AI BASED LOCALIZATION AND CLASSIFICATION OF SKIN DISEASE WITH ERYTHEMA

Team ID:PNT2022TMID28251

St. Joseph's Institute of Technology

Johny P – 312419106058 (Team Leader)

SN KailashSundaram – 312419106060 (Team Member)

Louis Santhosh – 312419106072 (Team Member)

Kamali V – 312419106064 (Team Member)

Kamali T – 312419106063 (Team Member)

TABLE OF CONTENTS

INTRODUCTION

1.1 Project Overview

Skin is the largest and most sensitive part of the human body which protects our inner vital parts and organs from the outside environment, hence avoiding contact with bacteria and viruses. Skin also helps in body temperature regulation. The skin consists of cells, pigmentation, blood vessels, and other components. It is comprised of 3 main layers, namely, the epidermis, the dermis, and the hypodermis.

Epidermis, being the outermost skin layer, forms a waterproof and protective sheath around the body's surface. The dermis, found beneath the epidermis, comprises of connective tissues and protects the body from stress and strain. A basement membrane tightly joins the dermis with the epidermis. The hypodermis, also called subcutaneous tissue, is not actually a part of the skin and lies below the dermis. It attaches the skin to the underlying bone and muscle and also supplies blood vessels and nerves to it.

1.2 Purpose

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 intelligencetechniques, in order to improve the accuracy of dermatology diagnosis.The segmentation and classification of skin diseases has been gaining attention in the field of artificial intelligence because of its promising results.

Here, the user can capture images of their skin, which are then sent to the trained model, where the information is processed using image processing techniques and then extracted for

machine interpretation. Finally, the model generates a result and determines whether or not the person has skin disease. Image processing technologies significantly reduce the time spent on a specific activity by the customer. Hence, it is a time- and money-saving process.

LITERATURE SURVEY

2.1 EXISTING PROBLEMS:

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 itis 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 ofskin 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. Themodel analyses the image and detect whether the person is having skin disease or not.

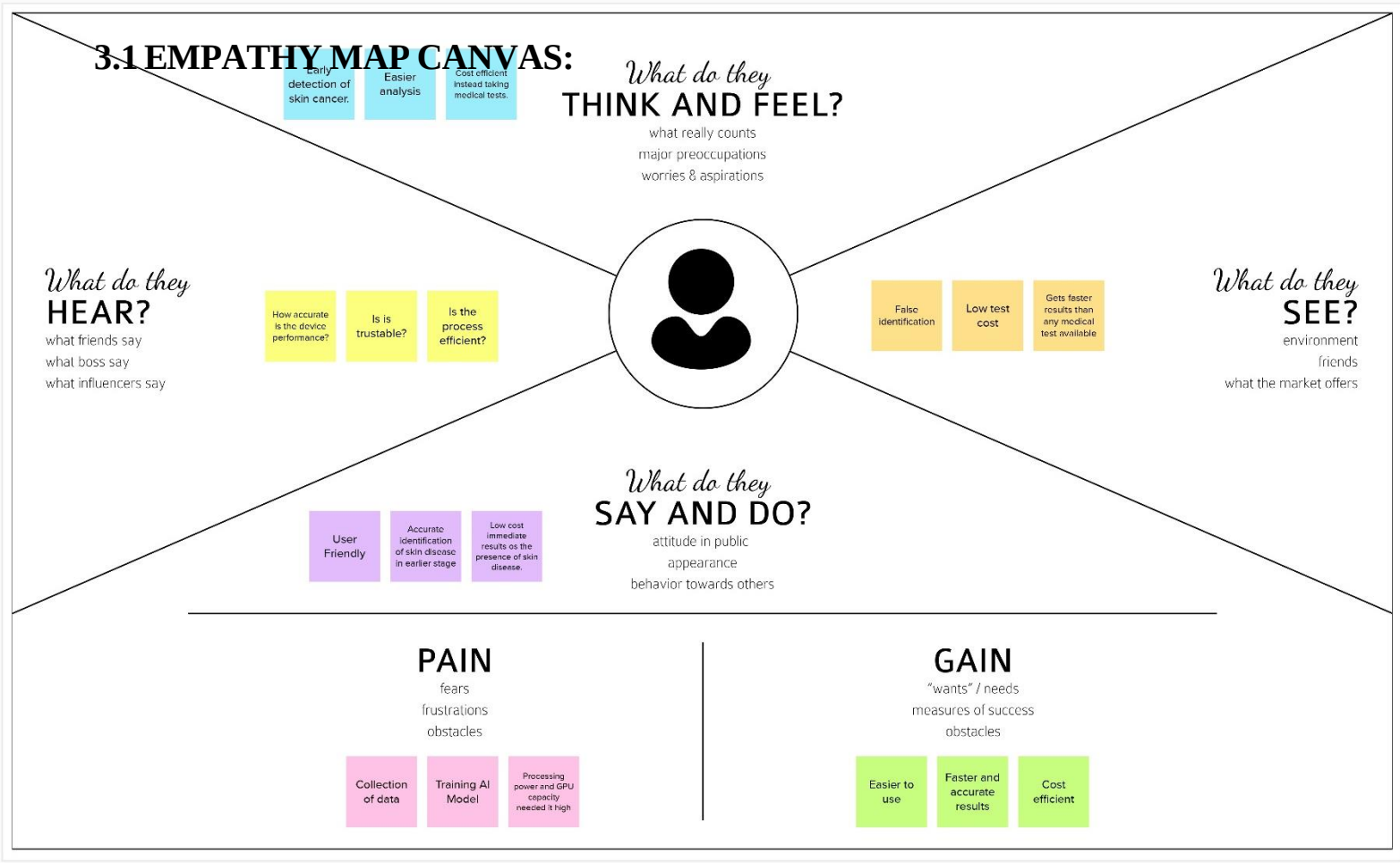
2.2 Problem Statement Definition

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 itsseverity.We are building a model which is used for the prevention and early detectionof skin cancer, psoriasis. Basically, skin disease diagnosis depends on the different characteristics like color, shape, texture etc…

The person can capture the images of skin and then the image will be sent to the trained model. The model analyzes the image and detects whether the person is havingskin disease or not.

3. IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS:



3.2 Ideation & Brainstorming

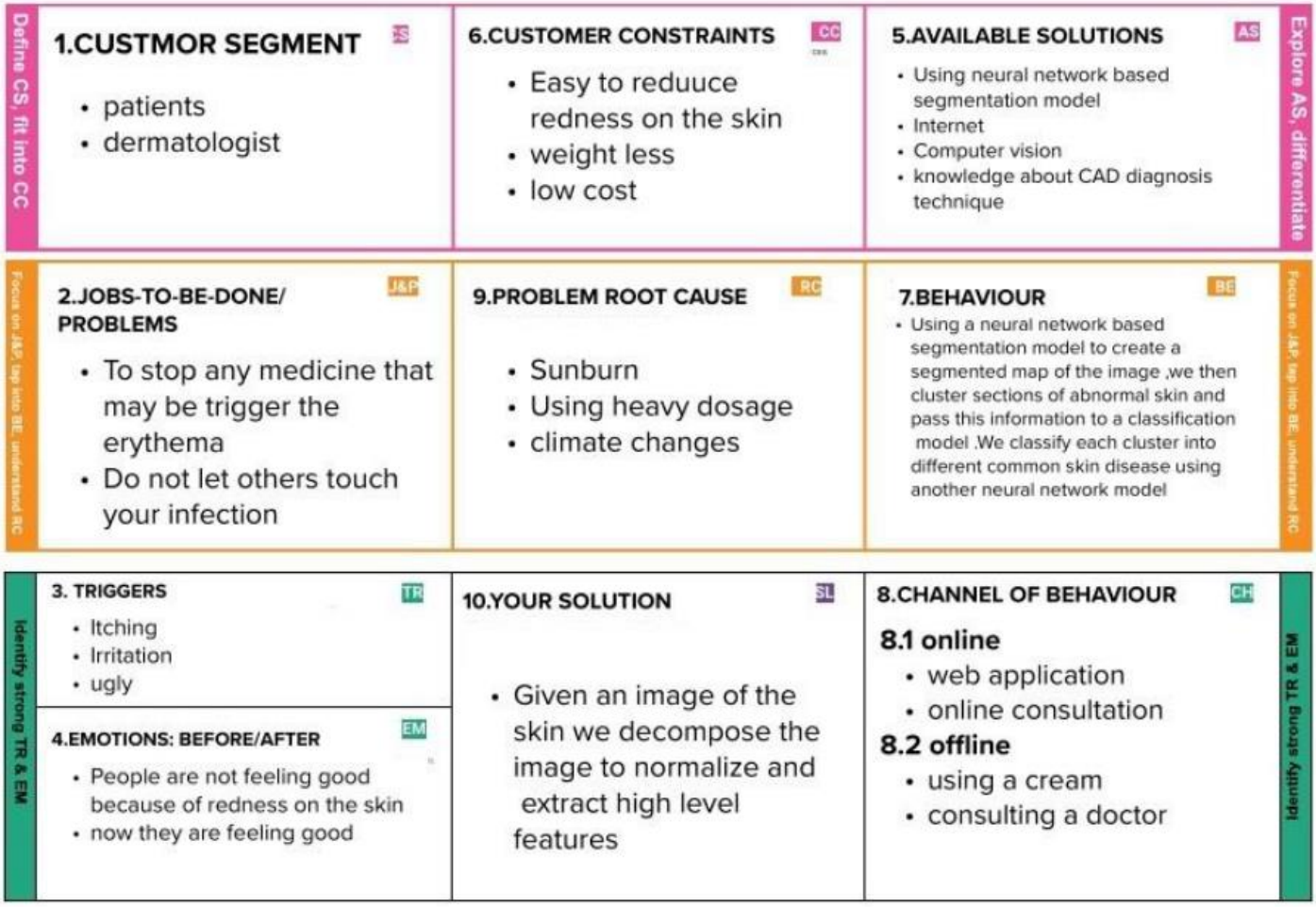
Brainstorming is a great way to generate a lot of ideas that you would not be able to generate by just sitting down with a pen and paper. The intention of brainstorming is to leverage the collective thinking of the group, by engaging with each other, listening, and building on other ideas. Conducting a brainstorm also creates a distinct segment of time when you intentionally turn up the generative part of your brain and turndown the evaluative part. You can use brainstorming throughout any design or work process, of course, to generate ideas for design solutions, but also any time you are trying to generate ideas, such as planning where to do empathy work, or thinking about product and services related to your project.

3.3 Proposed Solution

Given an image of the skin, we decompose the image to normalize and extract high-level features. Using a 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 model. Furthermore, classification of a disease is difficult due to the strong similarities between common skin disease symptoms. Therefore, it would be beneficial to use CAD with AI techniques to improve theaccuracy of dermatology Two of the more prominent approaches for skin disease segmentation and classification areclustering algorithms and support vector machines (SVMs).

The methods described above lack the ability to localize and classify multiple diseases within one image. However, we have developed a method to address this problem. In the past, skin disease models have been applied to either segmentation or classification. In this project, we sequentially combine both models by using the output of a segmentation model as input to a classification model.

3.4 Problem Solution fit



REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT

FR No.	Functional Requirement(Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Mobile Number Registration through Google Account
FR-2	User Confirmation	Confirmation via EmailConfirmation via OTP
FR-3	Patient Image CapturingProcess	Providing Access to Capture Image Through CameraProvide Access to Upload Image Through Gallery.
FR-4	Patient Medicine Reminder	Remind the patients to take their Medicine/ointmentsAt the right time through the remaining alarm.
FR-5	Suggestion Box	Patients can take suggestions from the doctors throughchats.
FR-6	Flareup Cycles	Patients can know their medicine level from doctorsthrough messages.

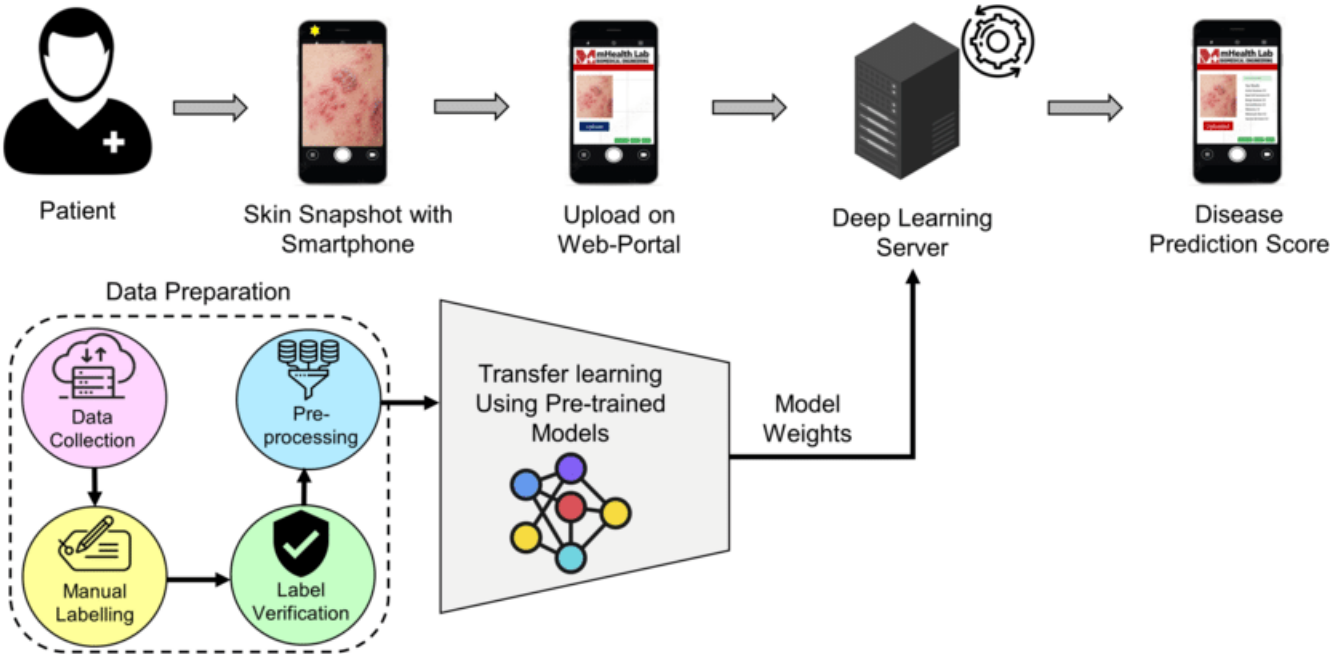
4.2 NON-FUNCTIONAL REQUIREMENTS:

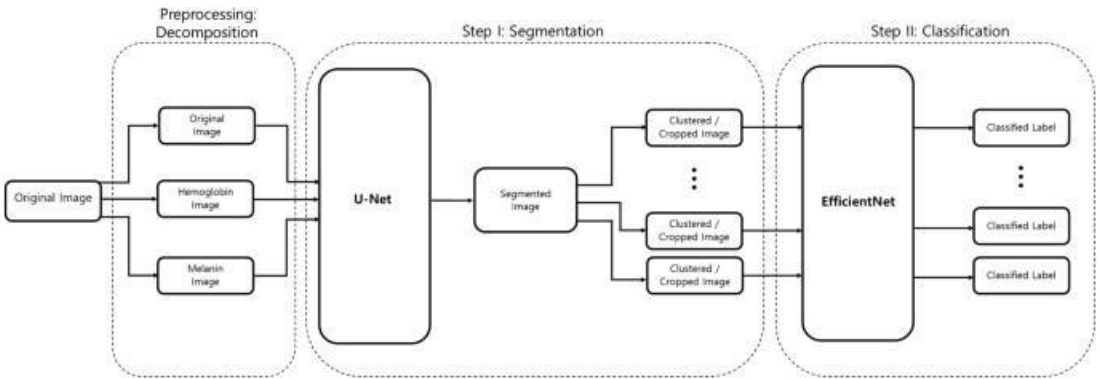
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Our Mobile phone application designed toimprove the quality of patient-held photos, and was developed togenerate and hold their own skin images to help guide their skin care.
NFR-2	**Security**	Data privacy and security practices may vary based on users and their age
NFR-3	**Reliability**	Easy to use app to get personalized answers toyour skin conditions questions
NFR-4	**Performance**	Good treatments are available for a variety of skin conditionsincluding rash, itchy skin, skin fungusetc.
NFR-5	**Availability**	Our app helps you to screenyour skin symptoms and prepare for your practitioner visit.

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.





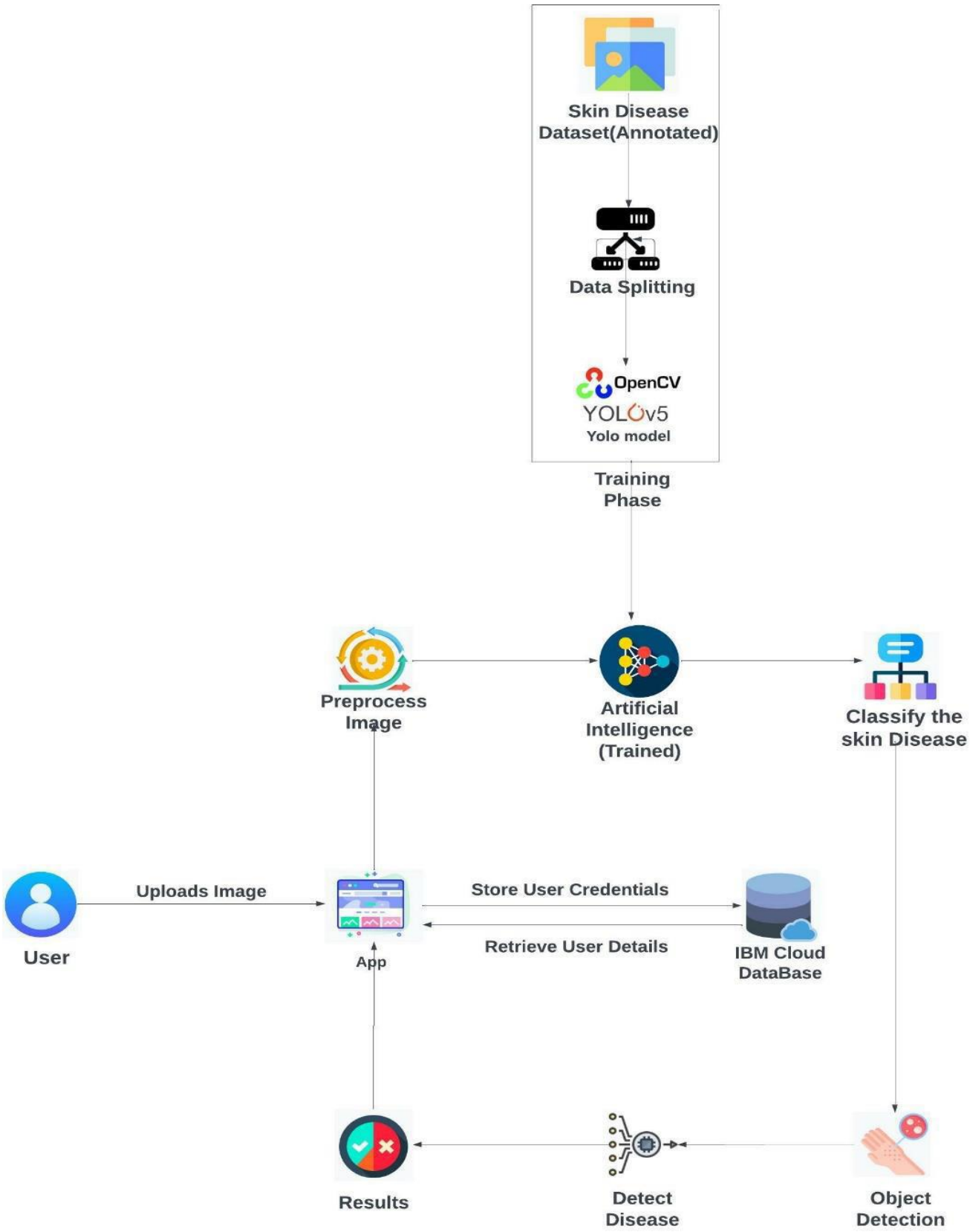
5.2 Solution & Technical Architecture

Solution architecture as well as technical architecture is a complex process with many sub-processesthat 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 Solution and Technical Architecture:



5.4 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
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Gmail	I can register &access the dashboardwith Gmail	Medium	Sprint-1
	Login	USN-4	As a user , I can log into the application by entering email & password	I can use a login id and password	High	Sprint-1
	Dashboard	USN-5	As a user, I can see the configuration in a dashboard and use them	I can use all features in dashboard	Medium	Sprint-2
Customer (Web user)	Register	USN-1	As a user, I can register for the application entering my email,password and confirming my password	I can access my account/dashboard	High	Sprint-1
	Login	USN-2	As a user , I can log into the site by entering email & password	I can use a login id and password	High	Sprint-1
CustomerCare Executive	Suggestadoctor	USN-2	Depend upon the skin disease the doctor can be suggested	Suggest a specialist doctor	Medium	Sprint-2
Administrator	Maintain	USN-1	A data given by a users are maintained by a administrator	Data is kept in safe	High	Sprint-1

PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	UserStory Number	UserStory Number	StoryPoints	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.	2	High	KailashSundaram SN Johny P Louis Santhosh Kamali T Kamali V
Sprint 1	Confirmation	USN 2	As a user, I can register for the application by entering my email, password, and confirming my password.	1	High	KailashSundaram SN Johny P Louis Santhosh Kamali T Kamali V
Sprint 1	Login	USN 3	As a user, I can login for the application throughGmail	2	Medium	KailashSundaram SN Johny P Louis Santhosh Kamali T Kamali V
Sprint 1	Login	USN 4	As a user, I can log into the application by entering email& password	2	High	KailashSundaram SN Johny P Louis Santhosh Kamali T Kamali V
Sprint 1	Dashboard	USN 5	As a user, I can log into the application by entering email& password	2	High	KailashSundaram SN Johny P Louis Santhosh Kamali T Kamali V
Sprint 1	Data input	USN 7	As a user, I can log into the application by entering email& password	2	High	KailashSundaram SN Johny P Louis Santhosh Kamali T
Sprint 1	Train Model	USN 8	As a user, I can log into the application by entering email& password	1	Medium	KailashSundaram SN Johny P Louis Santhosh Kamali T Kamali V

Sprint 1	Image processing	USN 9	As a user, I can log into the application by entering email& password	2	High	KailashSundaram SN Johny P Louis Santhosh Kamali T Kamali V
Sprint 1	Report generation	USN 10	Based on the detection of disease, report generated	2	High	KailashSundaram SN Johny P Louis Santhosh Kamali T Kamali V

6.2 Sprint Delivery Schedule

Project Tracker, Velocity & Burndown Chart:

Sprint	Total Story Points	Duration	SprintStart Date	Sprint EndDate (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	**20**	**6 Days**	**24 Oct 2022**	**07 Nov 2022**	**20**	**07 Nov 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**	**14 Nov 2022**
Sprint-4	**20**	**6 Days**	**07 Nov 2022**	**19 Nov 2022**	**20**	**19 Nov 2022**

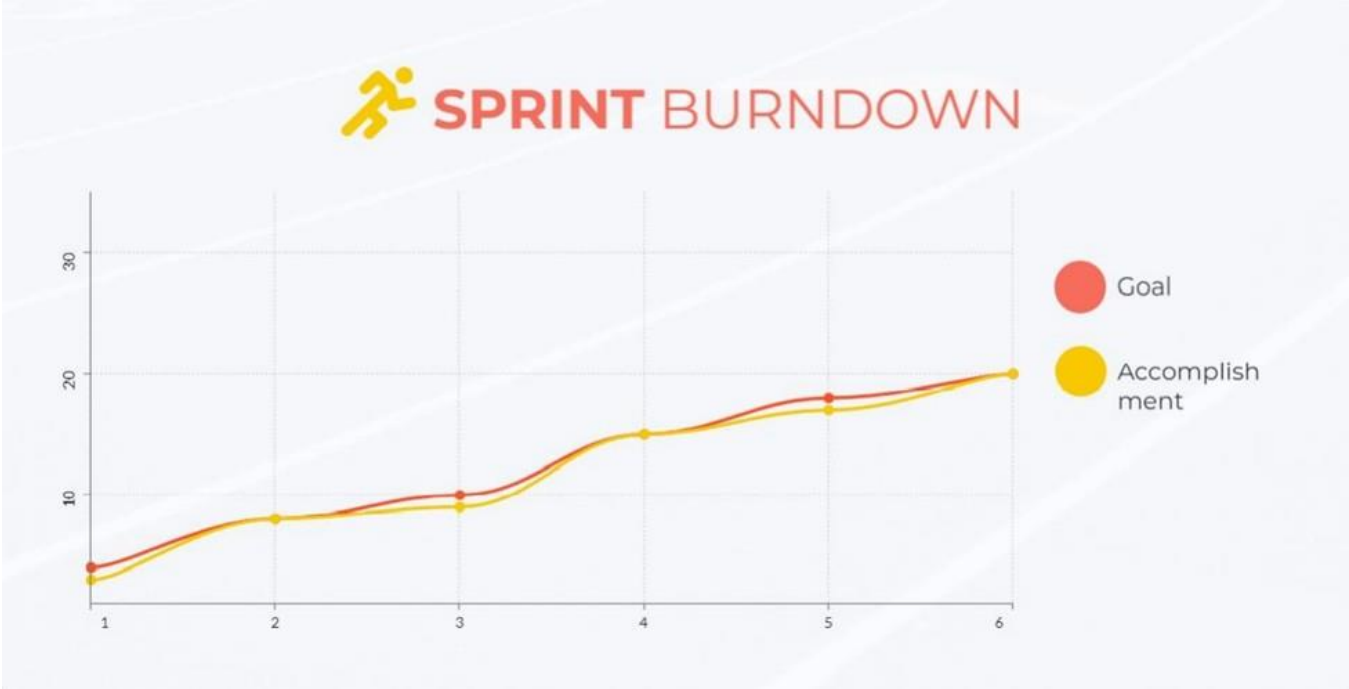
Velocity:

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)

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

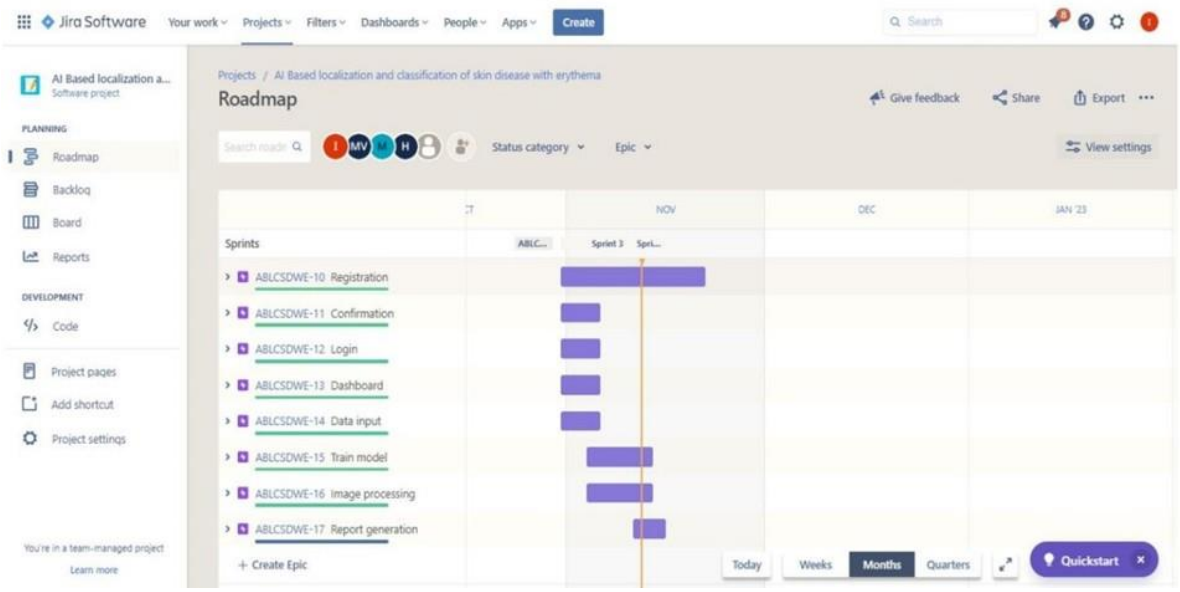
Burndown Chart:

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

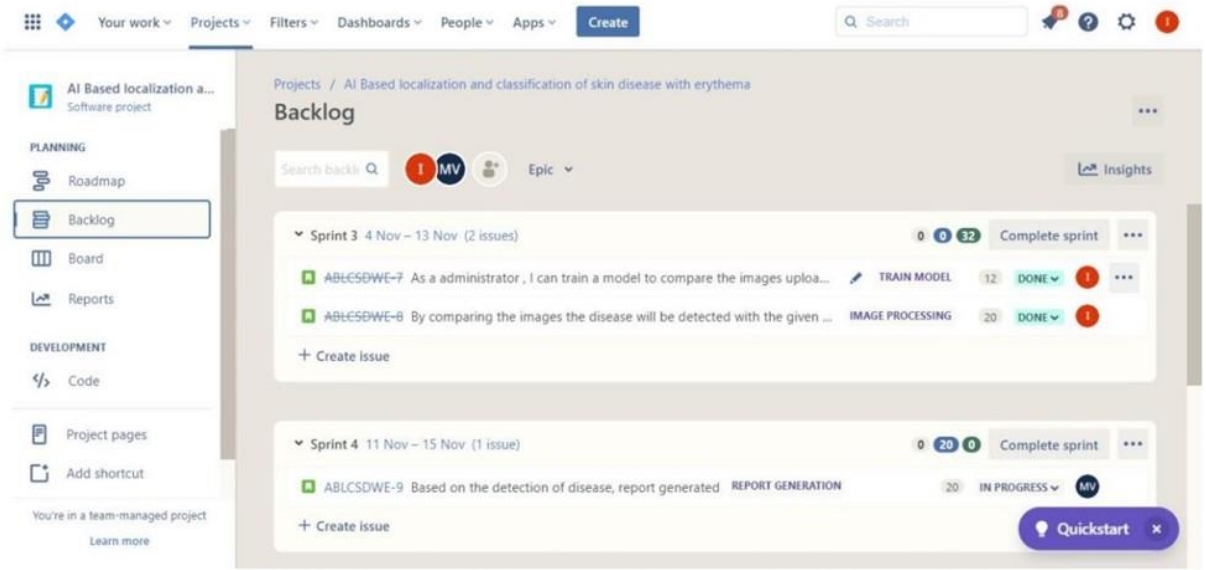


6.4 Reports from JIRA

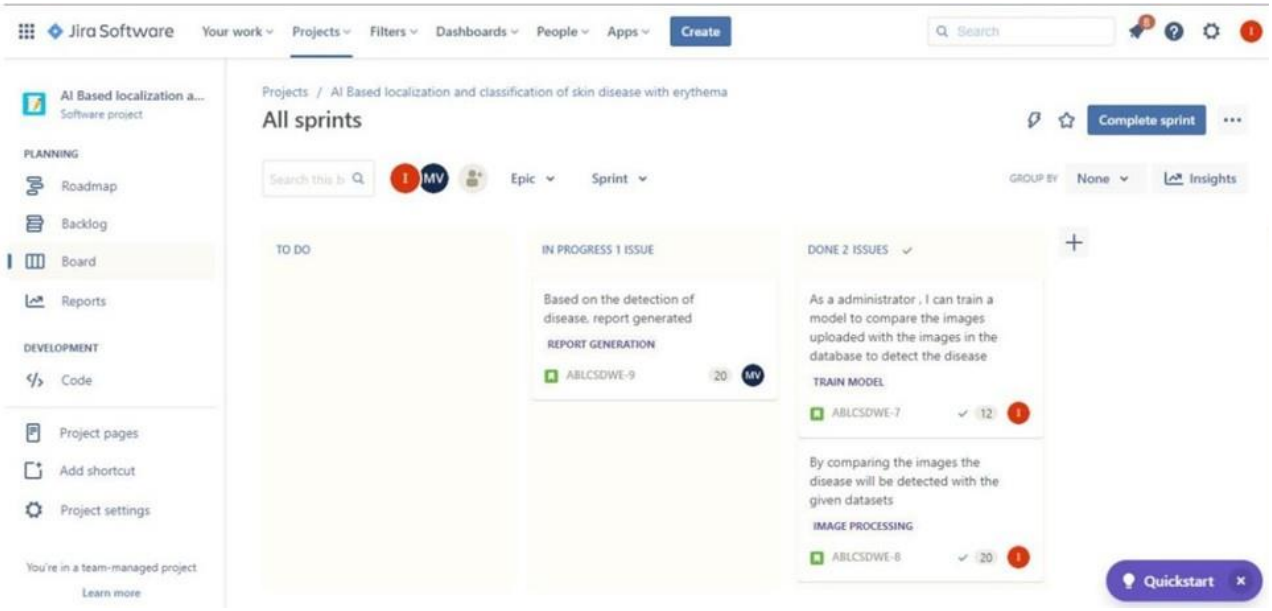
Roadmap:



Backlog:



Board:



CODING & SOLUTIONING

7.1 Feature 1

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

Code:

```
import os
import subprocess
import time
import sys
import argparse
```

```
import requests
import progressbar

FLAGS = None

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

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

    FLAGS = parser.parse_args()

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

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

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

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

7.2 Feature 2

Training Yolo

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

Code:

```
"""
MODIFIED FROM keras-yolo3 PACKAGE,
https://github.com/qqwweee/keras-yolo3
Retrain the YOLO model for your own dataset.
"""

import os
import sys
import argparse
import warnings

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


src_path = os.path.join(get_parent_dir(0), "src")
sys.path.append(src_path)

utils_path = os.path.join(get_parent_dir(1), "Utils")
sys.path.append(utils_path)
```

```
import numpy as np
import keras.backend as K
from keras.layers import Input, Lambda
from keras.models import Model
from keras.optimizers import Adam
from keras.callbacks import (
    TensorBoard,
    ModelCheckpoint,
    ReduceLROnPlateau,
    EarlyStopping,
)
from keras_yolo3.yolo3.model import (
    preprocess_true_boxes,
    yolo_body,
    tiny_yolo_body,
    yolo_loss,
)
from keras_yolo3.yolo3.utils import get_random_data
from PIL import Image
from time import time
import tensorflow.compat.v1 as tf
import pickle

from Train_Utils import (
    get_classes,
    get_anchors,
    create_model,
```

```
    create_tiny_model,
    data_generator,
    data_generator_wrapper,
    ChangeToOtherMachine,
)


keras_path = os.path.join(src_path, "keras_yolo3")
Data_Folder = os.path.join(get_parent_dir(1), "Data")
Image_Folder = os.path.join(Data_Folder, "Source_Images",
"Training_Images")
VoTT_Folder = os.path.join(Image_Folder, "vott-csv-export")
YOLO_filename = os.path.join(VoTT_Folder, "data_train.txt")

Model_Folder = os.path.join(Data_Folder, "Model_Weights")
YOLO_classname = os.path.join(Model_Folder,
"data_classes.txt")

log_dir = Model_Folder
anchors_path = os.path.join(keras_path, "model_data",
"yolo_anchors.txt")
weights_path = os.path.join(keras_path, "yolo.h5")

FLAGS = None

if __name__ == "__main__":
    # Delete all default flags
```

```
    parser =
argparse.ArgumentParser(argument_default=argparse.SUPPRESS)
    """
    Command line options
    """

    parser.add_argument(
        "--annotation_file",
        type=str,
        default=YOLO_filename,
        help="Path to annotation file for Yolo. Default is "
+ YOLO_filename,
    )
    parser.add_argument(
        "--classes_file",
        type=str,
        default=YOLO_classname,
        help="Path to YOLO classnames. Default is " +
YOLO_classname,
    )

    parser.add_argument(
        "--log_dir",
        type=str,
        default=log_dir,
        help="Folder to save training logs and trained
weights to. Default is "
        + log_dir,
```

```
    )

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

    parser.add_argument(
        "--weights_path",
        type=str,
        default=weights_path,
        help="Path to pre-trained YOLO weights. Default is "
+ weights_path,
    )
    parser.add_argument(
        "--val_split",
        type=float,
        default=0.1,
        help="Percentage of training set to be used for
validation. Default is 10%.",
    )
    parser.add_argument(
        "--is_tiny",
        default=False,
        action="store_true",
```

```
        help="Use the tiny Yolo version for better
performance and less accuracy. Default is False.",
    )
    parser.add_argument(
        "--random_seed",
        type=float,
        default=None,
        help="Random seed value to make script
deterministic. Default is 'None', i.e. non-deterministic.",
    )
    parser.add_argument(
        "--epochs",
        type=float,
        default=51,
        help="Number of epochs for training last layers and
number of epochs for fine-tuning layers. Default is 51.",
    )
    parser.add_argument(
        "--warnings",
        default=False,
        action="store_true",
        help="Display warning messages. Default is False.",
    )

    FLAGS = parser.parse_args()

    if not FLAGS.warnings:
        tf.logging.set_verbosity(tf.logging.ERROR)
```

```
        os.environ['TF_CPP_MIN_LOG_LEVEL']='3'
        warnings.filterwarnings("ignore")

    np.random.seed(FLAGS.random_seed)

    log_dir = FLAGS.log_dir

    class_names = get_classes(FLAGS.classes_file)
    num_classes = len(class_names)
    anchors = get_anchors(FLAGS.anchors_path)
    weights_path = FLAGS.weights_path

    input_shape = (416, 416)  # multiple of 32, height,
width
    epoch1, epoch2 = FLAGS.epochs, FLAGS.epochs

    is_tiny_version = len(anchors) == 6  # default setting
    if FLAGS.is_tiny:
        model = create_tiny_model(
            input_shape, anchors, num_classes,
freeze_body=2, weights_path=weights_path
        )
    else:
        model = create_model(
            input_shape, anchors, num_classes,
freeze_body=2, weights_path=weights_path
        )  # make sure you know what you freeze
```

```
    log_dir_time = os.path.join(log_dir,
"{}".format(int(time())))
    logging = TensorBoard(log_dir=log_dir_time)
    checkpoint = ModelCheckpoint(
        os.path.join(log_dir, "checkpoint.h5"),
        monitor="val_loss",
        save_weights_only=True,
        save_best_only=True,
        period=5,
    )
    reduce_lr = ReduceLROnPlateau(monitor="val_loss",
factor=0.1, patience=3, verbose=1)
    early_stopping = EarlyStopping(
        monitor="val_loss", min_delta=0, patience=10,
verbose=1
    )

    val_split = FLAGS.val_split
    with open(FLAGS.annotation_file) as f:
        lines = f.readlines()

    # This step makes sure that the path names correspond to
the local machine
    # This is important if annotation and training are done
on different machines (e.g. training on AWS)
    lines = ChangeToOtherMachine(lines, remote_machine="")
    np.random.shuffle(lines)
    num_val = int(len(lines) * val_split)
```

```
    num_train = len(lines) - num_val

    # Train with frozen layers first, to get a stable loss.
    # Adjust num epochs to your dataset. This step is enough
to obtain a decent model.
    if True:
        model.compile(
            optimizer=Adam(lr=1e-3),
            loss={
                # use custom yolo_loss Lambda layer.
                "yolo_loss": lambda y_true, y_pred: y_pred
            },
        )

        batch_size = 32
        print(
            "Train on {} samples, val on {} samples, with
batch size {}.".format(
                num_train, num_val, batch_size
            )
        )
        history = model.fit_generator(
            data_generator_wrapper(
                lines[:num_train], batch_size, input_shape,
anchors, num_classes
            ),
            steps_per_epoch=max(1, num_train // batch_size),
            validation_data=data_generator_wrapper(
```

```
                lines[num_train:], batch_size, input_shape,
anchors, num_classes
            ),
            validation_steps=max(1, num_val // batch_size),
            epochs=epoch1,
            initial_epoch=0,
            callbacks=[logging, checkpoint],
        )
        model.save_weights(os.path.join(log_dir,
"trained_weights_stage_1.h5"))

        step1_train_loss = history.history["loss"]

        file = open(os.path.join(log_dir_time,
"step1_loss.npy"), "w")
        with open(os.path.join(log_dir_time,
"step1_loss.npy"), "w") as f:
            for item in step1_train_loss:
                f.write("%s\n" % item)
        file.close()

        step1_val_loss =
np.array(history.history["val_loss"])

        file = open(os.path.join(log_dir_time,
"step1_val_loss.npy"), "w")
        with open(os.path.join(log_dir_time,
"step1_val_loss.npy"), "w") as f:
```

```
            for item in step1_val_loss:
                f.write("%s\n" % item)
        file.close()

    # Unfreeze and continue training, to fine-tune.
    # Train longer if the result is unsatisfactory.
    if True:
        for i in range(len(model.layers)):
            model.layers[i].trainable = True
        model.compile(
            optimizer=Adam(lr=1e-4), loss={"yolo_loss":
lambda y_true, y_pred: y_pred}
        )  # recompile to apply the change
        print("Unfreeze all layers.")

        batch_size = (
            4  # note that more GPU memory is required after
unfreezing the body
        )
        print(
            "Train on {} samples, val on {} samples, with
batch size {}.".format(
                num_train, num_val, batch_size
            )
        )
        history = model.fit_generator(
            data_generator_wrapper(
```

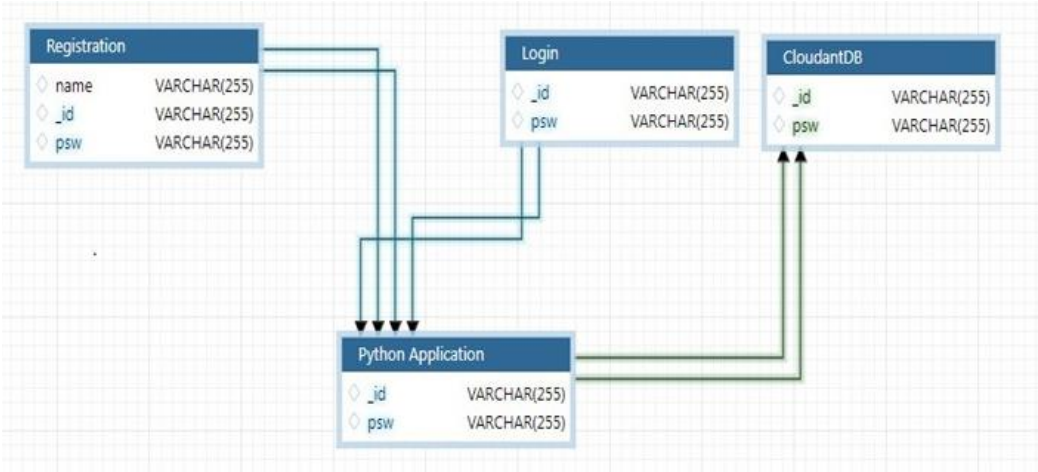
```
                lines[:num_train], batch_size, input_shape,
anchors, num_classes
            ),
            steps_per_epoch=max(1, num_train // batch_size),
            validation_data=data_generator_wrapper(
                lines[num_train:], batch_size, input_shape,
anchors, num_classes
            ),
            validation_steps=max(1, num_val // batch_size),
            epochs=epoch1 + epoch2,
            initial_epoch=epoch1,
            callbacks=[logging, checkpoint, reduce_lr,
early_stopping],
        )
        model.save_weights(os.path.join(log_dir,
"trained_weights_final.h5"))
        step2_train_loss = history.history["loss"]

        file = open(os.path.join(log_dir_time,
"step2_loss.npy"), "w")
        with open(os.path.join(log_dir_time,
"step2_loss.npy"), "w") as f:
            for item in step2_train_loss:
                f.write("%s\n" % item)
        file.close()

        step2_val_loss =
np.array(history.history["val_loss"])
```

```
        file = open(os.path.join(log_dir_time,
"step2_val_loss.npy"), "w")
        with open(os.path.join(log_dir_time,
"step2_val_loss.npy"), "w") as f:
            for item in step2_val_loss:
                f.write("%s\n" % item)
        file.close()
```

7.3 Database Schema

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

Diagram:



TESTING

8.1 Test Case

TestCase No.	Action	ExpectedOutput	Actual Output	Result
1	Register forthe website	Stores name,email, and password in Database	Stores name, email, and password in Database	Pass
2	Login to the website	Giving theright credentials,results in asuccessful login.	Giving the right credentials, results in asuccessful login.	Pass
3	Detectingthe disease	It should predict the disease	It should predict thedisease	Pass

8.2 User Acceptance Testing

User Acceptance Testing (UAT) is a type of testing performed by the end user or the client to verify/accept the software system before moving the software application to the production environment. UAT is done
in the final phase of testing after functional, integration and system testing is done.

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

RESULTS

9.1 Performance Metrics.

Performance metrics are a part of every machine learning pipeline. They tell you if you're making progress, and put a number on it. All machine learning models, whether it's linear regression, or a SOTA technique like BERT, need a metric to judge performance.

- ✓ R-CNN and YOLO is used for model creation. The mean average precision compares thedetected with new box and finally returns a returns a score.

- ✓ Accuracy which we obtained was Training Accuracy – 86% and Validation Accuracy – 94%

- ✓ The confidence score which we got are Class Detected – 93% and Confidence Score – 90%

ADVANTAGES & DISADVANTAGES

Advantages:

- ✓ Artificial Intelligence helps to solve complex problems that requiredifficult calculations and can be done without any error.
- ✓ Perform Repetitive Jobs and faster decision taking.
- ✓ AI can streamline workflows.
- ✓ AI systems eliminate the risk of human error, producing a more accurate result.

Disadvantages:

- ✓ High production cost
- ✓ Lacking Out of Box Thinking and unemployment.
- ✓ AI is making humans lazy with its applications automating the majority of the work
- ✓ AI cannot be accessed and utilized akin to human intelligence, but it can store infinite data.

CONCLUSION

The AI model facilitates the modelling of images with adequate precision.
Utilizing segmentation, the Yolo model assisted us in identifying the locations of the disease. By boosting the speed of diagnosis with a minimal error rate, AI-based methods reduce manual stress and tension. These models are user-friendly and adaptable, enabling self-detection of disease.

FUTURE SCOPE

The detection of skin diseases is making positive strides thanks to AI-based research. Despite several assertions that deep learning algorithms outperform professionals in the detection of skin disease, these algorithms face a number of obstacles before becoming a comprehensive diagnostic system. Due to the fact that such trials are conducted in controlled environments, algorithms are never evaluated in the actual diagnosis of patients. Real-world diagnosis requires considering a patient's ethnicity, skin, hair, and eye colour, occupation, illness, medications, existing sun damage, the number of nevi, and lifestyle habits (such as sun exposure, smoking, and alcohol consumption), as well as clinical history, response to previous treatments, and other information from the patient's medical records.

However, current deep learning models rely primarily on imaging data from patients. In addition, such algorithms frequently provide a danger of misdiagnosis when applied to skin lesions or disorders not present in the training dataset.

Computer vision and dermatological associations must collaborate to improve present AI solutions and increase the diagnostic precision of methods used to diagnose skin conditions. AI has the ability to bring about a paradigm shift in the detection of skin diseases, resulting in a cost-effective, remotely accessible, and precise healthcare solution

GitHub:

https://github.com/IBM-EPBL/IBM-Project-29167-1660121810