AI-BASED LOCALIZATION AND CLASSIFICATION OF SKIN DISEASE WITH ERYTHEMA

DOMAIN NAME : ARTIFICIAL INTELLIGENCE

TEAM ID : PNT2022TMID29580

BATCH :B9-3A5E

TEAM LEADER : SUBISHKA R [510419205031]

TEAM MEMBER : HEMALATHA J [510419205009]

TEAM MEMBER :PRIYA S [510419205016]

TEAM MEMBER : SHALINI S [510419205023]

TEAM MEMBER : SOWBARNIKA M S [510419205028]

Table of Contents

Chapter No.	Title	Page No.
1.	**INTRODUCTION**	**6**
1.1	Project Overview	6
1.2	Purpose	6
2.	**LITERATURE SURVEY**	**6**
2.1	Existing problem	6
2.2	References	6
2.3	Problem Statement Definition	7
3.	**IDEATION & PROPOSED SOLUTION**	**8**
3.1	Empathy Map Canvas	8
3.2	Ideation & Brainstorming	8
3.3	Proposed Solution	12

4.	**REQUIREMENT ANALYSIS**	**14**
4.1	Functional requirement	14
4.2	Non-Functional requirements	15
5.	**PROJECT DESIGN**	**16**
5.1	Data Flow Diagrams	16
5.2	Solution & Technical Architecture	17
5.3	User Stories	19
6.	**PROJECT PLANNING & SCHEDULING**	**20**
6.1	Sprint Planning & Estimation	20
6.2	Sprint Delivery Schedule	23
6.3	Reports from JIRA	25
7.	**CODING & SOLUTIONING**	28
7.1	Feature 1	28
7.2	Feature 2	29
7.3	Database Schema	41

8.	**TESTING**	**42**
8.1	Test Cases	42
8.2	User Acceptance Testing	43
9.	**RESULTS**	**44**
9.1	Performance Metrics	44
10.	**ADVANTAGES & DISADVANTAGES**	**45**
11.	**CONCLUSION**	**46**
12.	**FUTURE SCOPE**	**46**
13.	**APPENDIX**	**47**
13.1	Source Code	47
13.2	GitHub & Project Demo Link	64

List of Figures

Figure	Name of the Figure	Page number
1.	Empathy Map Canvas	8
2.	Ideation and Brainstorming	9
3.	Problem Solution Fit	13
4.	Data Flow Diagrams	16
5.	Solution Architecture Diagram	17
6.	Technical Architecture	18
7.	Burndown Chart	24
8.	Roadmap	24
9.	Sprint - 1	25
10.	Sprint - 2	25
11.	Sprint - 3	26
12.	Sprint - 4	27

1 INTRODUCTION

1.1 PROJECT OVERVIEW

Now days 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 on the body including spreading of the infection from one individual to the other. The skin disease can be prevented by investigating the infected region at an early stage. The characteristics 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 disease.

1.2 PURPOSE

To overcome this problem we are building a model which is used for the prevention and early detection of skin cancer, psoriasis. Building, 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 images will be sent to the trained model. The model analysis the image and detect whether the person is having skin disease or not.

2 LITERATURE SURVEY

2.1 Existing Problem

Most of these Diseases are Dangerous and harmful ,if not treated at an initial stage. It may cause Spreading in people.The Yolo v3 detector is the primary method for pre-screening skin lesions and detecting erythema. YOLO is an algorithm that detects and recognizes various objects in real-time pictures.This means that prediction for the entire image is done in a single algorithm run.Yolo-V3 boasts good performance over a wide range of input resolutions.

2.2 REFERENCES

https://pubmed.ncbi.nlm.nih.gov/34546174/

https://www.hindawi.com/journals/cin/2022/6138490/

https://aip.scitation.org/doi/abs/10.1063/5.0074207

https://www.sciencedirect.com/science/article/abs/pii/S0895611118303355

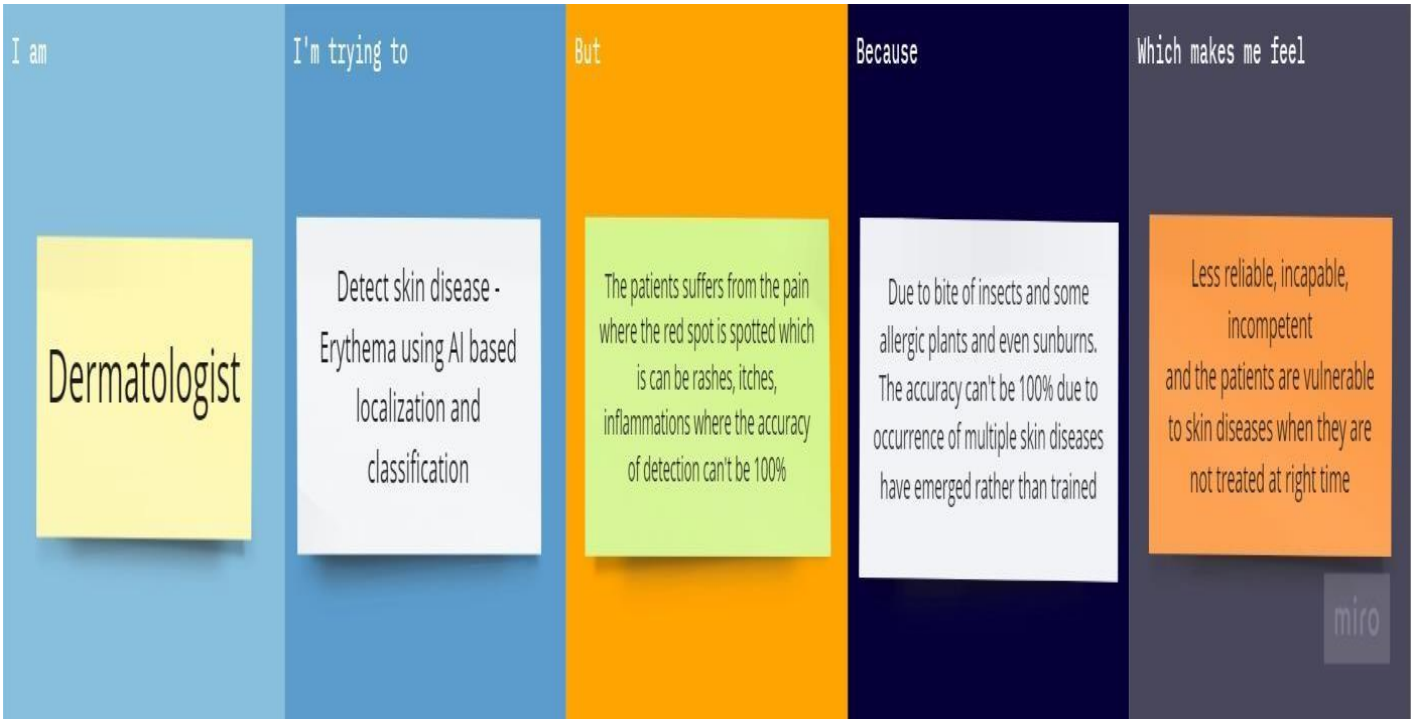
https://www.hindawi.com/journals/cmmm/2021/9998379/

https://www.sciencedirect.com/science/article/abs/pii/S1566253519300867

2.3 Problem Statement Definition

Customer Problem Statement

Doctor Problem Statement



3 IDEATION & PROPOSED SOLUTION

3.1 Empathy Map canvas

3.2 Ideation & Brainstorming

IBM PROJECT AI DOMAIN

AI BASED LOCALIZATION AND CLASSIFICATION OF SKIN DISEASE WITH ERYTHEMA

This 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.The model analyses the image and detect whether the person is having skin disease or not.

10 minutes to prepare

1 hour to collaborate

2-8 people recommended

Before you collaborate

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

10 minutes

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

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

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

Open article →

1 Define your problem statement

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

5 minutes

PROBLEM

This is used for the prevention and early detection of skin cancer, psoriasis. 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.

Key rules of brainstorming

To run an smooth and productive session

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

2 Brainstorm

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

10 minutes

TIP

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

SUBIKSHA R

- Training dataset on both healthy and affected patients
- Early detection and diagnosis
- Feasible prediction using inputs from smartphone cameras
- Reports should be issued

HEMALATHA J

- Saves time and cost
- Multiple image processing
- Live chat box
- Available 24*7

SOWBARNIKA M S

- User friendly
- Trust worthy
- Affordable
- Customized dashboard

PRIYA S

- Convenient customer support
- Ensure security
- Prediction should be presented in organized manner
- Helplines should be provided

SHALINI S

- Clean the dataset
- Faster prediction as user uploads the image
- Recommending suitable medication
- Prediction of skin disease by analyzing colour

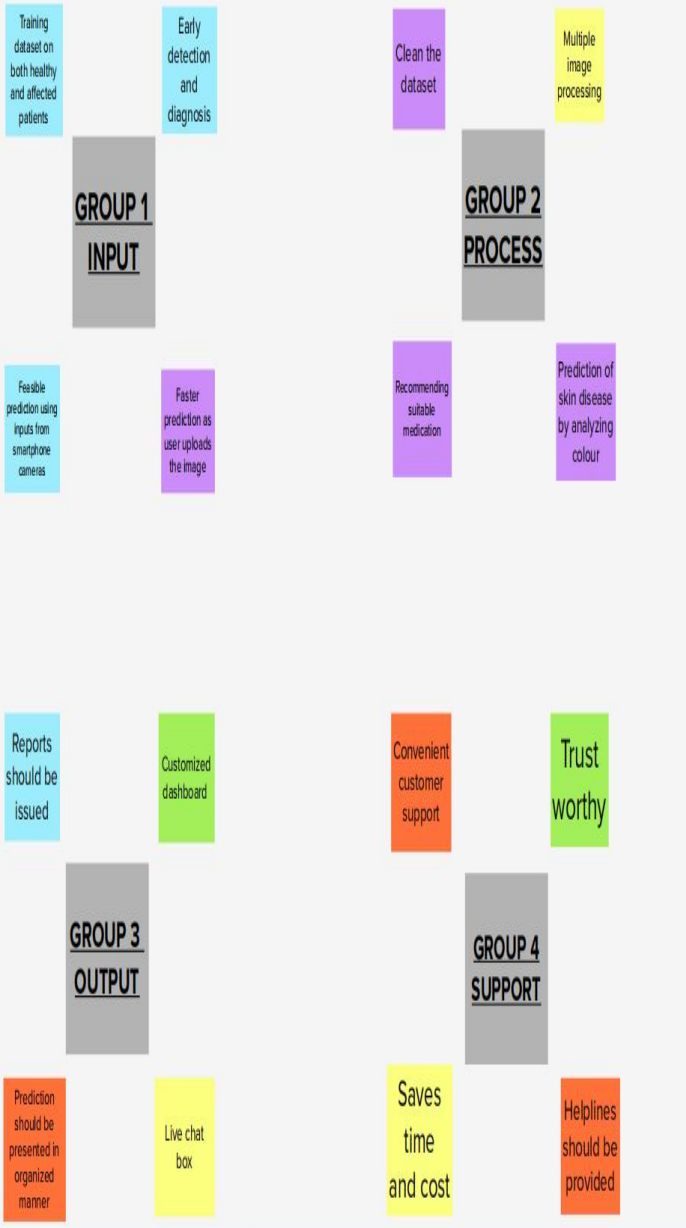
3 Group ideas

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

20 minutes

TIP

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



4

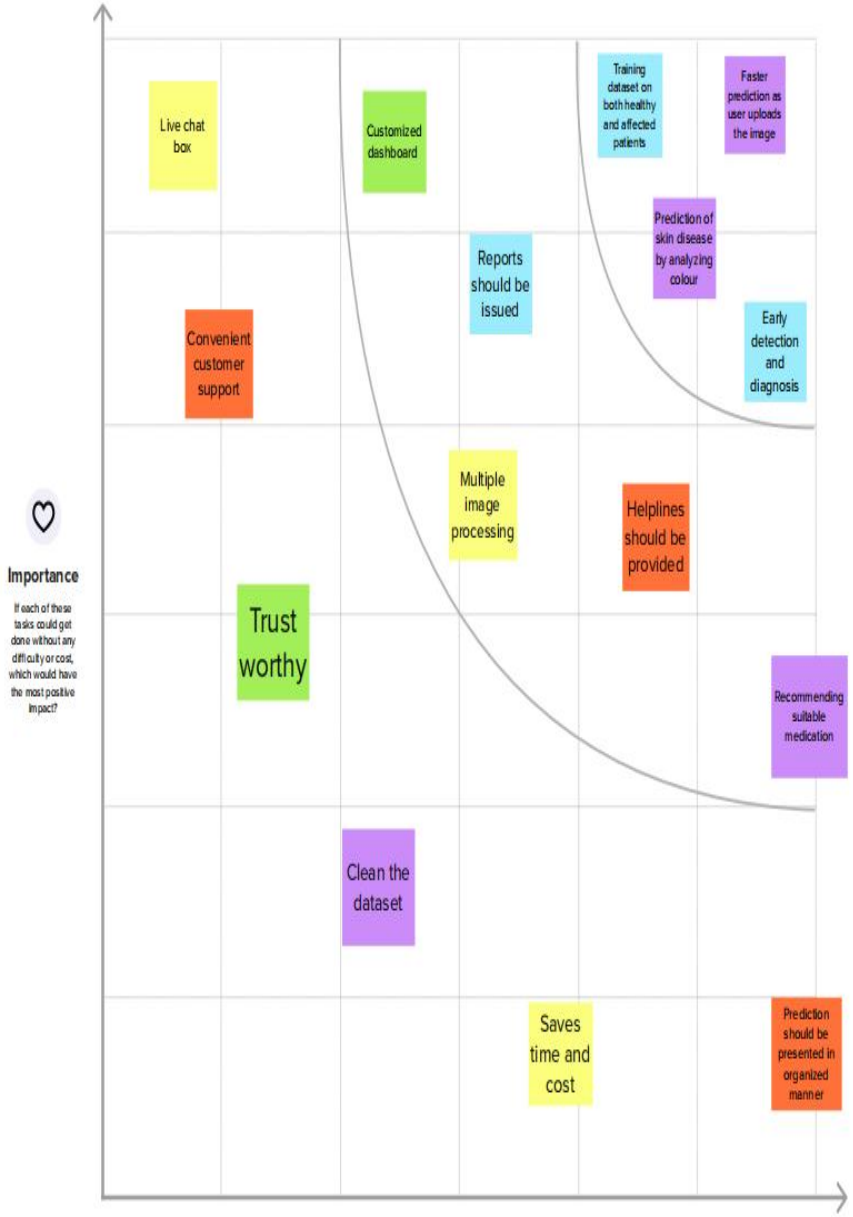
Prioritize

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

20 minutes

TIP

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



After you collaborate

You can export the mural as an image or pdf to share with members of your company who might find it helpful.

Quick add-ons

A **Share the mural**
Share a view link to the mural with stakeholders to keep them in the loop about the outcomes of the session.

B **Export the mural**
Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save in your drive.

Keep moving forward

Strategy blueprint
Define the components of a new idea or strategy.
Open the template →

Customer experience journey map
Understand customer needs, motivations, and obstacles for an experience.
Open the template →

Strengths, weaknesses, opportunities & threats
Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan.
Open the template →

Share template feedback

3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Most of these Diseases are Dangerous and harmful ,if not treated at an initial stage. It may cause Spreading in people
2.	Idea / Solution description	Find out the problem of skin disease and prevent or must get treated from model analysis at an early stage.
3.	Novelty / Uniqueness	Complexity manifests itself in the large number and variety of both saturated and unsaturated fatty chain synthesized by Human Skin.
4.	Social Impact / Customer Satisfaction	Social Impact people suffer from psoriasis and coarseness Skin Disease. Hope they get treated as well as possible.
5.	Business Model (Revenue Model)	Skin observation caught the attention of big business that spotted an opportunity to make some quick money through investment.
6.	Scalability of the Solution	It might be different like psoriasis and fungal skin infections. Get their treatment for solution

3.4 Problem Solution Fit

4.Requirement Analysis

4.1 Functional Requirement:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form.
FR-2	User Confirmation	Confirmation via Email.
FR-1	User Profile	Filling the profile page after logging in.
FR-1	Uploading Dataset (Skin images)	Images of the skin have to be uploaded
FR-1	Requesting solution	Uploaded images is compared with the pre-defined Model and solution is generated.
FR-1	Downloading Solution The solution in p	The solution in pdf format which contains the analysis of the image and detect whether the person is having skin disease or not.

4.2 Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The system allows the user to perform the tasks easily and efficiently and effectively.
NFR-2	**Security**	Assuring all data inside the system or its part will be protected against malware attacks or unauthorized access.
NFR-3	**Reliability**	The website and app will recover from failure quickly, it takes time as the application is running in single server.
NFR-4	**Performance**	Response Time and Net Processing Time is fast.
NFR-5	**Availability**	The system will be available up to 95% of the time.
NFR-6	**Scalability**	The website and app should be scalable.

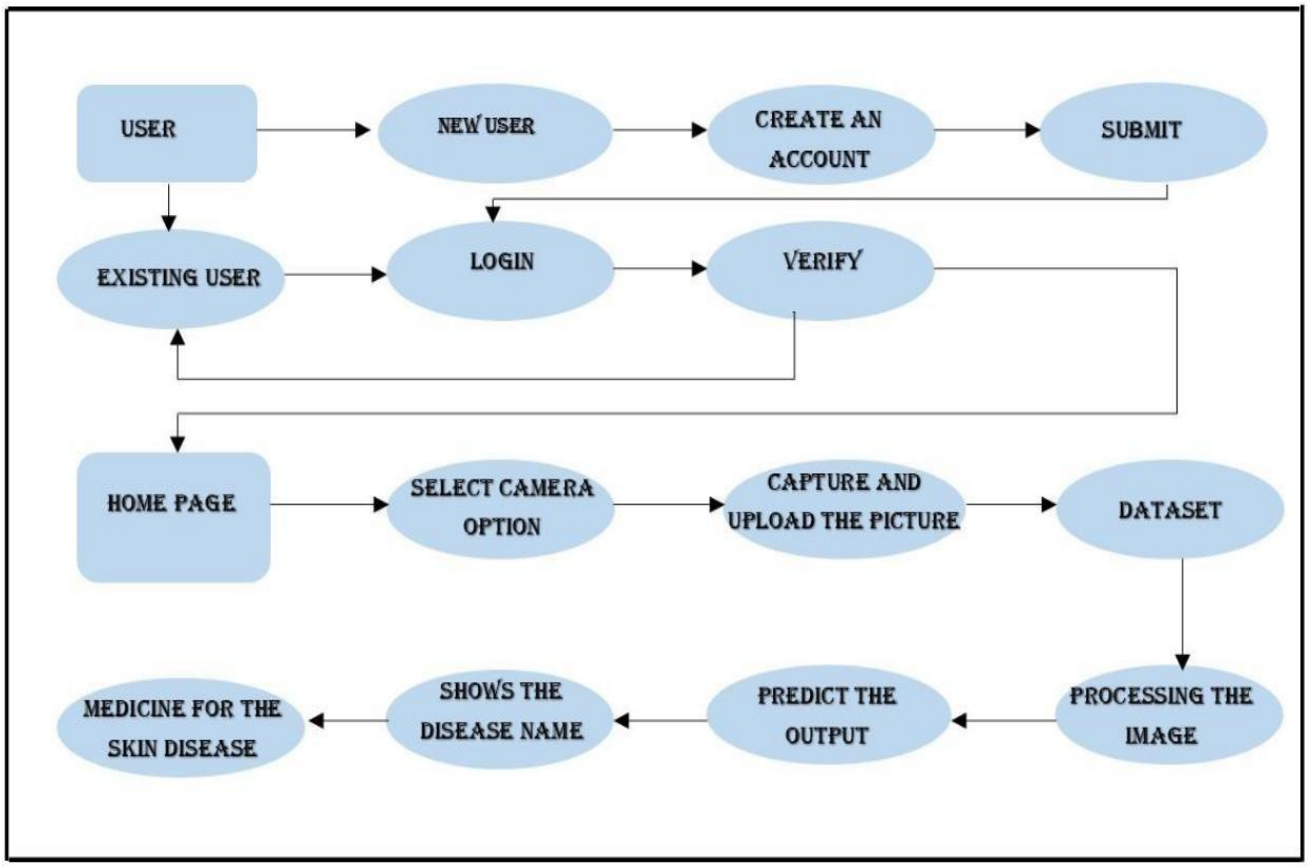
5. PROJECT DESIGN

5.1 Data Flow Diagram

- 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.

EXAMPLE(simplified);

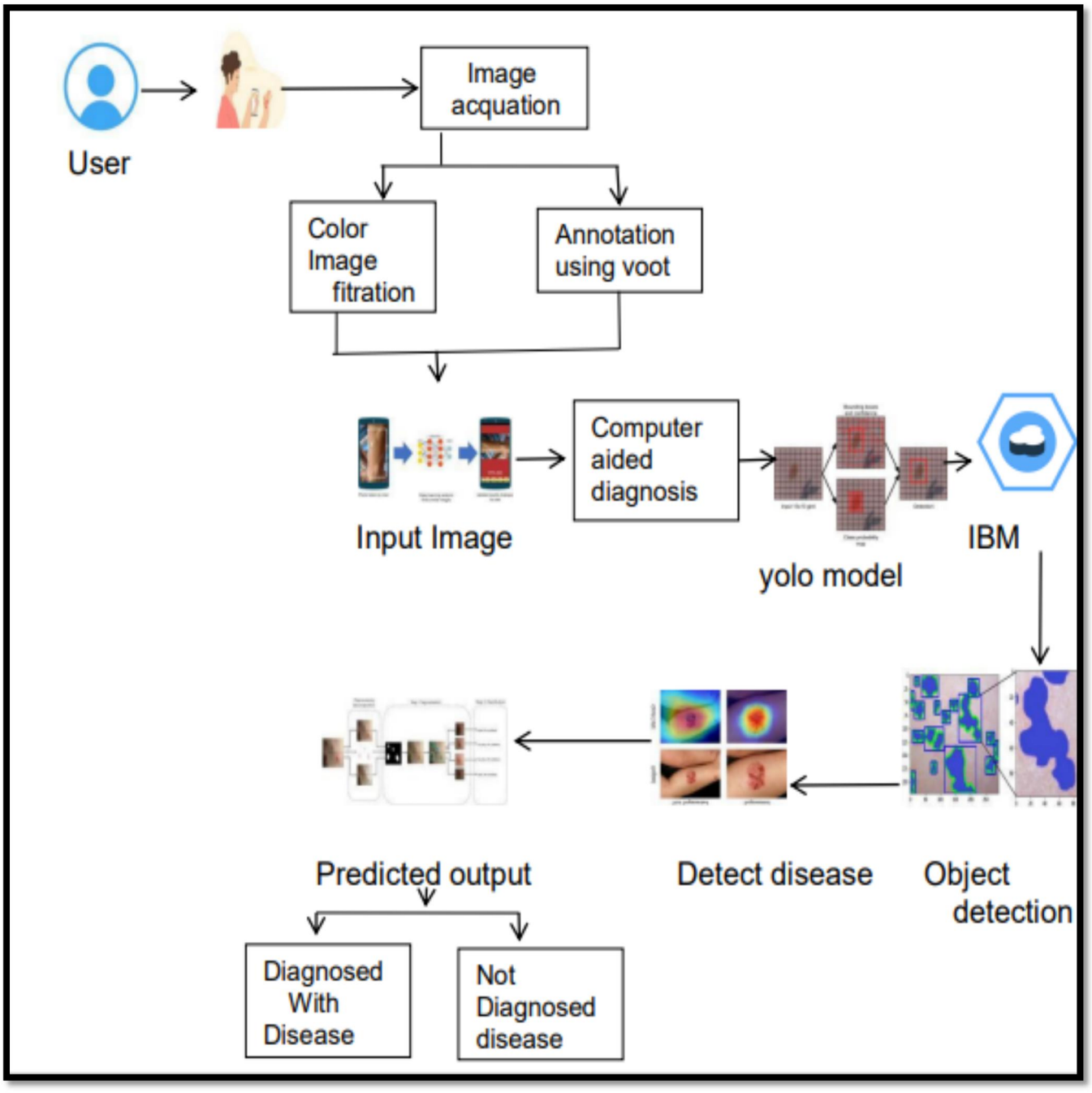
- Most of these Diseases are Dangerous and harmful ,if not treated at an initial stage
- Social Impact people suffer from psoriasis and coarseness Skin Disease.
- Problem of skin disease and prevent or must get treated from model analysis at an early s stage.
- Social Impact people suffer from psoriasis and coarseness Skin Disease



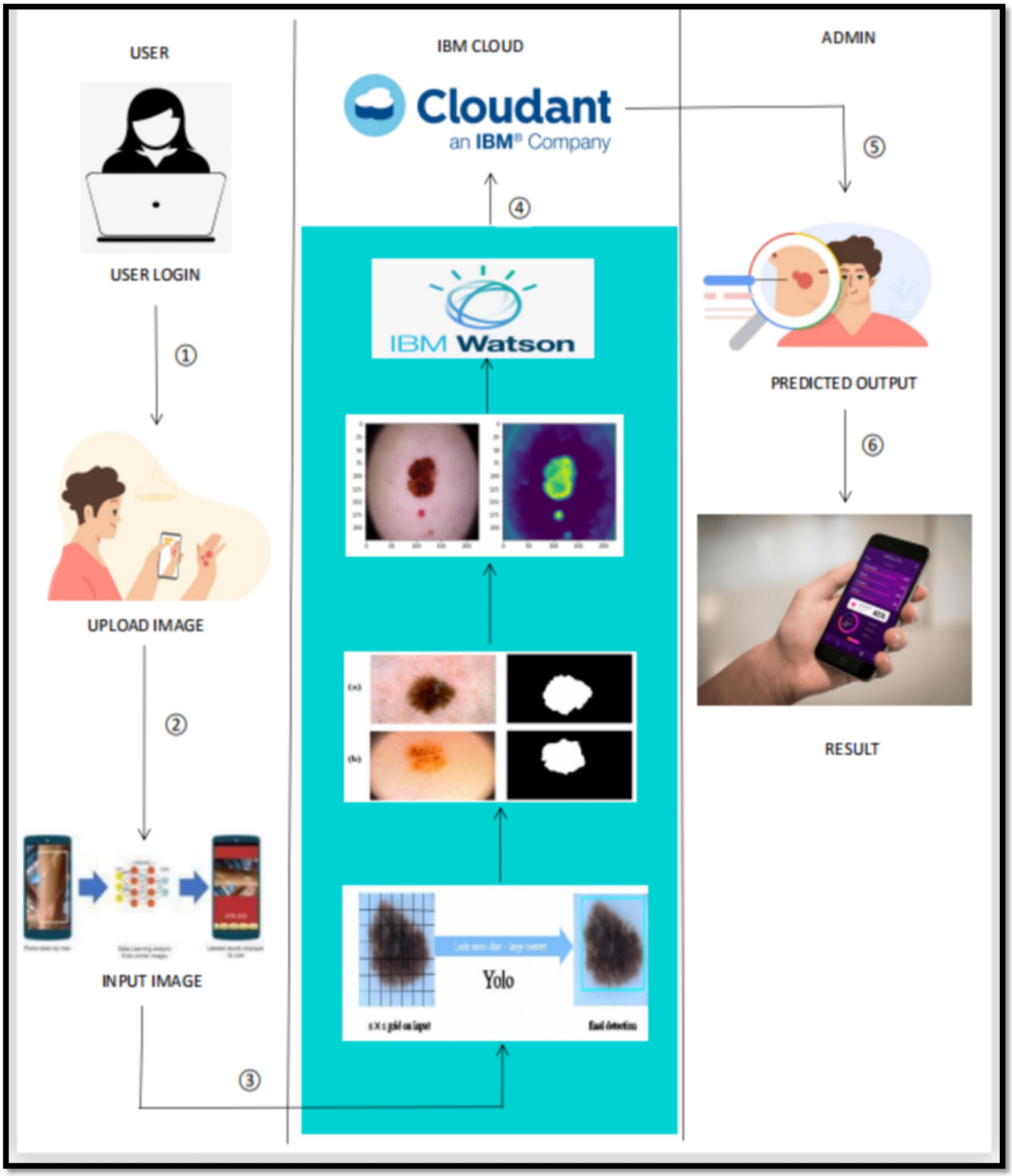
5.2 Solution Architecture

- Solution architecture is a complex process – with many sub-processes – that bridges the gap between skin disease with erythema. Its goals are to:
- To predict the skin disease using images.
- Although computer-aided diagnosis (CAD) is used to improve the quality of diagnosis in various medical fields such as mammography and colonography.
- To detect disease from the images using IBM Cloudantdb
- The skin disease is classified the diagonised with disease to predicted the output

Solution Architecture Diagram



Technical Architecture



5.3 User stories

User Type	(Epic)Functional Requirement		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 Facebook	I can register & access the dash board with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail		Medium	Sprint-1
	login	USN-5	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard	USN-5	As a user, I can Access my Dashboard.		Medium	Sprint-3
Customer (Web user)	Register	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-4

Customer Care Executive	Solution	USN-5	Responding to each email you receive can Responding to each email you receive can	Offer a solution for how your company can improve the customer's experience.	High	Sprint-3

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	Install python IDE(Spyder/PyCharm IDE is ideal to complete this project)	USN-1	To Install and refer Anaconda and Pycharm for Installation steps	**8**	High	Subiksha. R Shalini. S
	Install Microsoft's Visual Object Tagging Tool(VOTT)	USN-2	Head to VOTT Download and Install the version for your OS.	**7**	High	Subiksha. R Shalini. S
	Download YOLO project Structure	USN-3	Now you need to download the structure of the project to build your model.	**5**	medium	Subiksha. R Shalini. s
	Create Database From Scratch	USN-4	Now we are going to collect the images of different skin disease from Google	**6**	High	Subiksha. R Shalini. S

Sprint 2	Create A project in VOTT(Microsoft's Visual Object Tagging Tool)	USN-5	Create A New project and its called Annotations. Highly Recommended to use.	**7**	low	Subiksha. R Shalini. S Hemalatha. J
	Create A project in VOTT-Part 2	USN-6	Under source section choose Add Connection and put images as Display Name . Target connection choose the same folder as for source.	**8**	Medium	Subiksha. R Shalini. S Hemalatha. J
	Create A project in VOTT-Part 3	USN-7	CRTL+E to export the project The folder of CSV file called[Annotations export-csv].	**9**	High	Subiksha. R Shalini. S Hemalatha. J
	Create A project in VOTT-Part 4		As a final step, Convert the VOTT csv format to the YOLOv3 format.	**7**	Medium	Subiksha. R Shalini. S Hemalatha. J
	Download And Convert Pre-Trained Weight	USN-8	Using the Training images located in yolo structures/Data structures.	**5**	High	Subiksha. R Shalini. S Hemalatha. J
	Train YOLOv3 Detector	USN-9	To start the Training, To Run the Training script from within the Yolo structure directory.	**6**	High	Subiksha. R Shalini. S Hemalatha. J
Sprint 3	Register and Login To IBM Cloud	USN-10	Register to IBM cloud:-Link Sign in with your Credentials:-Link	**9**	High	Subiksha. R Shalini. S Hemalatha. J Priya .S Sowbarnika .M.S
	Create Service Instance	USN-11	Log in to your IBM cloud account, and click on Catalog	**7**	Medium	Subiksha. R Shalini. S Priya. S

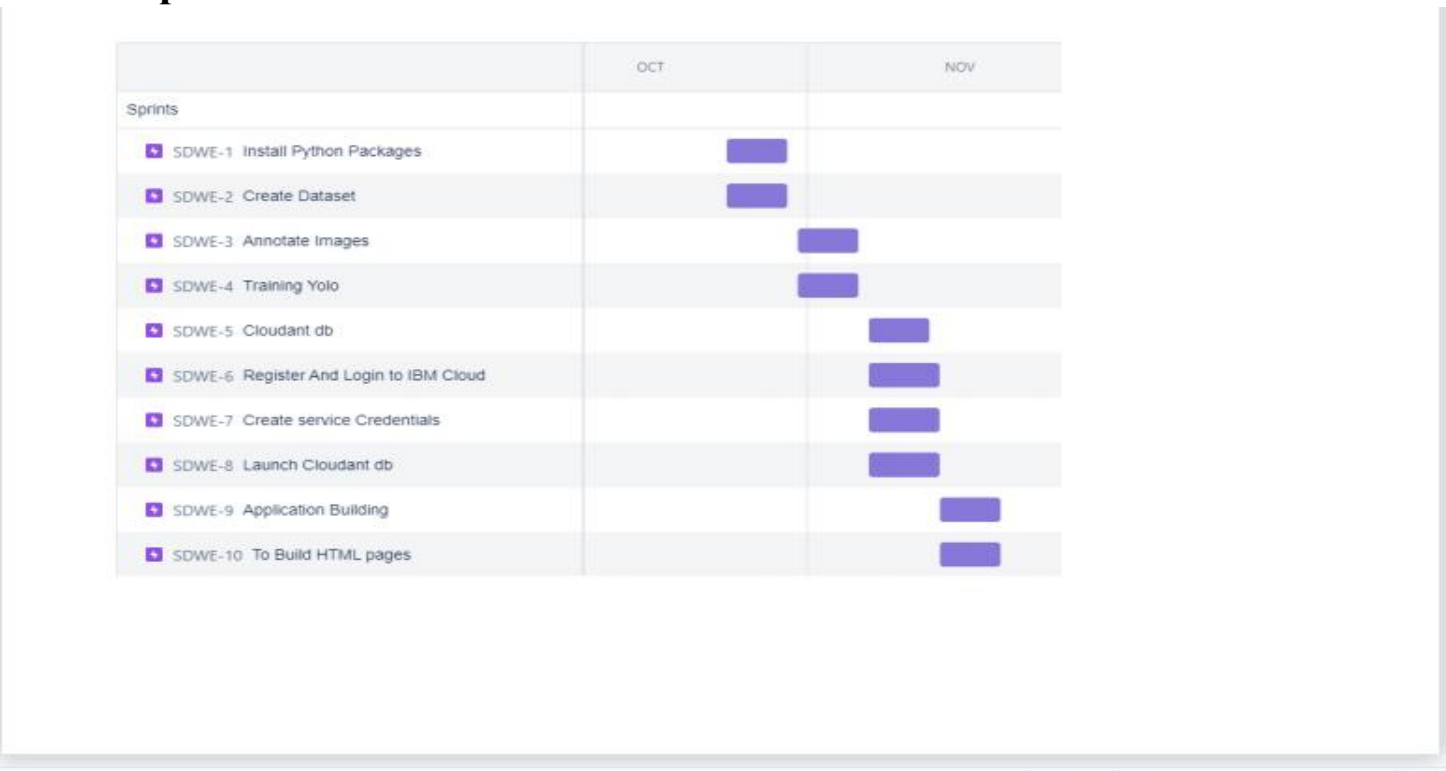
	Creating Service Credentials	USN-12	To create the connection information that your application needs to connect to the instance, click New credential	**8**	High	Subiksha. R Shalini. S Priya. S

	Launch Cloudant DB	USN-13	If you are a new user you will find empty database and it will create.		Medium	Subiksha. R Shalini. S Priya. s
	Create Database	USN-14	In order to manage a connection from a local system. IBM cloud identity& Access Management enables you to securely authenticate users and control access.		Medium	Subiksha. R Shalini. S Priya. s
Sprint 4	Building HTML pages	USN-15	For this project create three HTML files and save them in the templates folder.		Medium	Subiksha. R Shalini. S Priya. S Hemalatha. J Sowbarnika. M.S
	Build python code	USN-16	Creating a function get_parent_dir() to get parent directory.		Medium	Subiksha. R Shalini. S Sowbarnika.M.S
	Run The Application	USN-17	Open the anaconda prompt from the start menu. Now type “python app.py” command.		High	Subiksha. R Shalini. S Priya. S Hemalatha. J Sowbarnika.M.S

6.2 sprint delivery schedule

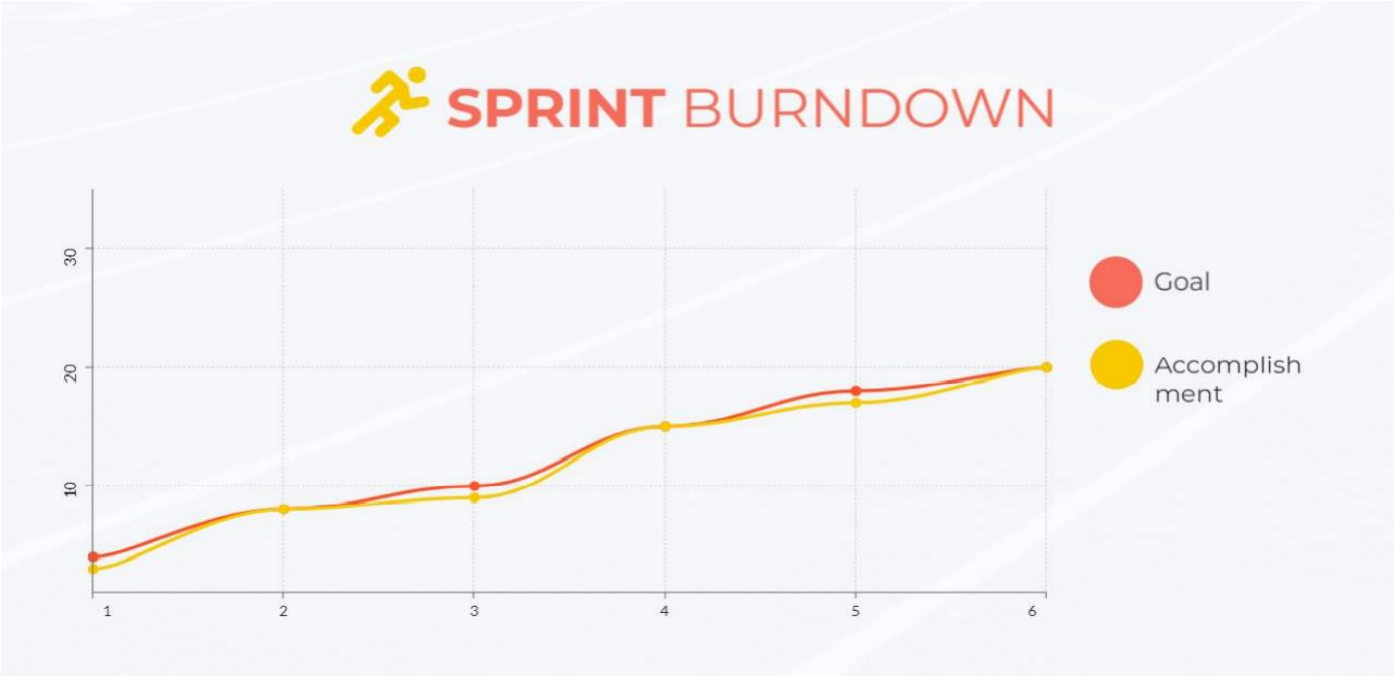
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date(Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date(Actual)
Sprint-1	**26**	6 Days	24 Oct 2022	29 Oct 2022	**26**	30 Oct 2022
Sprint-2	**42**	6 Days	31 Nov 2022	05 Nov 2022	**42**	06 Nov 2022
Sprint-3	**37**	6 Days	07 Nov 2022	12 Nov 2022	**37**	13 Nov 2022
Sprint-4	**27**	6 Days	14 Nov 2022	19 Nov 2022	**27**	19 Nov 2022

RoadMap:



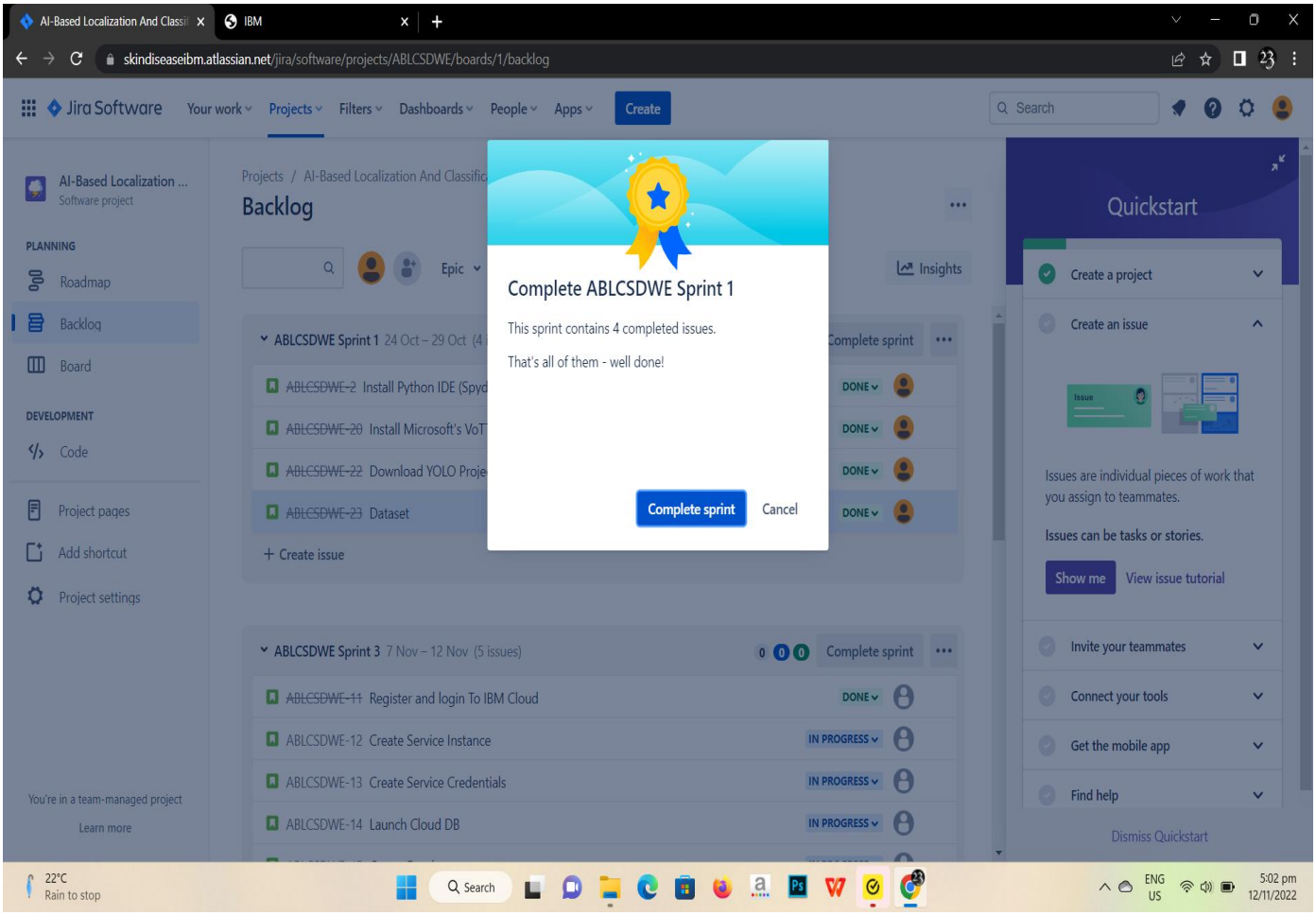
Burndown Chart:

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.

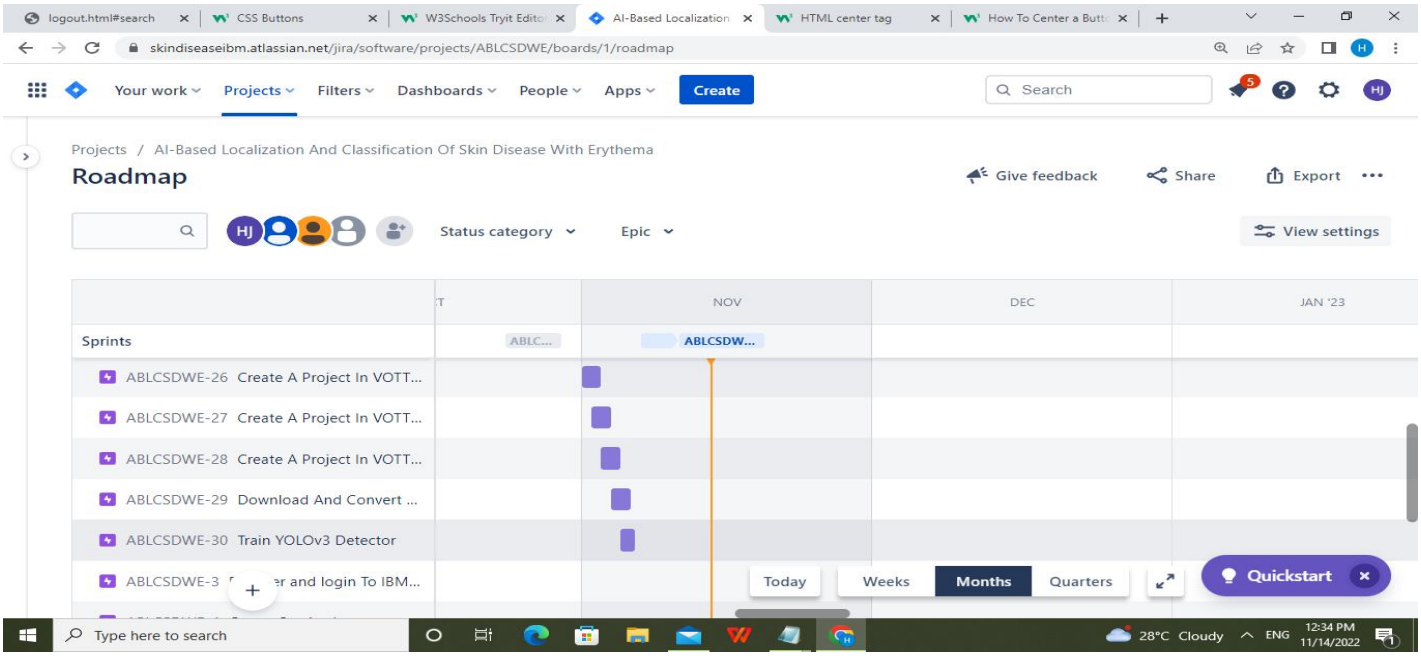


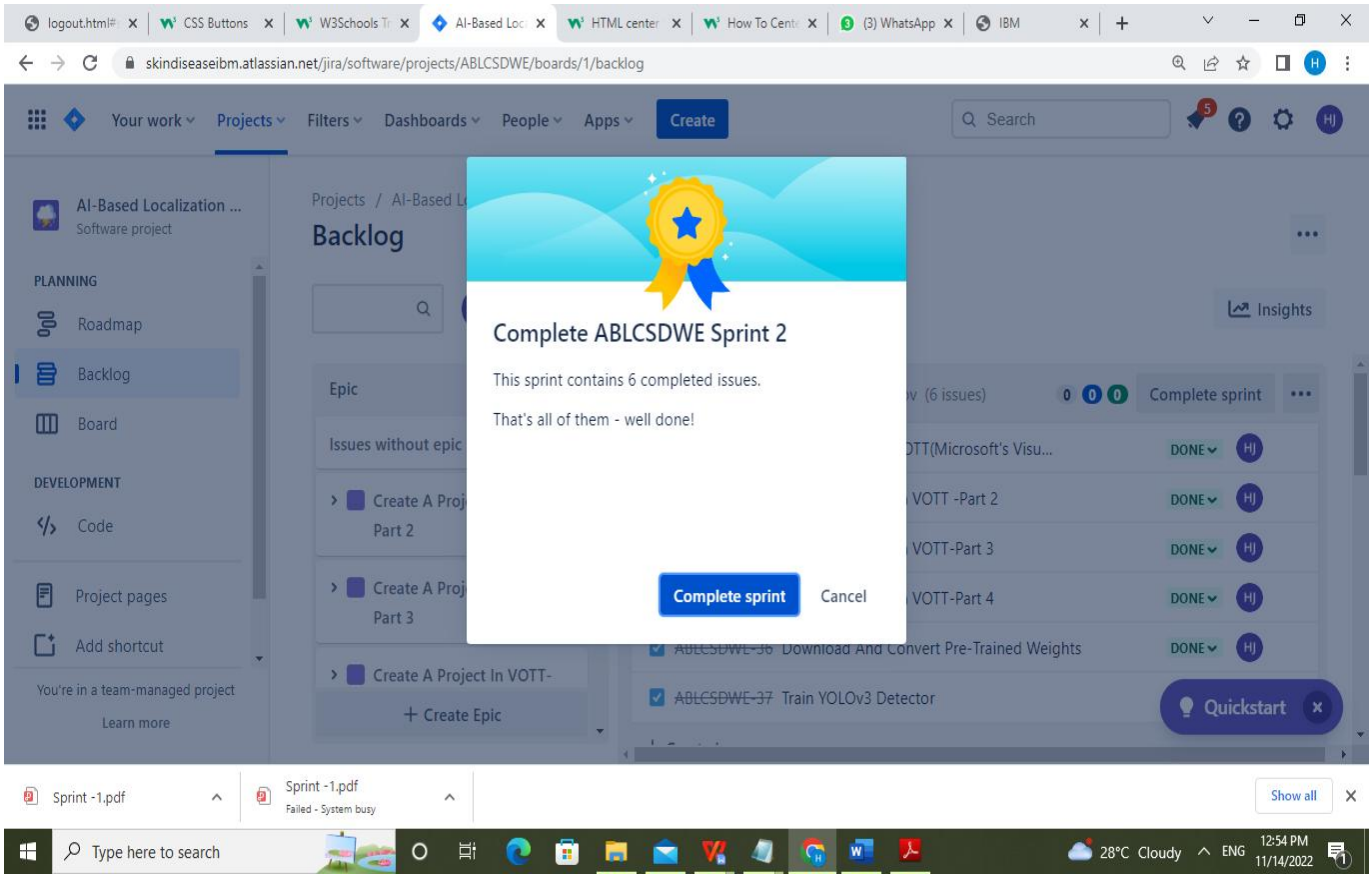
6.3 Reports From Jira

Sprint-1

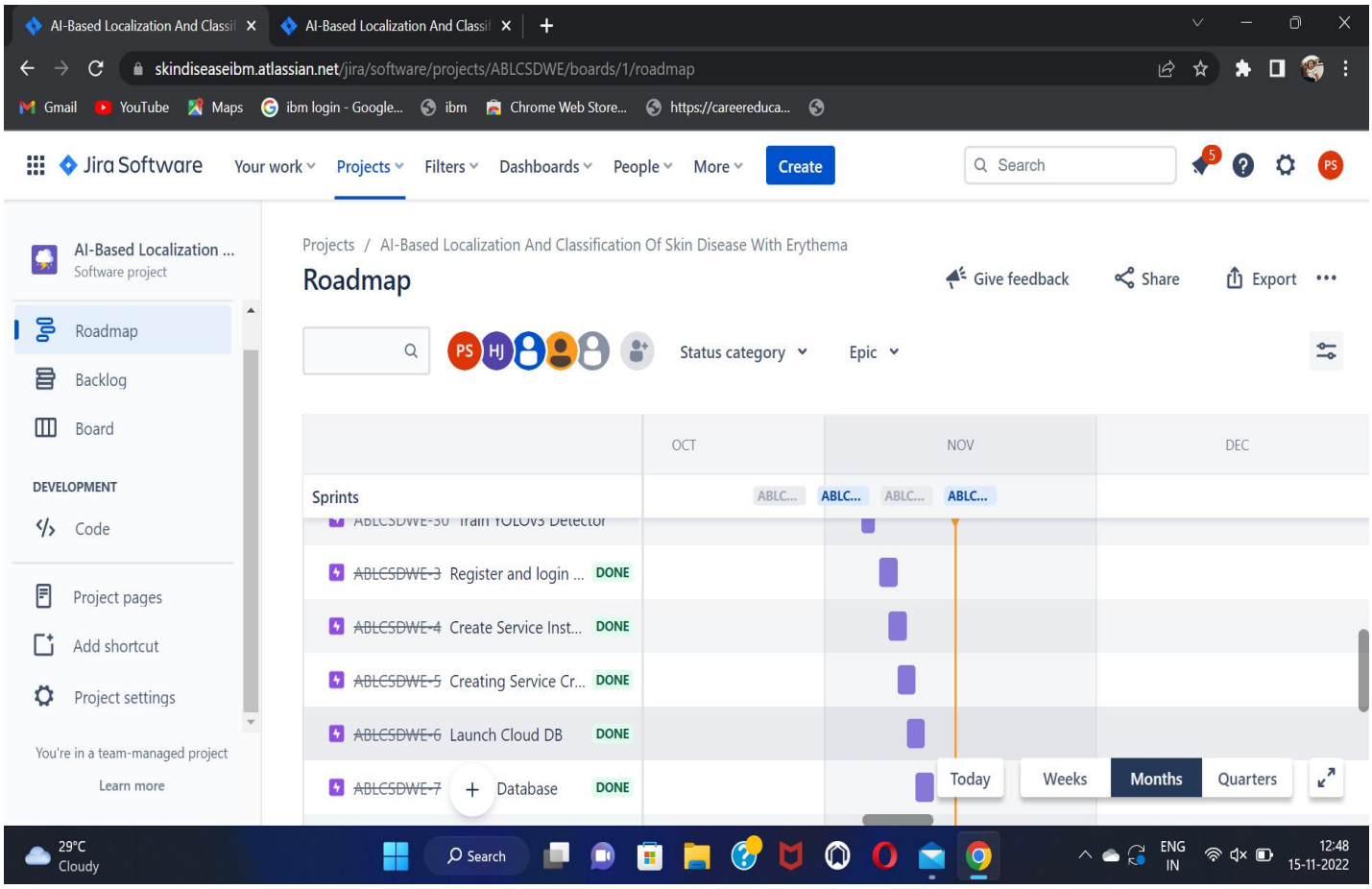


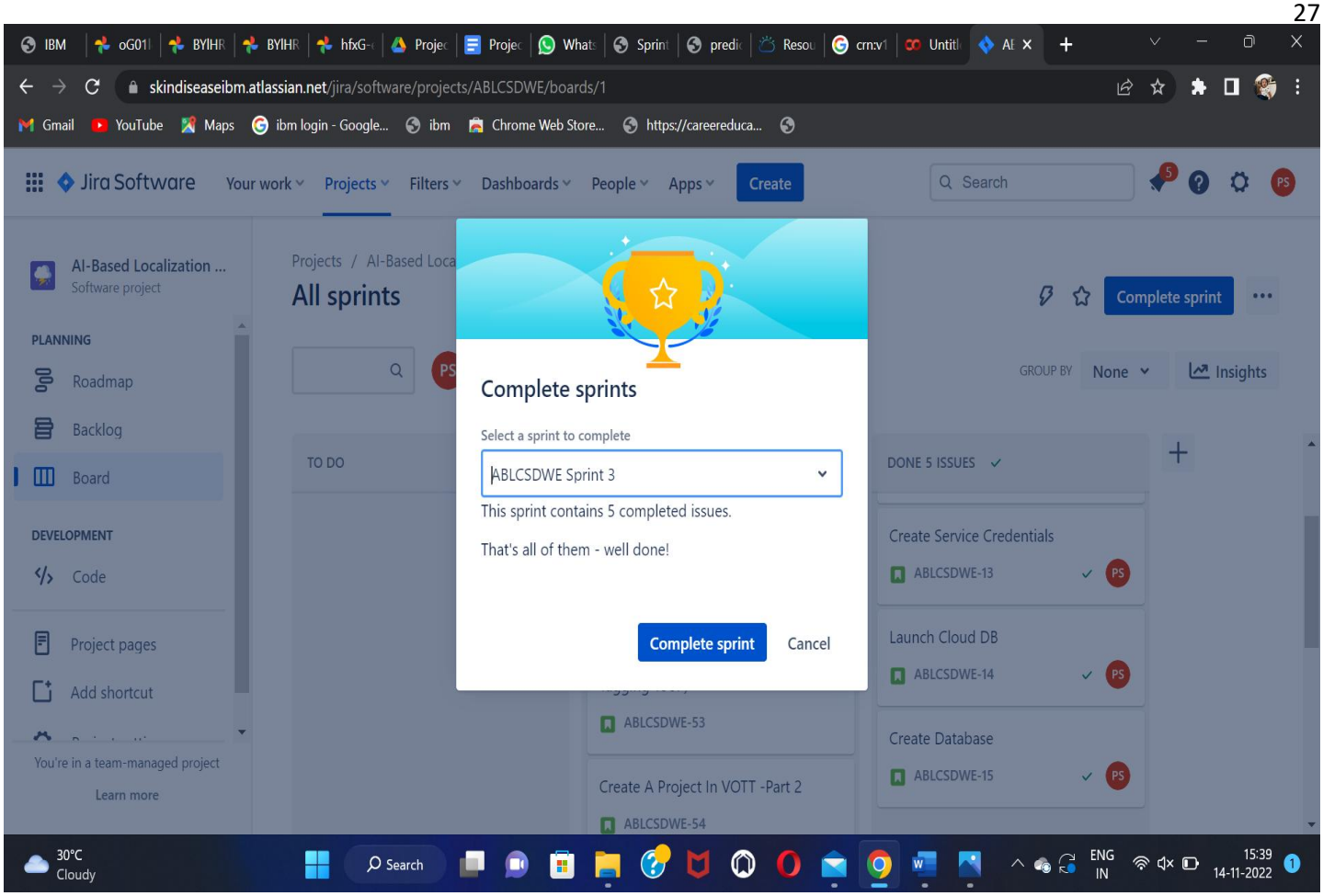
Sprint-2



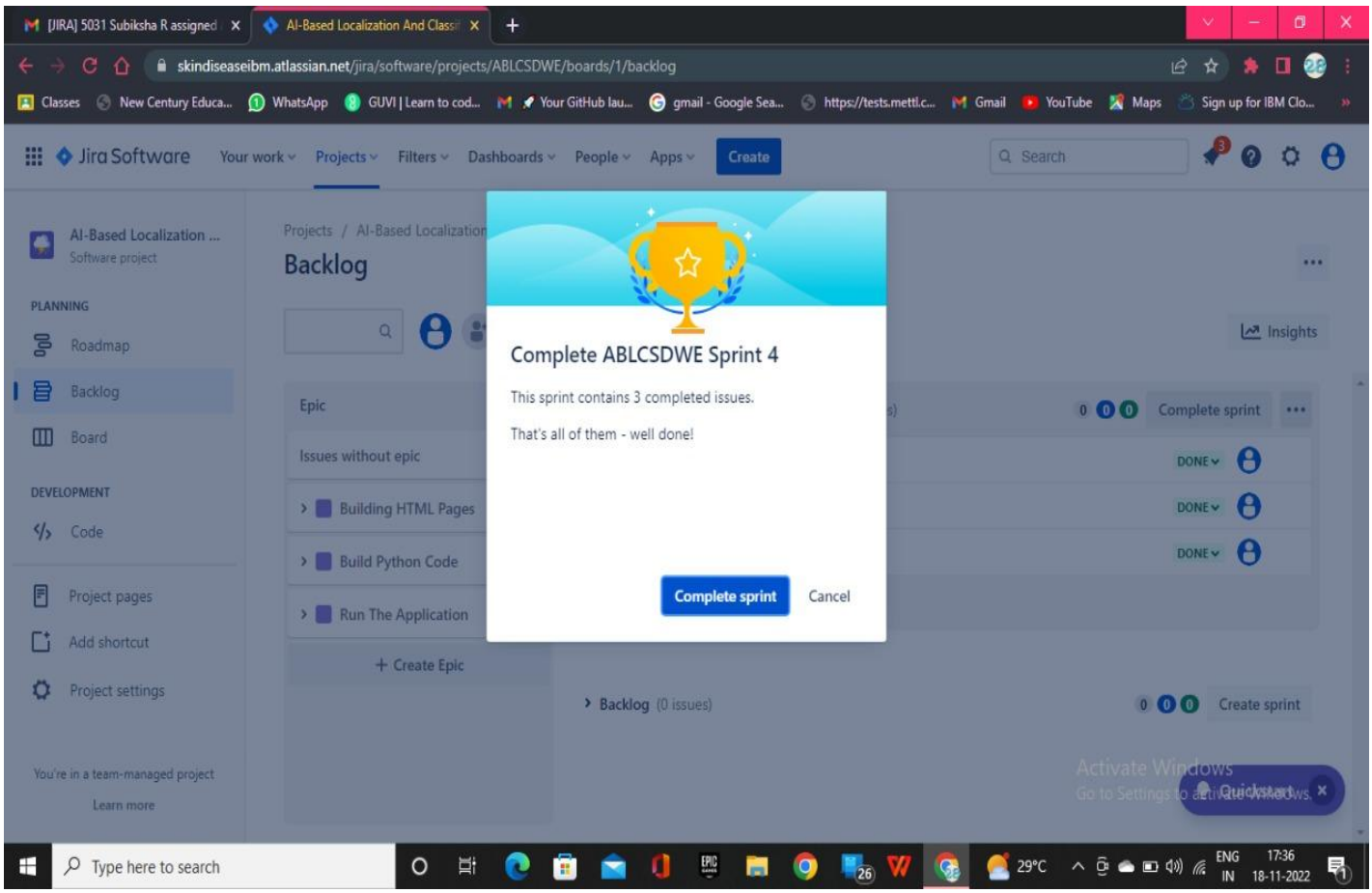

skindiseaseibm.atlassian.net/jira/software/projects/ABLCSDWE/boards/1/backlog
Your work
Projects
Filters
Dashboards
People
Apps
Create
Search
AI-Based Localization ...
Software project
PLANNING
Roadmap
Backlog
Board
DEVELOPMENT
Code
Project pages
Add shortcut
You're in a team-managed project
Learn more
Backlog
Insights
Epic
Issues without epic
Create Epic
Complete sprint
Complete ABLCSDWE Sprint 2
This sprint contains 6 completed issues.
That's all of them - well done!
Complete sprint
Cancel
DONE
Quickstart
Sprint -1.pdf
Failed - System busy
Show all
Type here to search
28°C Cloudy
ENG
12:54 PM
11/14/2022

Sprint- 3


skindiseaseibm.atlassian.net/jira/software/projects/ABLCSDWE/boards/1/roadmap
Jira Software
Your work
Projects
Filters
Dashboards
People
More
Create
Search
AI-Based Localization ...
Software project
Roadmap
Backlog
Board
DEVELOPMENT
Code
Project pages
Add shortcut
Project settings
You're in a team-managed project
Learn more
Projects / AI-Based Localization And Classification Of Skin Disease With Erythema
Roadmap
Give feedback
Share
Export
Status category
Epic
OCT
NOV
DEC
Sprints
ABLCSDWE-3 Register and login ... DONE
ABLCSDWE-4 Create Service Inst... DONE
ABLCSDWE-5 Creating Service Cr... DONE
ABLCSDWE-6 Launch Cloud DB DONE
ABLCSDWE-7 Database DONE
Today
Weeks
Months
Quarters
29°C
Cloudy
Search
ENG
IN
12:48
15-11-2022


skindiseaseibm.atlassian.net/jira/software/projects/ABLCSDWE/boards/1
Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
Search
AI-Based Localization ...
Software project
PLANNING
Roadmap
Backlog
Board
DEVELOPMENT
Code
Project pages
Add shortcut
You're in a team-managed project
Learn more
Projects / AI-Based Loca
All sprints
Complete sprint
GROUP BY
None
Insights
TO DO
DONE 5 ISSUES
Complete sprints
Select a sprint to complete
ABLCSDWE Sprint 3
This sprint contains 5 completed issues.
That's all of them - well done!
Complete sprint
Cancel
Create Service Credentials
ABLCSDWE-13
Launch Cloud DB
ABLCSDWE-14
Create Database
ABLCSDWE-15
ABLCSDWE-53
Create A Project In VOTT -Part 2
ABLCSDWE-54
30°C
Cloudy
Search
ENG
IN
15:39
14-11-2022

Sprint-4


skindiseaseibm.atlassian.net/jira/software/projects/ABLCSDWE/boards/1/backlog
Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
Search
AI-Based Localization ...
Software project
PLANNING
Roadmap
Backlog
Board
DEVELOPMENT
Code
Project pages
Add shortcut
Project settings
You're in a team-managed project
Learn more
Projects / AI-Based Localization
Backlog
Insights
Epic
Issues without epic
Building HTML Pages
Build Python Code
Run The Application
+ Create Epic
Complete sprint
DONE
Backlog (0 issues)
Create sprint
Complete ABLCSDWE Sprint 4
This sprint contains 3 completed issues.
That's all of them - well done!
Complete sprint
Cancel
Activate Windows
Quickstart
Type here to search
29°C
ENG
IN
17:36
18-11-2022

7.coding & solution

7.1 Feature 1

```
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\Super User\Desktop\yolo_structure\2_Training\src'

print(src_path)

utils_path = r'C:\Users\Super User\Desktop\yolo_structure\Utils'

print(utils_path)

sys.path.append(src_path

)sys.path.append(utils_path)
```

7.2 Feature 2

```
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



import cloudant
import cloudant.client
from cloudant.client import Cloudant

# Authenticate using an IAM API key
client = Cloudant.iam('dc238670-af60-480f-8a22-b3e951c12602-bluemix','rJj6IPhnXzHSnGFWyTEWEYk-pdQiTmhfjlP3tbMZmVcV', 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('/')
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')



@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=False)
```

7.3 Database Schema

- Registration: When a new user registers, the backend connects to the IBM Cloudant and stores the user's credentials in the database.

- Login: To check if a user is already registered, the backend connects to Cloudant when they attempt to log in. They are an invalid user if they are not already registered.

- IBM cloudant: Stores the data which is registered.

- app.py: Connects both Frontend and the cloudant for the verification of user credentials

8 TESTING

8.1 Test Cases

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

8.2 User Acceptance Testing

Section	Total Cases	Not Tested	Fail	Pass
Registration	8	0	0	8
Login	36	0	0	36
Security	2	0	0	2
Disease Detection	12	0	0	12
Exception Reporting	8	0	0	8
Final Report Output	4	0	0	4
Version Control	2	0	0	2

9 RESULTS

9.1 Performance Metrics

S.No.	Parameter	Values
1.	Model Summary	To evaluate object detection models like R-CNN and YOLO, the mean average precision (mAP) is used. Them AP compares the ground-truth bounding box to thedetected box and returns a score.
2.	Accuracy	Training Accuracy – 88% Validation Accuracy – 93%
3.	Confidence Score (Only Yolo Projects)	Class Detected – 95% Confidence Score – 90%

ADVANTAGES & DISADVANTAGES

Advantages:

- Report any skin lesions or sores to your coaching staff immediately (and parent or guardian if you are under 18 years of age).
- Have rashes and sores evaluated by a medical provider before resuming practice or competition.
- Do not share cell phones, beverage containers (such as water bottles or sports drinks), cigarettes, or anything else that touches the lips, enters the mouth, or has contact with an affected skin area.
- Avoid touching your eyes, nose, or mouth with your hands to help prevent the spread of infections.
- Do not share cell phones, beverage containers (such as water bottles or sports drinks), cigarettes, or anything else that touches the lips, enters the mouth, or has contact with an affected skin area

Disadvantages:

- Skin diseases such as acne, psoriasis, and eczema are associated with a significant impairment in the quality of the patient's daily life
- Several instruments assess quality-of-life (QoL) in adults and children with skin disease and help us understand its impact.
- Three groups of investigators have recently examined the psychosocial effects of skin disorders.
- Evers and colleagues analyzed the effects of psychological stressors on skin disease in patients with psoriasis.
- This report follows their earlier finding of clinical exacerbation of psoriasis in the month following stressful life events.
- The present longitudinal, prospective study assessed how stressors affect serum levels of cortisol, a key component of the hypothalamic-pituitary-adrenal (HPA) axis, in psoriasis patients.

CONCLUSION

In conclusion, the fact that psychological interventions can have important effects on the severity of chronic dermatological disorders offers an exciting prospect for the management of skin patients The system proposed is a Skin Disease Detection System. This system uses images of skin captured with a camera to detect if it is healthy or not; if not, then classified as Melanoma, Eczema or LeprosyThe proposed system uses image processing and machine learning techniques. The process begins with pre-processing an input image using contrast enhancement and grayscale conversion. Global Value Thresholding technique is used to segment the pre-processed image through which the actual affected region is obtained.

FUTURE SCOPE

The researchers recommend that future research be performed to examine the feature extraction actions based on biomarkers, even though there is ample data, depending on the specific findings This system can be used by dermatologists to give a better diagnosis and treatment to the patients. The system can be used to diagnose skin diseases at a lower cost. In future, this system can be improved to detect and classify more diseases as well as their severityThe proposed model is computationally efficient as it is designed to work on top of lightweight capability devices. Furthermore, with accurate segmentation, we gain knowledge of the location of the disease, which is useful in the preprocessing of data used in classification, as it allows the CNN model to focus on the area of interest.

APPENDIX

Source Code

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


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\Super User\Desktop\yolo_structure\2_Training\src'

print(src_path)

utils_path = r'C:\Users\Super User\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



import cloudant
import cloudant.client
from cloudant.client import Cloudant
```

```
# Authenticate using an IAM API key
client = Cloudant.iam('dc238670-af60-480f-8a22-b3e951c12602-bluemix','rJj6IPhnXzHSnGFWyTEWEYk-pdQiTmhfjlP3tbMZmVcV', 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('/')
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')



@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=False)
```

GitHub & Project Demo Link

Github: https://github.com/IBM-EPBL/IBM-Project-50907-1660929397

Project Demo Link:

https://drive.google.com/file/d/1r0zBot1IoTb8wIr-4iu42oulJx07ZlEa/view?usp=share_link