicators">
      <li data-target="#carouselExampleIndicators" data-slide-to="0" class="active
"></li>
      <li data-target="#carouselExampleIndicators" data-slide-to="1"></li>
      <li data-target="#carouselExampleIndicators" data-slide-to="2"></li>
    </ol>
    <div class="carousel-inner">

      <div class="carousel-item active">
        
      </div>
      <div class="carousel-item">
        
      </div>

    </div>
    <a class="carousel-control-prev" href="#carouselExampleIndicators"
role="button" data-slide="prev">
      <span class="carousel-control-prev-icon" aria-hidden="true"></span>
      <span class="sr-only">Previous</span>
    </a>
    <a class="carousel-control-next" href="#carouselExampleIndicators"
role="button" data-slide="next">
      <span class="carousel-control-next-icon" aria-hidden="true"></span>
      <span class="sr-only">Next</span>
    </a>
  </div>

</section>
</header>

<section id="about">
  <div class="top">
    <h3 class="title text-muted">
      ABOUT PROJECT
    </h3>
  </div>

```

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

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

<section id="footer">

</section>
</body>
```

contact.html

```
<body>

  <div class="header">
```



```
<div style="width:50%;float:left;font-size:2vw;text-align:left;color:white;
padding-top:1%">RASKIN</div>
```

```
<div class="topnav-right" style="padding-top:0.5%;">
```

```
<a href="index">Home</a>
```

```
<a href="login">Sign In</a>
```

```
<a href="register">Sign Up</a>
```

```
</div>
```

```
</div>
```

```
<section>
```

```
<div class="container">
```

```
<div class="contactInfo">
```

```
<div>
```

```
<h2>Contact Info</h2>
```

```
<ul class="info">
```

```
<li>
```

```
<span></span>
```

```
<span><a href = "mailto:
```

```
cse221107@saranathan.ac.in">cse221107@saranathan.ac.in</a></span>
```

```
</li>
```

```
<li>
```

```
<span></span>
```

```
<span>8825600826</span>
```

```
</li>
```

```
</ul>
```

```
</div>
```

```
</div>
```

```
<div class="contactForm">
```

```
<h2>Send a Message</h2>
```

```
<div class="formBox">
```

```
<div class="inputBox w50">
```

```
<input type="text" name="" required>
```

```
<span>First Name</span>
```

```
</div>
```

```
<div class="inputBox w50">
```

```
<input type="text" required>
```

```
<span>Last Name</span>
```

```
</div>
```

```
<div class="inputBox w50">
  <input type="email" required>
  <span>Email Address</span>
</div>
<div class="inputBox w50">
  <input type="text" required>
  <span>Mobile Number</span>
</div>
<div class="inputBox w100">
  <textarea required></textarea>
  <span>Write your message here...</span>
</div>
<div class="inputBox w100">
  <input type="submit" value="Send">
</div>
</div>
</div>
```

```
</section>
</body>
```

GitHub & Project Demo Link

<https://github.com/IBM-EPBL/IBM-Project-16815-1659623334>

https://drive.google.com/file/d/10cH0wImJBbrnzRKU-sgSVmuzF0_Q4VHT/view?usp=sharing