Project Report

INTRODUCTION

1.1. Project Overview

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 color play an important role in skin disease detection. Color and coarseness of skin are visually different. Automatic processing of such images for skin analysis requires quantitative discriminator to differentiate the diseases.

1.2. Project Purpose

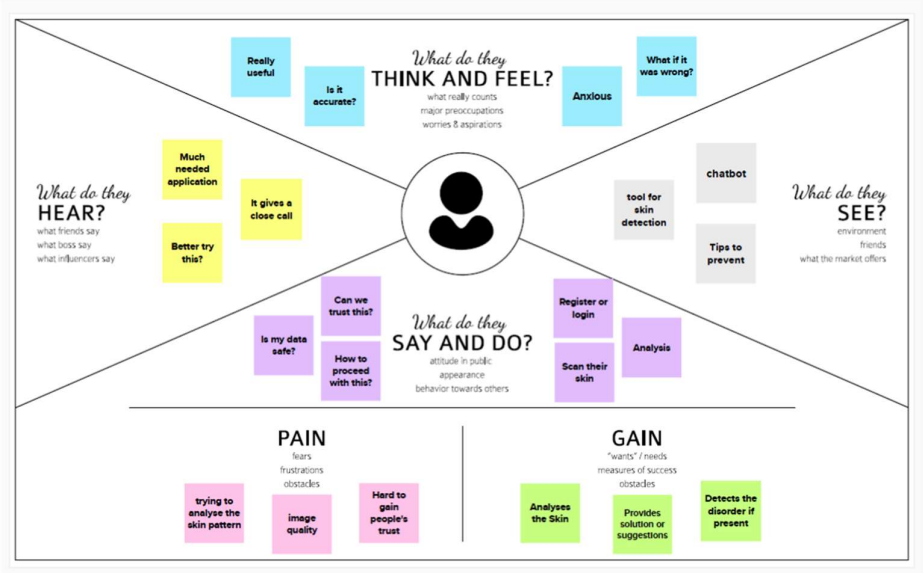
The goal is to identify and find cure for the affecting skin disease in an efficient and easy way possible by using advance artificial intelligence. We make it possible by building a model which is used for the prevention and early detection of skin cancer, psoriasis. Basically, skin disease diagnosis 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. LITERATURE SURVEY

2.1 Existing Problem
2.2 References
2.3 Problem Statement Definition

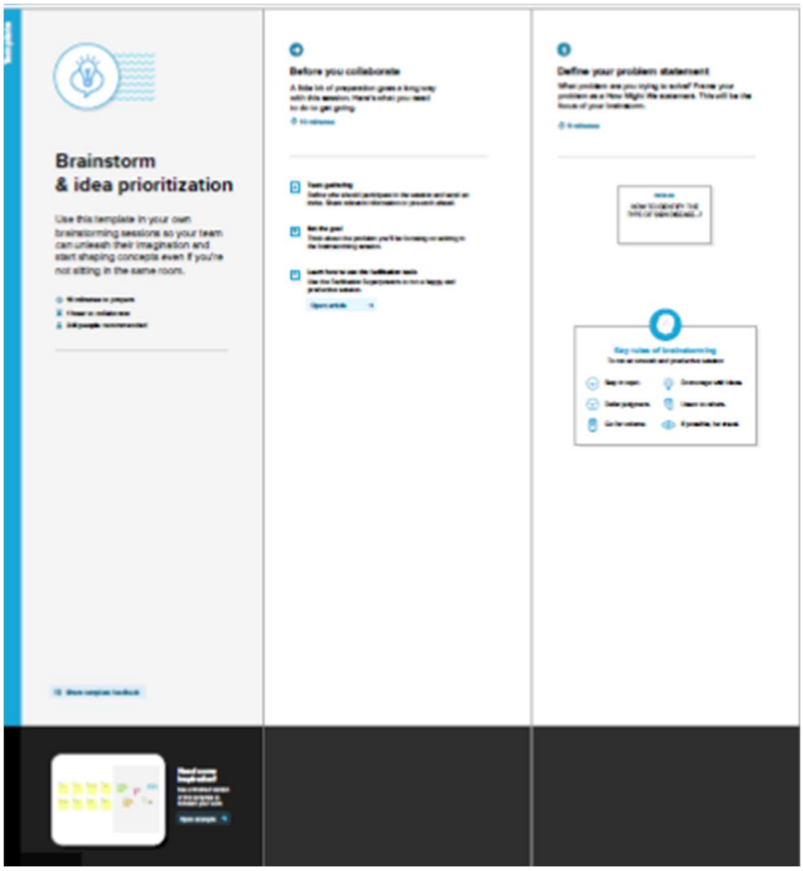
3. IDEATHON AND PROPOSED SOLUTION

3.1 Empathy Canvas Map

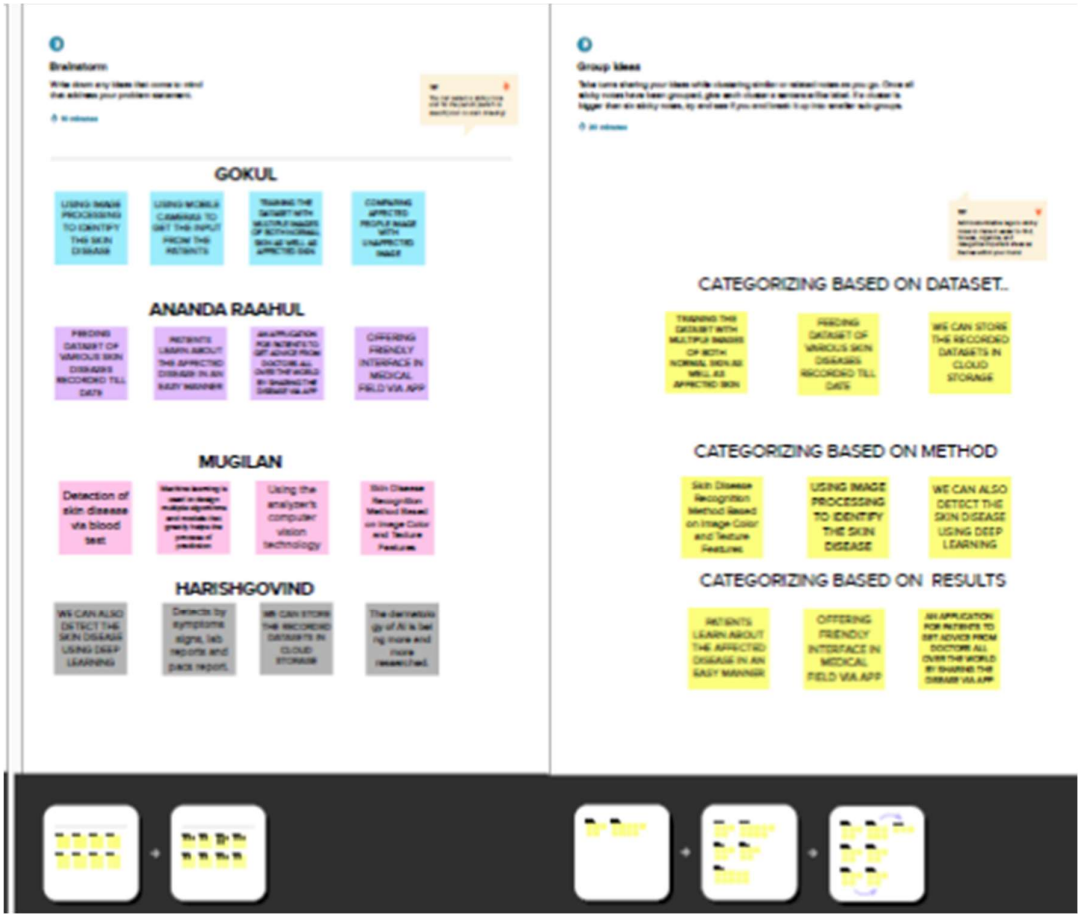


3.2 Ideation & Brainstorming

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



Step-2: Brainstorm, Idea Listing and Grouping



Step-3: Idea Prioritization

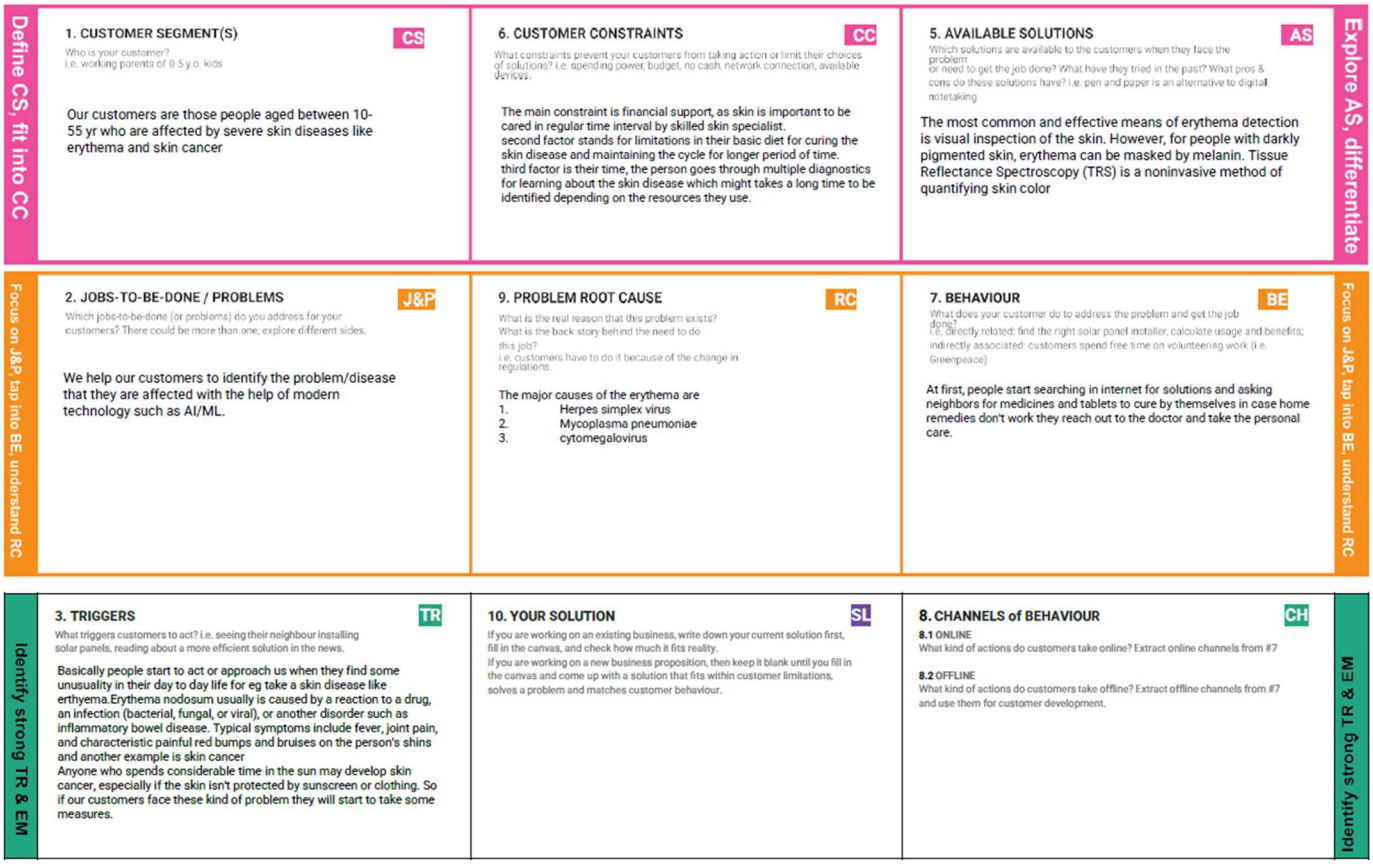
3.3 Proposed Solution

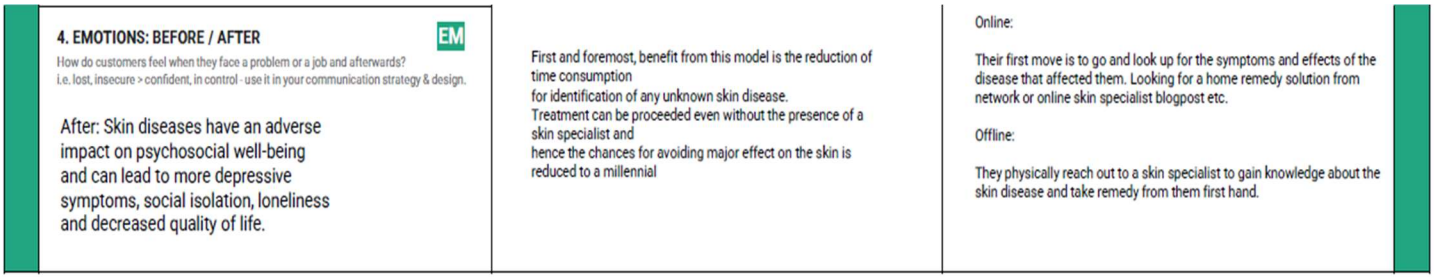
S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Skin diseases vary in types, we are given the problem to identify the type of skin disease using AI based localization with Erythema. Our main goal is to identify the type of disease that the patient is affected with to make them aware and start to proceed by treating their skin.
2.	Idea / Solution description	Our idea is to train the machine with various types of skin diseases by applying the knowledge of Data Science and Advance

		Machine Learning to make it capable for identifying the skin disease by using the set of fed Dataset. The machine uses advanced algorithms and image processing method to achieve the main goal/objective.
3.	Novelty / Uniqueness	The machine will be trained with the latest version of YOLO which currently is YOLOv5. Compared to other solutions performances, this machine is set to be fast and accurate in terms of mean average precision (mAP) and intersection over union (iOU) as well. It runs significantly faster than other detection methods with comparable performance. The images that are trained to follow the algorithm are trained by using Microsoft Visual Object Tagging Tool or VOTT in short, which is an open-source annotation and labelling tool for image and video assets including features such as; The ability to label images or video frames, extensible model for importing data from local or cloud storage providers, and model for exporting labelled data to local or cloud storage providers. For the current project, our datasets and image annotate will be stored in Cloudant DB, which is a non-relational, distributed database service provided by IBM.
4.	Social Impact / Customer Satisfaction	○ With our product the patients will learn about the type of disease that had affected them so that they could act accordingly to cure it. ○ In the absence of a skin specialist, the machine can identify the affecting disease and hence the patient will be able to look for a cure rather than waiting for a doctor. ○ Providing a user-friendly interface and an ease for the patients to use it.
5.	Business Model (Revenue Model)	► First and foremost, benefit from this model is the reduction of time consumption for identification of any unknown skin disease. ► Treatment can be proceeded even without the presence of a skin specialist and hence the chances for not avoiding

		major effect on the skin is reduced to a millennial.
6.	Scalability of the Solution	With the advancement of technology, the software will be updated annually. New datasets will be recorded and stored in the cloud storage for new cases of unknown skin diseases. The training of images and algorithm applied will also be updated timely with maintenance.

3.4 Problem Solution fit





4. REQUIREMENT ANALYSIS

4.1 Functional requirements

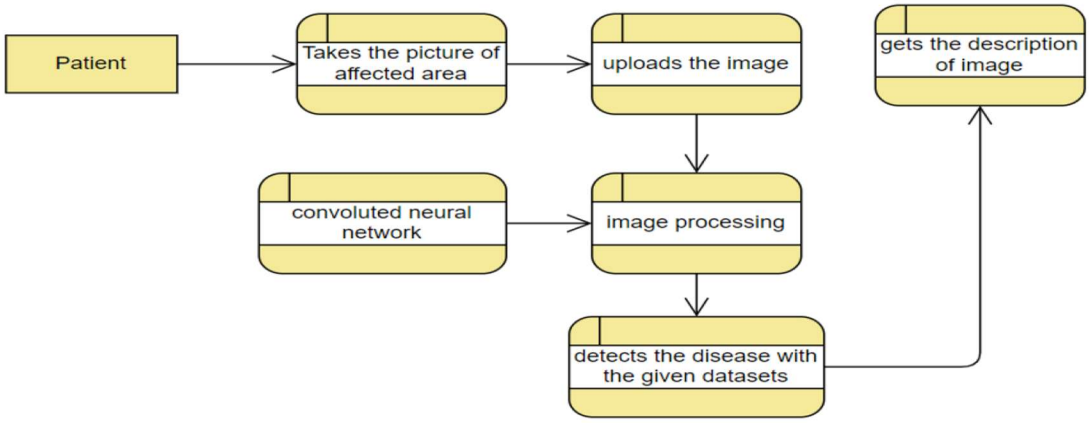
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Profile	Users provides their medical history.
FR-4	User Uploads Images (Input)	Upload the images as jpeg/jpg's Upload images as png's
FR-5	Output Analysis	Output analyzed through trained model
FR-6	Provides Description	Gives the detailed description of the skin disease found

4.2 Non-Functional requirements

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Used to classify skin disease with erythema
NFR-2	**Security**	It offers greater security and prevents unauthorized individuals from accessing user's data.
NFR-3	**Reliability**	Even with more users, there will be a good performance without failure.
NFR-4	**Performance**	With greater accuracy, the performance is high.
NFR-5	**Availability**	With a good system, all authorized users can access it.
NFR-6	**Scalability**	Performance will be good even with the higher user traffic,

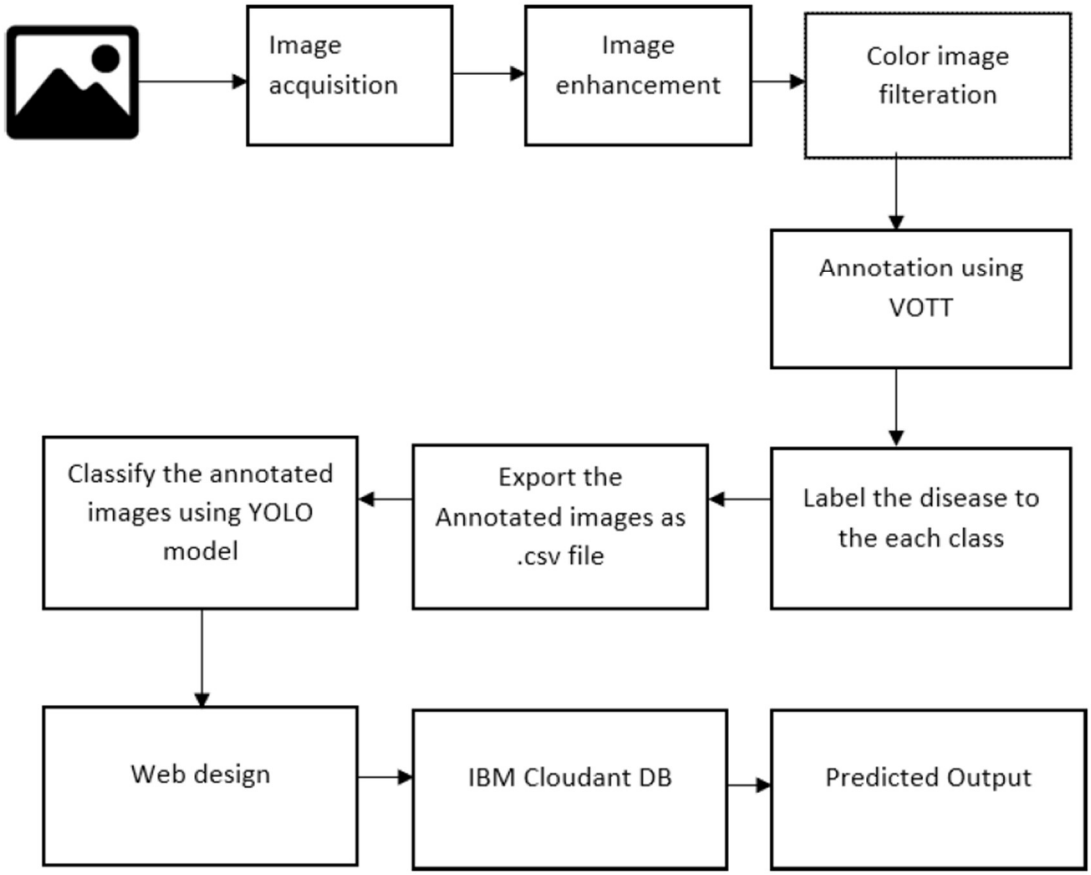
5. PROJECT DESIGN

5.1 Data Flow Diagram

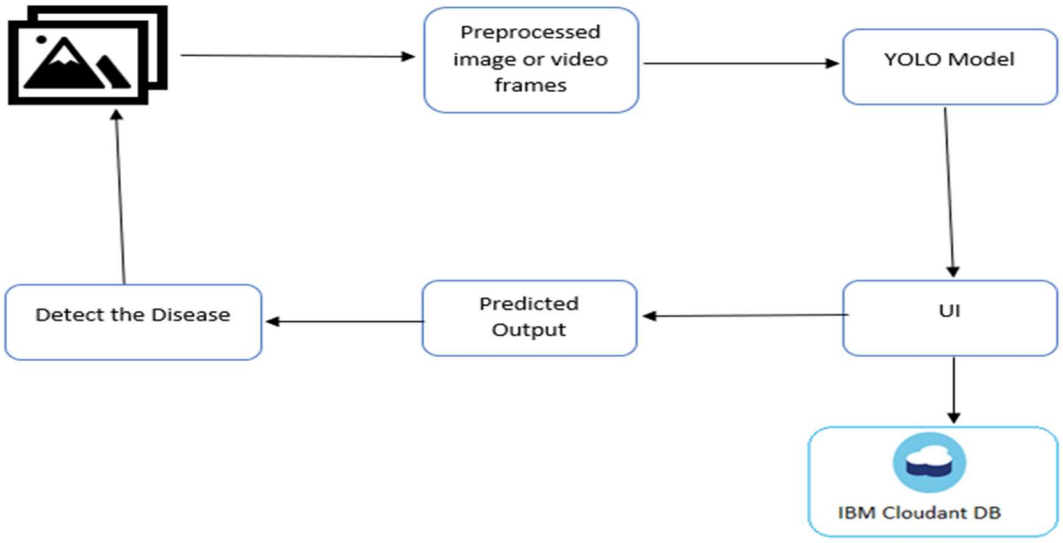


5.2 Solution & Technical Architecture

Solution Architecture:-



Technical Architecture:-



5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
	Confirm	USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
	Registration	USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	High	Sprint-2
	Registration	USN-4	As a user, I can register for the application through Gmail	I can access my account / dashboard	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can access Dashboard	High	Sprint-1
	Dashboard	USN-6	As a user, I can Access my Dashboard.	I can interact with the inteface	Medium	Sprint-3
	Data Input	USN-7	As a user, I can upload the images of the affected skin area	I can submit it to the application	High	Sprint-4

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer Care Executive	Solution	USN-8	Responding to each email you receive	Offer a solution for how your company can improve the customer's experience.	High	Sprint-3
Administrator	Manage	USN-9	Do-it-yourself service for delivering Everything.	set of predefined requirements that must be met to mark a user story complete.	High	Sprint-4
Training Model	Image processing	USN-10	By comparing the images, the disease will be detected with the given datasets	All the necessary operation performed and information extracted	High	Sprint-3
	Report generation	USN-11	Based on the detection of disease, report generated	The results will be shown on the screen to the patients	High	Sprint-4

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Pre-requisites	USN-1	Install Python IDE, Python packages, Microsoft Visual Object Tagging Tool, Yolo Structure	7	High	Gokul Mugilan
Sprint-1	Data Collection	USN-2	The dataset should be collected in realtime or from the gallery or collect it from google.	10	High	Ananda raahul Harishgovindh
Sprint-1	Annotate Images	USN-3	Create a project in Visual Object Tagging Tool	3	Medium	Gokul Mugilan
Sprint-2	Training YOLO	USN-4	In this we will train our model using YOLO weights	5	Medium	Ananda raahul Harishgovindh
Sprint-2		USN-5	Download and convert pre-trained weights	5	High	Gokul Ananda raahul
Sprint-2		USN-6	To start training run the training script within the YOLO structure.	10	Low	Mugilan Harish govind
Sprint-3	Cloudant DB	USN-7	Register and Login to IBM Cloud	5	Medium	Mugilan
Sprint-3		USN-8	Create Service Instant and credentials	5	High	Ananda raahul Harishgovindh

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-3		USN-9	Launch Cloudant DB and then create database	2	High	Gokul Mugilan
Sprint-3	Developing Phase	USN-10	In this build a web application that is integrated to the caffemodel.	3	Low	Ananda raahul Harishgovindh
Sprint-3		USN-11	For this build HTML Pages	2	Medium	Gokul Mugilan
Sprint-3		USN-12	Develop and build the python code to run the application.	3	High	Gokul Ananda raahul
Sprint-4	Testing Phase	USN-13	As a user login to the dashboard	10	High	Harishgovindh Mugilan
Sprint-4		USN-14	As a user import the skin affected disease image to the software application.	5	Medium	Anandaraahul Mugilan
Sprint-4		USN-15	YOLO will process the image and give the result as unaffected or affected with other details	5	Medium	Harishgovindh Gokul

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

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

6.2 Sprint Delivery Schedule

S.no	Milestone	Activities	Start Date	End Date
1	Project Objectives	Prepare the project objectives.	22-Aug-2022	27-Aug-2022
2	Pre-Requisites	➢ Install Python IDE ➢ Install Microsoft's Visual Object Tagging Tool ➢ Download YOLO Project Structure	22-Aug-2022	27-Aug-2022
3	Create Dataset	Creating Datasetfromscratch.	22-Aug-2022	27-Aug-2022
4	Annotate Images	Create a project in VOTT ▪ VOTT Project Creation.	27-Aug-2022	02-Sep-2022
5	Training YOLO	Download and Convert Pre-Trained weights. Train YOLOV3 Detector	24-oct-2022	19-Nov-2022
6	Cloudant DB	➢ Register and login to the IBM cloud ➢ Create service instance ➢ Create service credentials. ➢ Launch Cloudant DB ➢ Create Database	24-oct-2022	12-Nov-2022
7	Application Building	➢ Build HTML page ➢ Build PYTHON code ➢ Run the application	24-oct-2022	19-Nov-2022
8	Ideation Phase	➢ Literature Survey ➢ Empathy Map ➢ Ideation	29-Aug-2022	01-Oct -2022

S.no	Milestone	Activities	Start Date	End Date
9	Project Design Phase - I	➢ Proposed Solution ➢ Problem Solution Fit ➢ Solution Architecture	19-Sept-2022	01-Oct-2022
10	Project Design Phase - II	➢ Customer Journey ➢ Functional Requirement ➢ Data Flow Diagrams ➢ Technology Architecture	03-Oct-2022	15-Oct-2022
11	Project Planning Phase	➢ Prepare Milestone & Activity List ➢ Sprint Delivery Plan	17-Oct-2022	21-Oct-2022
12	Project Development Phase	➢ Project Development - Delivery of Sprint-1 ➢ Project Development - Delivery of Sprint-2 ➢ Project Development - Delivery of Sprint-3 ➢ Project Development - Delivery of Sprint-4	24-Oct-2022	19-Nov-2022

CODING AND SOLUTIONING

7.1 Feature 1

//Index Page HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">


   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">
   <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>
   <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous"></script>
   <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl" crossorigin="anonymous"></script>


   <script src="https://kit.fontawesome.com/8b9cdc2059.js" crossorigin="anonymous"></script>
   <link href="https://fonts.googleapis.com/css2?family=Akronim&family=Times New Roman&display=swap" rel="stylesheet">
```

```
    <title>SKINBOSS</title>
</head>
<body>
<style>
.icon-bar {
  width: 90px; /* Set a specific width */
  background-color: #555; /* Dark-grey background */
}

.icon-bar a {
  display: block; /* Make the links appear below each other instead of side-by-side */
  text-align: center; /* Center-align text */
  padding: 16px; /* Add some padding */
  transition: all 0.3s ease; /* Add transition for hover effects */
  color: white; /* White text color */
  font-size: 36px; /* Increased font-size */
}

.icon-bar a:hover {
  background-color:black; /* Add a hover color */
}

.active {
  background-color:grey; /* Add an active/current color */
}
.nav--items {
  overflow: hidden;
  background-color: #f1f1f1;
}

/* Style the buttons that are used to open the tab content */
.nav--items a {
  color:black;
  background-color: inherit;
  float: right;
  border: none;
  outline: none;
  cursor: pointer;
  padding: 14px 16px;
  transition: 0.3s;
}

/* Change background color of buttons on hover */
.nav--items a:hover {
  background-color: #ddd;
}

.nav--items a.active {
```

```
        color:black;
  background-color: #ccc;
}
.heading{
text-align:center;
color:white;
background-color:#22DDCA;
}
.top{
background-color:#7BEADF;
}
p{
color:black;
}
.head{
font-family:timesnewroman;
color:black;
text-align:center;
}

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

  <section id="navbar">
        <h1 class="heading">SKIN BOSS</h1>
     <div class="nav--items">

        <a href="{{url_for('login')}}">Log In</a>
        <a href="{{url_for('signup')}}">Sign Up</a>
          <a href="{{url_for('logout')}}">Log Out</a>
              <a href="{{url_for('prediction')}}">Prediction</a>
     </div>
  </section>

  <div class="top">
   <h2 class="title text-muted">
      <p style="font-family:timesnewromain">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-item active">
        <img class="d-block w-100" src="https://regencyhealthcare.in/wp-content/uploads/2021/04/Artboard-5-1.png" alt="First slide" style="height:500px">
      </div>
      <div class="carousel-item">
        <img alt="Second slide" class="d-block w-100" src="https://img.graphicsurf.com/2020/10/skin-disease-vector-design-concept.jpg" style="height:500px">
      </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>
      <div class="Problem">
        <h1 class="head">Problem Statement</h1>
        <p style="font-family:tahoma;">
          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.</p>
        <h1 class="head">Proposed Solution</h1>
        <p style="font-family:tahoma;">
          Different skin disorders can be detected by just
          submitting photographs, and this approach is
          quite effective at helping people in the
          community identify ailments earlier.
          Our return on investment will be the creation
          and distribution of a proprietary product that
          will be used as a solution.
          This system is more scalable because it accepts
          any picture type, regardless of resolution, and
          offers good performance in any situation.
```

```
        </p></div>

    </header>



</body>
</html>
```

7.2 Feature 2

//Model to train

```
# -*- coding: utf-8 -*-
"""Untitled0.ipynb
Automatically generated by Colaboratory.
Original file is located at
 https://colab.research.google.com/drive/1PYFZ7zKhWpFF5YilnguhZ8X1EgtSIJN4
"""
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
from pyngrok import ngrok
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'/content/drive/MyDrive/IBM_PROJECT/yolo_structure/2_Training/src'
print(src_path)
utils_path=r'/content/drive/MyDrive/IBM_PROJECT/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('ef7f4729-2486-45c5-a7fa-f4140373e2e6-bluemix','6GfFjs3engXLnSJB8Kp4f
bs7HTKwrJpWJE7wNPGzZPVW', connect=True)
# Create a database using an initialized client
my_database = client.create_database('my_database')
app=Flask(_name_)
port_no=5000
ngrok.set_auth_token("2H7aM94zEuTa40t3J6jKpIqWAc3_B2UxzZs6qxetntgadxQW")
public_url = ngrok.connect(port_no).public_url
print(f"To acces the Gloable link please click {public_url}")
#default home page or route
@app.route('/')
def index():
 return render_template('index.html')
@app.route('/index.html')
def home():
 return render_template("index.html")
#registration page
@app.route('/register')
def register():
 return render_template('register.html')
@app.route('/afterreg', methods=['POST'])
def afterreg():
 x = [x for x in request.form.values()]
 print(x)
```

```
 data = {
 '_id': x[1], # Setting _id is optional
 'name': x[0],
 'psw':x[2]
 }
 print(data)

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

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

 print(len(docs.all()))

 if(len(docs.all())==0):
 url = my_database.create_document(data)
 #response = requests.get(url)
 return render_template('register.html', pred="Registration Successful, please
login using your details")
 else:
 return render_template('register.html', pred="You are already a member, please
login using your details")
#login page
@app.route('/login')
def login():
 return render_template('login.html')
@app.route('/afterlogin',methods=['POST'])
def afterlogin():
 user = request.form['_id']
 passw = request.form['psw']
 print(user,passw)

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

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

 print(len(docs.all()))

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

```
@app.route('/logout')
def logout():
 return render_template('logout.html')
@app.route('/prediction')
def prediction():
 return render_template('prediction.html',path="../static/img/6623.jpg",)
@app.route('/result',methods=["GET","POST"])
def res():
 # Delete all default flags
 parser = argparse.ArgumentParser(argument_default=argparse.SUPPRESS)
 """
 Command line options
 """
 f = request.files['file']
 f.save("./drive/MyDrive/IBM_PROJECT/Flask/static/img/"+f.filename)



 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"',
)
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),
}
)
img_path="/drive/MyDrive/IBM_PROJECT/Flask/static/img/"+f.filename
prediction, image,lat,lon= detect_object(
yolo,
img_path,
save_img=save_img,
save_img_path=FLAGS.output,
postfix=FLAGS.postfix,
)

yolo.close_session()
return
render_template('prediction.html',prediction=str(prediction),path="../static/img/"+f.filena
me)


""" Running our application """
if _name_ == "_main_":
app.run(port=port_no)
```

7.3 Database Schema

Currently we will be using the cloudantDB provided by IBM for storing all annotated images for already identified skin diseases and store in the cloud sever. As the images are scanned, the similar images are pulled out from the server for identifing the disease nature.

Output:

Number of training samples: 2000
Number of validation samples: 150

```
# preprocess data def
decode_img(img):

# convert the compressed string to a 3D uint8 tensor
img = 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 desired size.return
        tf.image.resize(img, [299, 299])


def process_path(filepath, label):
  # load the raw data 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 parameters
      batch_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()
  # shuffle the 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 while the 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:

```
# building the 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()
```

Output:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
keras_layer (KerasLayer)     multiple                  21802784
_________________________________________________________________
dense (Dense)                multiple                  2049
=================================================================


Total params: 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 checkpoint whenever we reach better weights
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,
              steps_per_epoch=n_training_samples // batch_size,
          validation_steps=n_validation_samples // batch_size, verbose=1, epochs=100,callbacks=[tensorboard,
              modelcheckpoint])
```

Output:

```
Train for 31 steps, validate for 2 steps
Epoch 1/100
30/31 [============================>.] - ETA: 9s - loss: 0.4609 - accuracy: 0.7760
Epoch 00001: val_loss improved from inf to 0.49703, saving model to benign-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 to benign-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("Number of 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")
```

```
Number of testing samples: 600#
    evaluation
# load testing set test_metadata_filename =
    "test.csv"
df_test = pd.read_csv(test_metadata_filename)
    n_testing_samples = len(df_test)
print("Number of 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")
```

```
# 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)
```

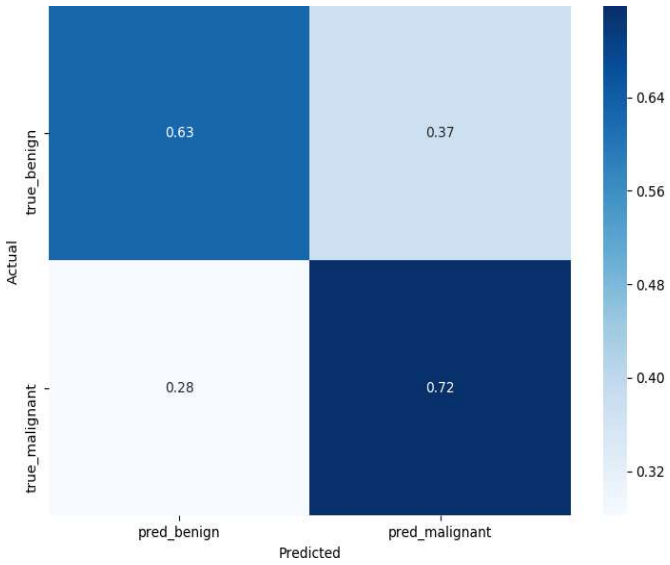
Output:

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

```
def  get_predictions(threshold=None):
  """
  Returns predictions for binary classification given `threshold`
  For instance, if threshold is 0.3, then it'll output 1 (malignant) for that sample ifthe
        probability of 1 is 30% or more (instead of 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 melanoma probabilityif
          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)
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
        it
  print(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 confusion matrix
      plt.show()

plot_confusion_matrix(y_test, y_pred)
```

Output:




```
sensitivity = sensitivity_score(y_test, y_pred)
      specificity = specificity_score(y_test, y_pred)

print("Melanoma Sensitivity:", sensitivity)
      print("Melanoma Specificity:", specificity)
```

Output:

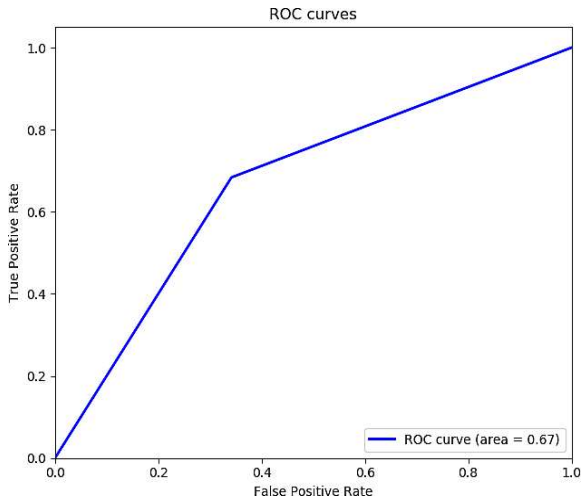
Melanoma Sensitivity: 0.717948717948718

Melanoma Specificity: 0.6252587991718427

```
def plot_roc_auc(y_true, y_pred):
   """
   This function plots the ROC curves and provides the scores."""
        # prepare for figure
```

```
        plt.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:

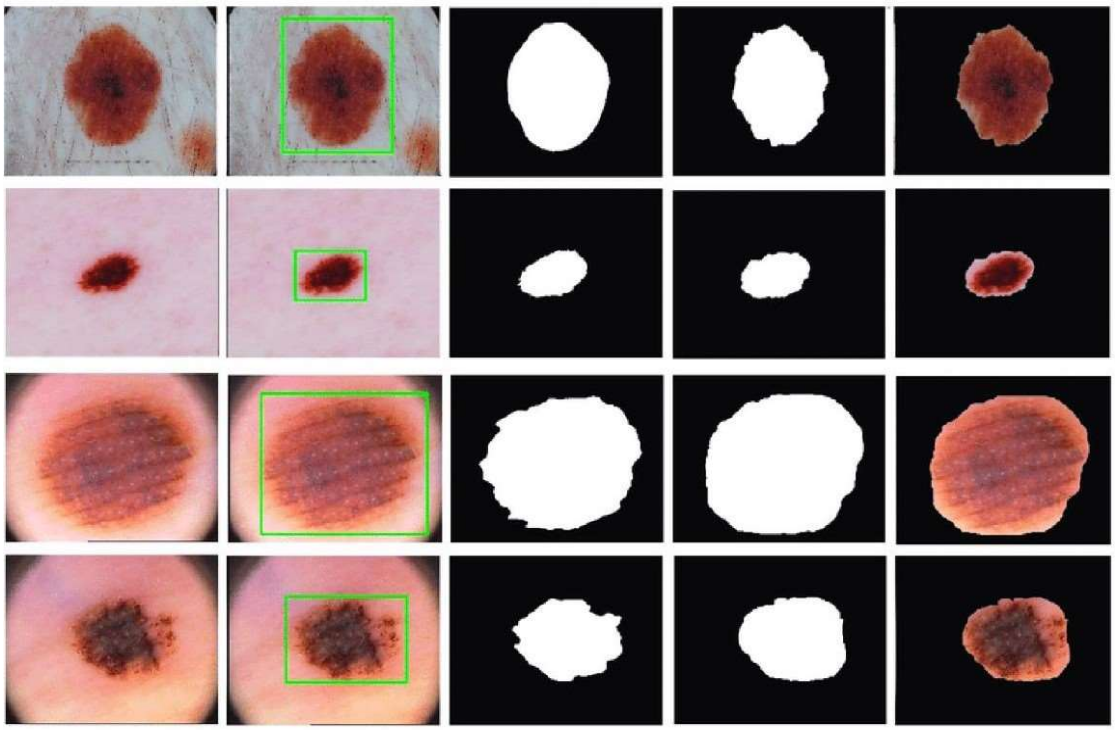


ROC AUC: 0.671

8. TESTING

8.1 Test Cases

The final results are based on the accuracy results in the form of the melanoma and the non-melanomaskin diseases classifications.

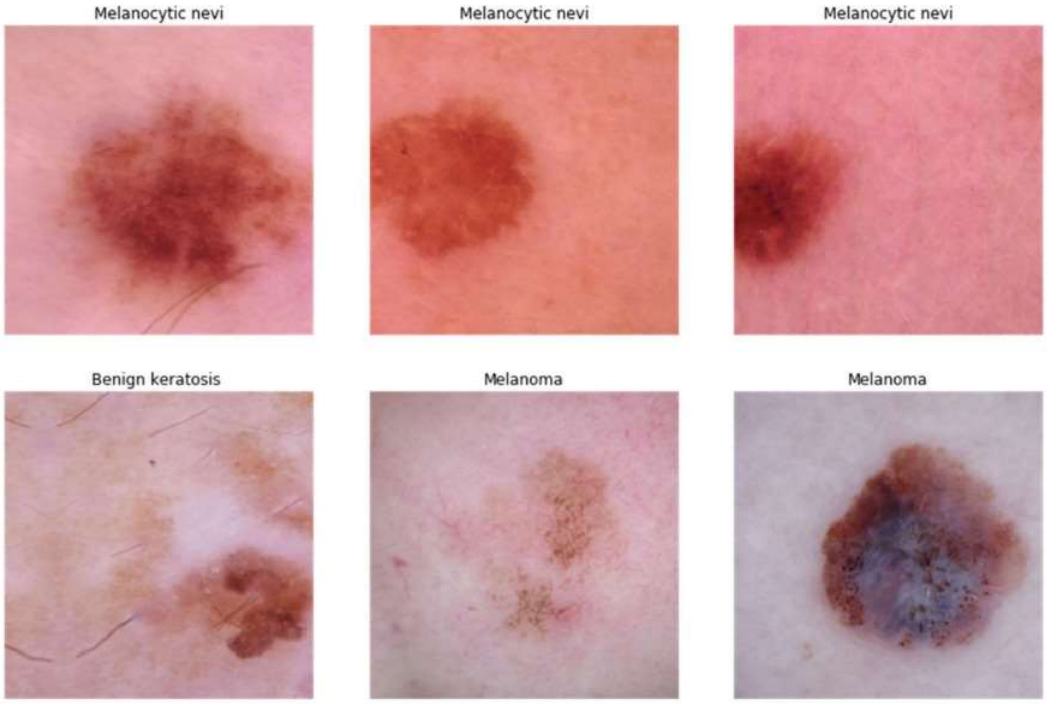


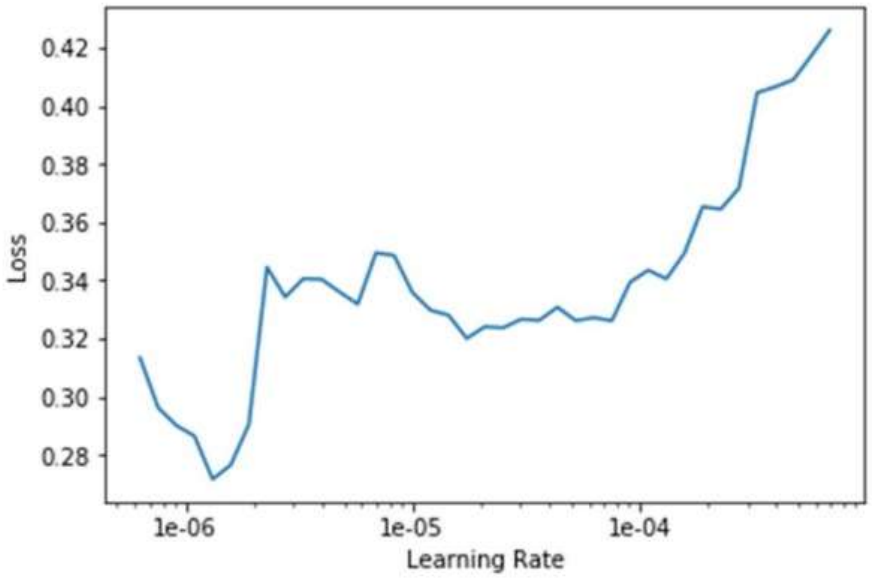
(a) (b) (c) (d) (e)

8.2 User Acceptance Testing

In [13]:

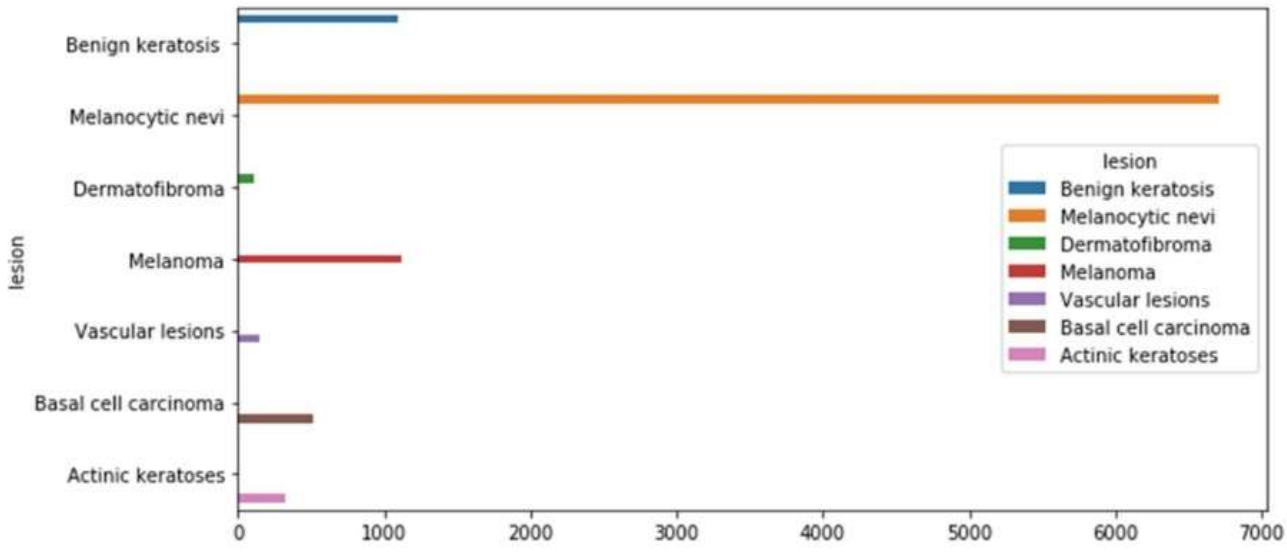
```
data.show_batch(rows=3)
```

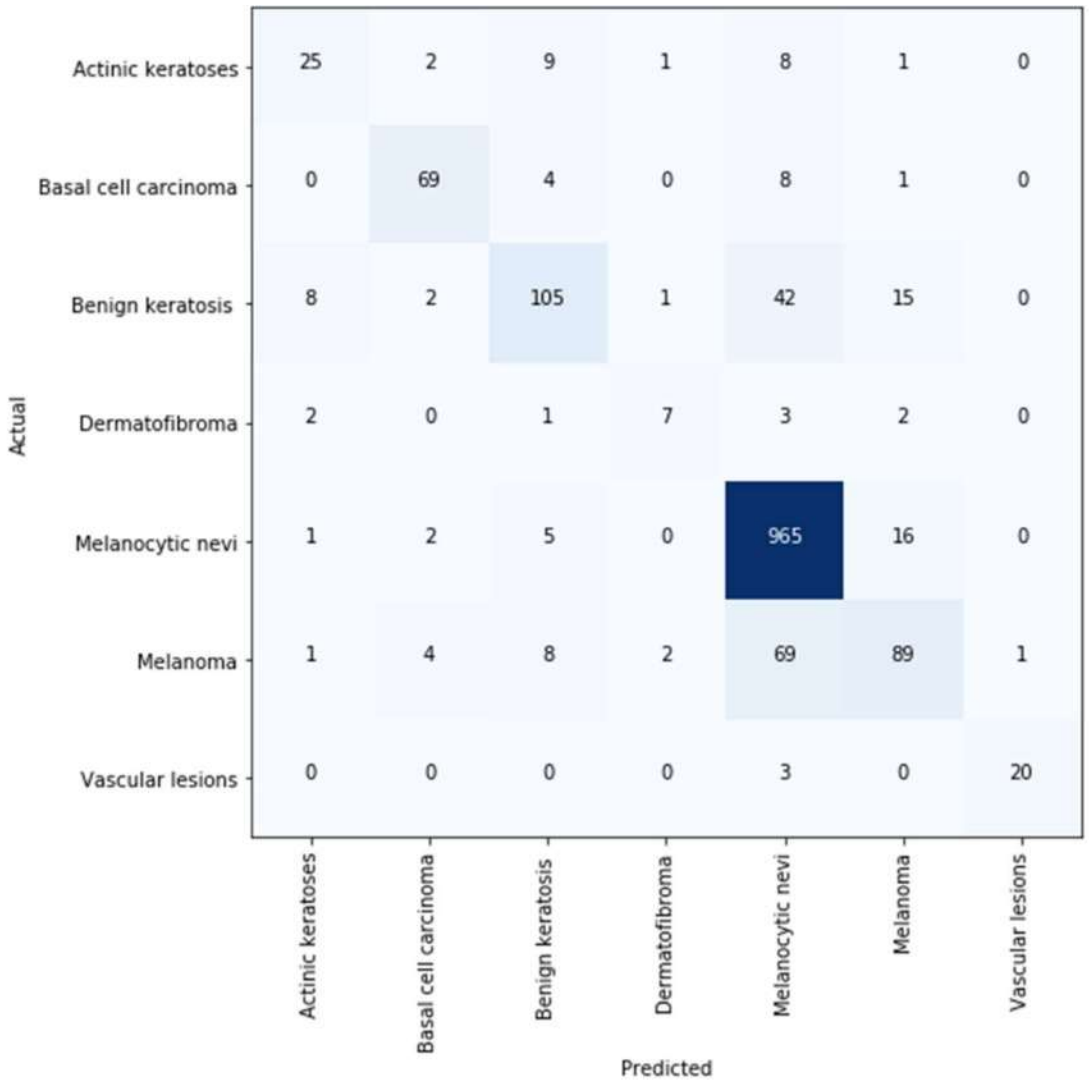




Out[8]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff27a102550>
```



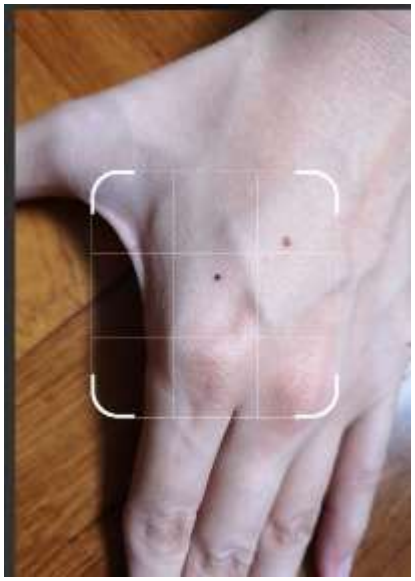

Actinic keratoses
Basal cell carcinoma
Benign keratosis
Dermatofibroma
Melanocytic nevi
Melanoma
Vascular lesions
Actual
25 2 9 1 8 1 0
0 69 4 0 8 1 0
8 2 105 1 42 15 0
2 0 1 7 3 2 0
1 2 5 0 965 16 0
1 4 8 2 69 89 1
0 0 0 0 3 0 20
Actinic keratoses
Basal cell carcinoma
Benign keratosis
Dermatofibroma
Melanocytic nevi
Melanoma
Vascular lesions
Predicted

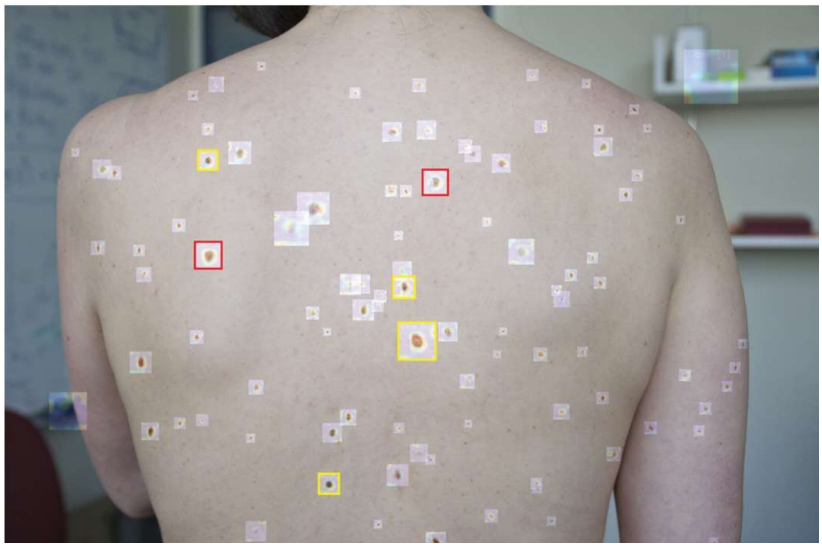
RESULTS

9.1 Performance Metrics

Performance metrics for classification with dement images.

Method	AUC	Specificity	Sensitivity	F1-score
Without segmentation	0.8207	0.9642	0.4748	0.4092
Contextual segmentation	0.8104	0.9652	0.4185	0.3876
Refined contextual segmentation	0.8802	0.9513	0.6141	0.6079





10. ADVANTAGES & DISADVANTAGES

10.1 Advantages

- ➔ An efficient and quick way to identify the skin problem in absense of a doctor.
- ➔ User friendly interface.
- ➔ User is able to seek possible cure after identification.
- ➔ Portable to use anywhere around the world.
- ➔ Secure data storage in cloud network.

10.2 Disadvantages

- ➔ Requires internet connection.
- ➔ A quality camera is required for capturing.

11. CONCLUSION

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

12. FUTURE SCOPE

This implementation of the Structural Co-Occurrence matrices for feature extraction in the skin diseases classification and the pre-processing techniques are handled by using the Median filter, this filter helps toremove the salt and pepper noise in the image processing; thus, it enhances the quality of the images, and normally, the skin diseases are considered as the risk factor in all over the world. Our proposed approach provides 97% of the classification of the accuracy results while another existing model such asFFT + SCM gives 80%, SVM + SCM gives 83%, KNN + SCM gives 85%, and SCM + CNN gives 82%. Future work is dependent on the increased support vector machine's accuracy in classifying skin illnesses, and SCM is used to manage the feature extraction technique.

13.APPENDIX

GitHub Link: https://github.com/IBM-EPBL/IBM-Project-18370-1659684341