AI-BASED LOCALIZATION AND CLASSIFICATION OF SKIN DISEASE WITH ERYTHEMA

PROJECT REPORT

PROJECT ID:PNT2022TMID25885

Team Size: 4

Team Leader: REEHANA S

Team member 1: KIRAN KOUSHIKA R

Team member 2: BIRDULA DEEKSHITHA REDDY

Team member 3: ANUJHA S B

PANIMALAR INSTITUTE OF TECHNOLOGY

CHENNAI − 600 123

PROJECT REPORT

1. INTRODUCTION

1.1 Project Overview

1.2 Purpose

2. LITERATURE SURVEY

2.1 Existing problem

2.2 References

2.3 Problem Statement Definition

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

3.2 Ideation & Brainstorming

3.3 Proposed Solution

3.4 Problem Solution fit

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

4.2 Non-Functional requirements

5. PROJECT DESIGN

5.1 Data Flow Diagrams

5.2 Solution & Technical Architecture

5.3 User Stories

6. PROJECT PLANNING & SCHEDULING

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

1.2 PURPOSE

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

2. LITERATURE SURVEY

2.1 EXISTING PROBLEM

The existing applications does not use large datasets and thus will not produce accurate results . And the algorithms used in these are generally designed to select a single likely diagnosis , thus providing suboptimal results for patients .Analyzing big data is a major challenge that these applications face.

2.2 REFERENCES

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

2.Burlina, P.M., Joshi, N.J., Mathew, P.A., Paul, W., Rebman, A.W. and Aucott, J.N., 2020. AI-based detection of erythema migrans and disambiguation against other skin lesions. *Computers in biology and medicine*, *125*, p.103977.

3. Ranjan, R., Partl, R., Erhart, R., Kurup, N. and Schnidar, H., 2021. The mathematics of erythema: Development of machine learning models for artificial intelligence assisted measurement and severity scoring of radiation induced dermatitis. *Computers in Biology and Medicine*, *139*, p.104952.

4Wu, Z.H.E., Zhao, S., Peng, Y., He, X., Zhao, X., Huang, K., Wu, X., Fan, W., Li, F., Chen, M. and Li, J., 2019. Studies on different CNN algorithms for face skin disease classification based on clinical images. *IEEE Access*, *7*, pp.66505-66511.

5. Sreekala, K., Rajkumar, N., Sugumar, R., Sagar, K.V., Shobarani, R., Krishnamoorthy, K.P., Saini, A.K., Palivela, H. and Yeshitla, A., 2022. Skin Diseases Classification Using Hybrid AI Based Localization Approach. *Computational Intelligence and Neuroscience*, *2022*.

6.Tschandl, P., Rinner, C., Apalla, Z., Argenziano, G., Codella, N., Halpern, A., Janda, M., Lallas, A., Longo, C., Malvehy, J. and Paoli, J., 2020. Human–computer collaboration for skin cancer recognition. *Nature Medicine*, *26*(8), pp.1229-1234.

7. Bajwa, M.N., Muta, K., Malik, M.I., Siddiqui, S.A., Braun, S.A., Homey, B., Dengel, A. and Ahmed, S., 2020. Computer-aided diagnosis of skin diseases using deep neural networks. *Applied Sciences*, *10*(7), p.2488.

8.Goyal, M., Knackstedt, T., Yan, S. and Hassanpour, S., 2020. Artificial intelligence-based

image classification methods for diagnosis of skin cancer: Challenges and opportunities. *Computers in Biology and Medicine, 127*, p.104065.

9. Lucieri, A., Dengel, A. and Ahmed, S., 2021. Deep Learning Based Decision Support for Medicine--A Case Study on Skin Cancer Diagnosis. *arXiv preprint arXiv:2103.05112*.

10. Attallah, O. and Sharkas, M., 2021. Intelligent dermatologist tool for classifying multiple skin cancer subtypes by incorporating manifold radiomics features categories. *Contrast media & molecular imaging, 2021*.

2.3 PROBLEM STATEMENT DEFINITION
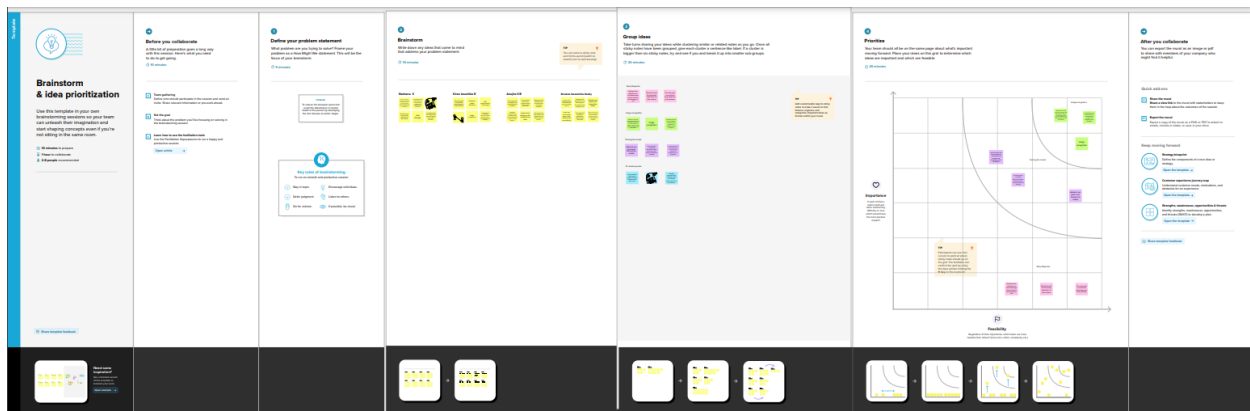

1. To virtually identify the skin disease of the patient at earlier stages.

2. To make the transportation effort of the patient easier.

3. To reduce the time span spent on finding a specialized doctor.

4. To avoid the disturbance in the mental health of the person.

3. IDEATION AND PROPOSED SYSTEM

3.1 EMPATHY MAP CANVAS

3.2 IDEATION AND BRAINSTORMING

3.3 PROPOSED SYSTEM

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	To reduce the timespan spent and avoid the disturbance in mental health of the person by identifying the skin disease at earlier stages.
2.	Idea / Solution description	To check whether the person is having a skin disease or not by diagnosing the different characteristics of affected skin by capturing the images and sent to the trained model.
3.	Novelty / Uniqueness	Its uniqueness is that it specifically exposes the disease at earlier stages, in the process of diagnosis.
4.	Social Impact / Customer Satisfaction	The advantage of knowing the type of disease the person is affected with, lets he or she to consult the respective specialized doctor for successful treatment and faster recovery.
5.	Business Model (Revenue Model)	It saves the patient's life by detecting the disease at earlier stage. Profits made from virtual consulting and by making people access the premium facilities.
6.	Scalability of the Solution	Scalability is high since this SaaS model enables people to access the services and features in detecting the skin disease more easily and also provides cloud storage.

3.4 PROBLEM FIT SOLUTION

| Project Title: | AI-Based Localization And Classification Of Skin Disease With Erythema | Project Design Phase-I - Solution Fit Template | Team ID: PNT2022TMID25885 |

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) — CS
- Children those who are under 10 and also older children and adults

6. CUSTOMER CONSTRAINTS — CC
- If the product isn't right for the particular skin rashes it may create new ones
- Skin products are costly
- Visiting a doctor may not cure skin disease, dermatologists are trained to diagnose skin disease

5. AVAILABLE SOLUTIONS — AS
- Oral medications, creams, laser and surgical treatments are available solutions. Visiting a dermatologist is an alternative to diagnosing the skin rashes using trained images.

Explore AS, differentiate

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

2. JOBS-TO-BE-DONE / PROBLEMS — J&P
- Not investigating infected regions at early stage.
- Melanoma is a serious skin cancer that looks like a mole and needs to be diagnosed at an early stage.
- Skin rashes that appear flat, rough, and scaly need to be diagnosed and are often a precancerous stage.

9. PROBLEM ROOT CAUSE — RC
- Customers have to do it because it makes the skin worse.
- People with diabetes may increase their chances of increasing a skin disease so they need to take early precautions.
- It will lead to skin cancer if not diagnosed at an early stage.

7. BEHAVIOUR — BE
- They looking for a fast recovery after self-diagnosing.
- Most of the customers use skin care products based on the recommendation of a friend or acquaintance to get the job done rather than visiting a dermatologist which makes it easier to adjust and balance life.

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

Identify strong TR & EM

3. TRIGGERS — TR
- When it causes allergies, irritants, or when skin disease develops in their body it leads to fear and makes them stressed.

4. EMOTIONS: BEFORE / AFTER — EM
- Customers feel if it is malignant or benign when they face the problem.
- They feel that should they consult a doctor and also they feel stressed and irritated.

10. YOUR SOLUTION — SL
- To check whether the person is having a skin disease or not by diagnosing the different characteristics of the affected skin by capturing the images and detecting them using AI. And then building an application for customers to identify skin conditions easily.

8. CHANNELS of BEHAVIOUR — CH

8.1 ONLINE
- Customers can consult an online dermatologist and ask their queries. AI will detect the skin images and compare them with the trained model and give results.

8.2 OFFLINE
- Customers need to search for a specialized doctor for a particular skin disease.
- They ask for suggestions from close ones.
- Doctors might be confused with the type of skin disease by their shape and skin color.

Identify strong TR & EM

4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Interface	Login Form
FR-4	Uploading the Image	Upload the captured images from Gallery Give access to capture images through Camera
FR-5	Display the Output	Display the Predicted Result
FR-6	User Logout	Logout indication through Gmail

4.2 NON-FUNCTIONAL REQUIREMENTS

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	People who have less knowledge about technology should also able to use the website
NFR-2	Security	Data privacy and security practices may vary based on users and their age
NFR-3	Reliability	The website and its system should consistently perform the specified functions without failure
NFR-4	Performance	The loading time of the front page of the website must be no more than 2 seconds
NFR-5	Availability	How likely the system is accessible to a user at a given point in time
NFR-6	Scalability	The ability to appropriately handle different workloads

5. PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.
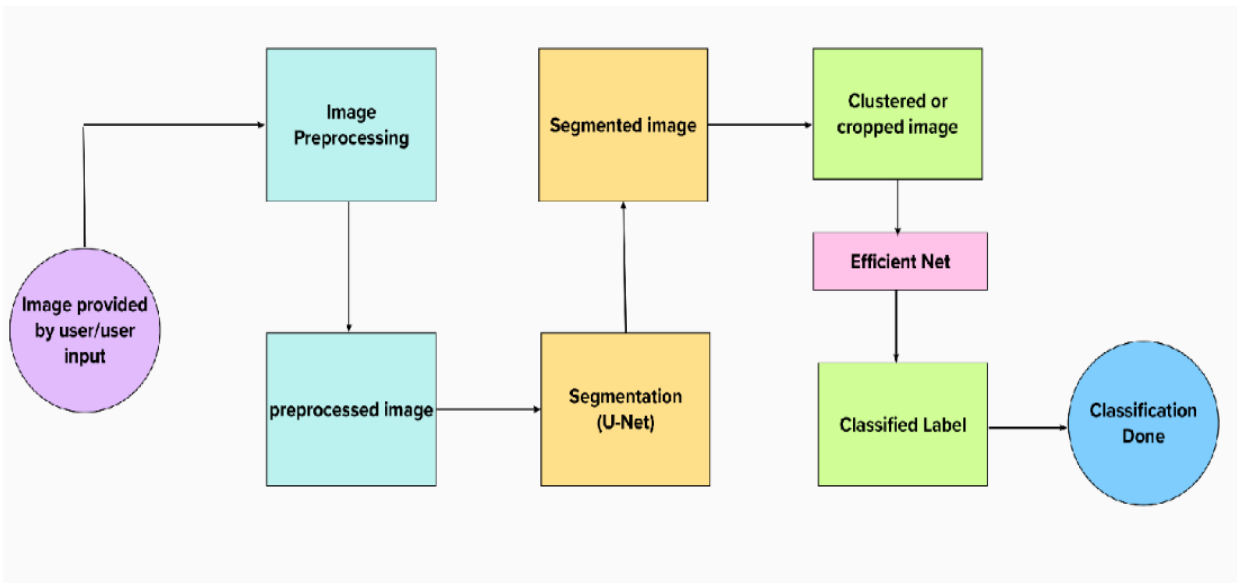
5.2 SOLUTION AND TECHNICAL ARCHITECTURE

Solution Architecture:

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

- Find the best tech solution to solve existing business problems.

- Describe the software's structure, characteristics, behavior, and other aspects to project stakeholders.

- Define features, development phases, and solution requirements.

- Provide specifications according to which the solution is defined, managed, and delivered.
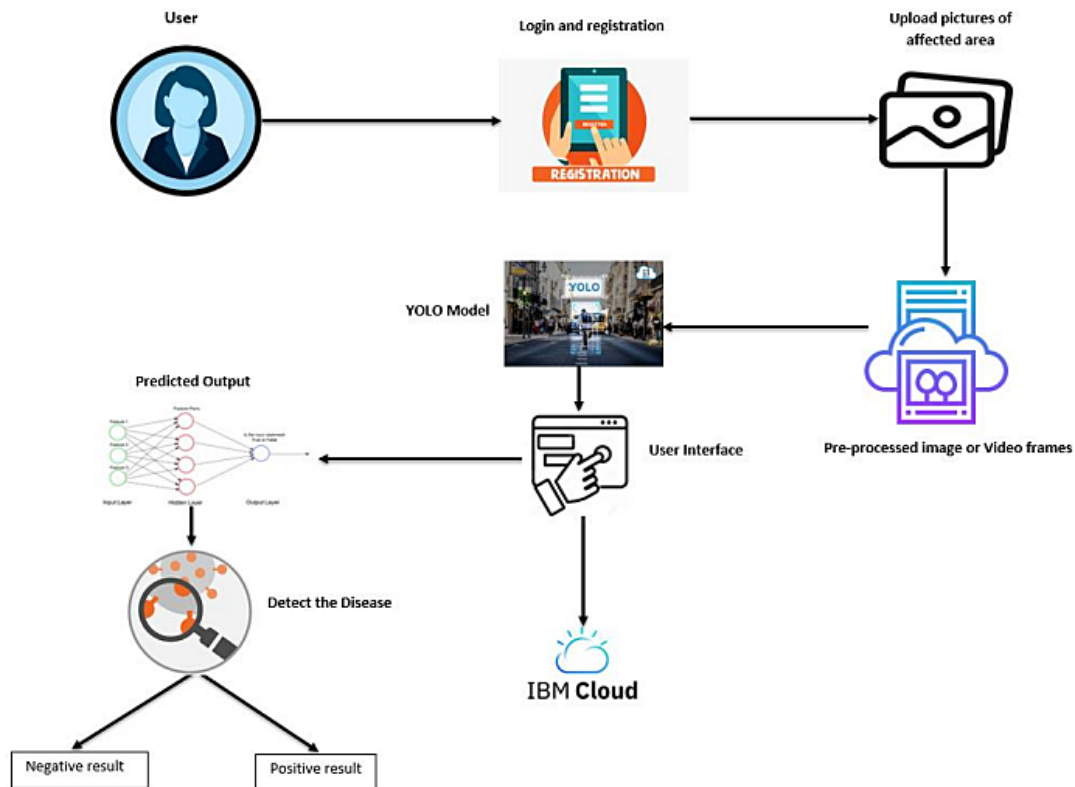
Technical Architecture:

The Deliverable shall include the architectural diagram below and the information as per the table1 & table 2
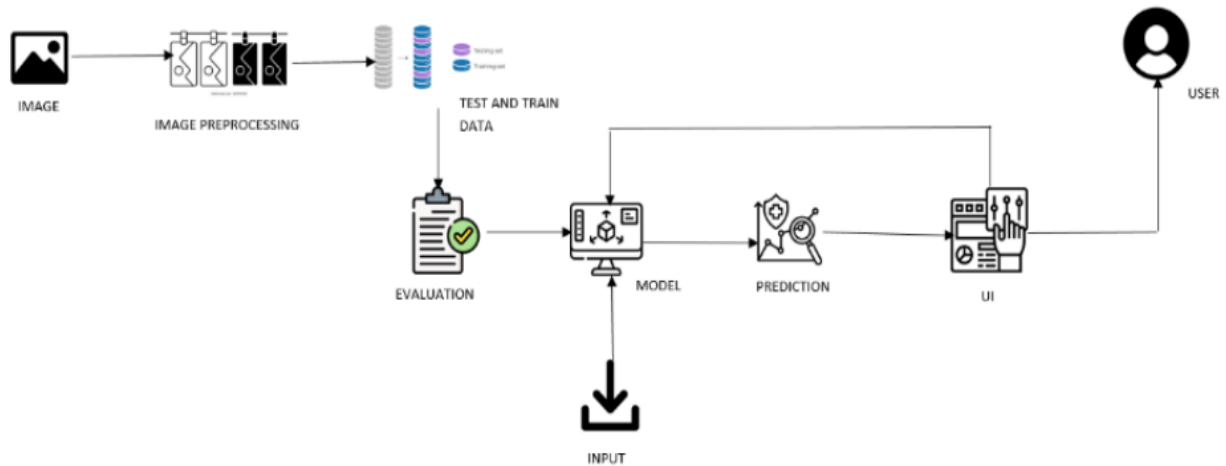
Table 1: Components and Technologies:

S.No	Component	Description	Technology
1.	User Interface	How the user interacts with the application	HTML, CSS, JavaScript
2.	Application Logic-1	HTML page for login, Registration, Prediction, and Logout	Python
3.	Application Logic-2	YOLOv3 detector is a real-time object detection algorithm that specifies the object in an image.	Python
4.	Application Logic-3	Computer vision can gain a high understanding of images.	OpenCV, machine learning software

5.	Database	Using chrome extensions such as batch downloader where you can search and download images from chrome	MySQL, NoSQL, etc.
6.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc.
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem
8.	External API-1	Application registration using Email	HTML
9.	External API-2	Confirmation via Email	Email
10.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud server	IBM Platform

Table 2: Application Characteristics:

S.No	Characteristics	Description	Technologies
1.	Open-Source Frameworks	Annotate images using VOTT	Cloud DB
2.	Security Implementations	List all the security/access controls implemented, use of firewalls, etc.	Encryptions, IAM Controls

3.	Scalable Architecture	This method is ensured accurate information about the patient skin disease.	Artificial intelligence
4.	Availability	Prediction of the disease at early stages helps in the early cure of the disease	Image Processing
5.	Performance	The application can predict accurate results at perfect times.	IBM Cloud

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
		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 dashboard 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)	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-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
Administrator	Manage	USN-5	Do-it-yourself service for delivering Everything.	set of predefined requirements that must be met to mark a user story complete.	High	Sprint-4

6. PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING AND ESTIMATION

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Index	USN-1	As a user, I can able to know the basic information of skin disease	1	Low	
Sprint-1	Registration	USN-2	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	
Sprint-1		USN-3	As a user, I will receive confirmation email once I have registered for the application	1	High	
Sprint-1		USN-4	As a user, I can register for the application through Gmail	2	Medium	
Sprint-1	Login	USN-5	As a user, I can log into the application by entering name, email and password	1	High	
Sprint-2	Prediction	USN-6	As a user, I should able to predict the type of skin Disease.	3	High	
Sprint-3	Demo	USN-7	As a user, I should able use the demo button, on how to use the application	3	Medium	
Sprint-4	Logout	USN-8	As a user, I can logout from the website once completed the process	2	Medium	
Sprint-4	Run	USN-9	As a user, I should test the app and run the Application.	3	High	

6.2 SPRINT DELIVERY SCHEDULE

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	3 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	10 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.3 REPORTS FROM JIRA

SPRINT 1:

Sprints		NOV	DEC	JAN '23
	ABLC... ABLC... ABLC... ABLC... ABLC...			
⌄ ⚡ ABLCSDWE-10 Index				
🟩 ABLCSDWE-1 As a user, I can able to kn... DONE				
⌄ ⚡ ABLCSDWE-11 Registration				
🟩 ABLCSDWE-4 As a user, I can register fo... DONE				
🟩 ABLCSDWE-3 As a user, I will receive co... DONE				
🟩 ABLCSDWE-2 As a user, I can register fo... DONE				
⌄ ⚡ ABLCSDWE-12 Login				
🟩 ABLCSDWE-5 As a user, I c... DONE KIRAN KOU...				

Sprints	OCT			OCT			NOV		
	19 20 21 22 23			24 25 26 27 28 29 30			31 1 2 3 4 5		
				ABLCSDWE Sprint 1			ABLCSDWE Sprint 2		
› ⚡ ABLCSDWE-10 Index									
› ⚡ ABLCSDWE-11 Registration									
› ⚡ ABLCSDWE-12 Login									

SPRINT 2:

Sprints	OCT	NOV	DEC
		ABLC... ABLC... ABLC... ABLC... ABLC...	
› ⚡ ABLCSDWE-13 Prediction		⚠	

SPRINT 3:

> ⚡ ABLCSDWE-19 Demo

SPRINT 4:

		NOV					NOV						NOV						
		3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Sprints		ABLCSDWE S...			ABLCSDWE Sprint 3							ABLCSDWE Sprint 4						AB...
⚡ ABLCSDWE-20 logout	DONE																	
⚡ ABLCSDWE-21 run																		

7. CODING & SOLUTIONING

7.1 FEATURE 1

index.html

```html
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-
   scale=1.0">
7
8      <!--Bootstrap -->
9      <link rel="stylesheet"
   href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.
   css" integrity="sha384-
   Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
   crossorigin="anonymous">
10     <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
   integrity="sha384-
   KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
   crossorigin="anonymous"></script>
11     <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>
12     <script
```

```html
    src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.j
    s" integrity="sha384-
    JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
    crossorigin="anonymous"></script>
13
14
15      <script src="https://kit.fontawesome.com/8b9cdc2059.js"
    crossorigin="anonymous"></script>
16      <link
    href="https://fonts.googleapis.com/css2?family=Akronim&family=Roboto&dis
    play=swap" rel="stylesheet">
17      <link rel="stylesheet" href="../static/style.css">
18      <!-- <script defer src="../static/js/main.js"></script> -->
19      <title>AI-based localization and classification of skin disease with
    erythema</title>
20 </head>
21 <body>
22      <header id="head" class="header">
23      <section  id="navbar">
24          <h1 class="nav-heading">AI-based localization and
    classification of skin disease with erythema</h1>
25          <div class="nav--items">
26              <ul>
27                      <li><a  href="index.html">Home</a></li>
28                      <li><a href="login.html">Login</a></li>
29                      <li><a href="register.html">Register</a></li>
30                  <!-- <li><a href="#about">About</a></li>
31                  <li><a href="#services">Services</a></li> -->
32                  <li><a href="prediction.html">Prediction</a></li>
33              </ul>
34          </div>
35      </section>
36      <section id="slider">
37       <div id="carouselExampleIndicators" class="carousel" data-
    ride="carousel">
38          <ol class="carousel-indicators ">
39              <li data-target="#carouselExampleIndicators" data-slide-
    to="0" class="active "></li>
40              <li data-target="#carouselExampleIndicators" data-slide-
    to="1"></li>
41              <li data-target="#carouselExampleIndicators" data-slide-
    to="2"></li>
42          </ol>
43          <div class="carousel-inner">
```

```html
44
45              <div class="carousel-item active">
46                  <img class="d-block w-100"
    src="https://storage.googleapis.com/gweb-uniblog-publish-
    prod/images/derm_-_hero_image_2_1.max-1000x1000.jpg" alt="First slide"
    style="height:500px">
47              </div>
48
49          <a class="carousel-control-prev"
    href="#carouselExampleIndicators" role="button" data-slide="prev">
50              <span class="carousel-control-prev-icon" aria-
    hidden="true"></span>
51              <span class="sr-only">Previous</span>
52          </a>
53          <a class="carousel-control-next"
    href="#carouselExampleIndicators" role="button" data-slide="next">
54              <span class="carousel-control-next-icon" aria-
    hidden="true"></span>
55              <span class="sr-only">Next</span>
56          </a>
57      </div>
58
59      </section>
60 </header>
61 <section id="about">
62      <div class="top">
63          <h3 class="title text-muted">
64              ABOUT PROJECT
65          </h3>
66          <div class="line"></div>
67      </div>
68 <div class="body">
69 <div class="left">
70      <h2>Problem:</h2>
71      <p>
72          Nowadays people are suffering from skin diseases, More than 125
    million people suffering from Psoriasis also skin cancer rate is rapidly
    increasing over the last few decades especially Melanoma is most
    diversifying skin cancer. If skin diseases are not treated at an earlier
    stage, then it may lead to complications in the body including spreading
    of the infection from one individual to the other. The skin diseases can
    be prevented by investigating the infected region at an early stage. The
    characteristic of the skin images is diversified so that it is a
    challenging job to devise an efficient and robust algorithm for
```

```
       automatic detection of skin disease and its severity. Skin tone and skin
       colour play an important role in skin disease detection. Colour and
       coarseness of skin are visually different. Automatic processing of such
       images for skin analysis requires quantitative discriminator to
       differentiate the diseases.
73
74          </p>
75  </div>
76  <div class="right">
77      <h2>Solution:</h2>
78      <p>
79          To overcome this problem we are building a model which is used
       for the prevention and early detection of skin cancer, psoriasis.
       Basically, skin disease diagnosis depends on the different
       characteristics like colour, shape, texture etc. Here the person can
       capture the images of skin and then the image will be sent the trained
       model. The model analyses the image and detect whether the person is
       having skin disease or not
80      </p>
81  </div>
82  </div>
83  </section>
84  <section id="services">
85  <h3 class="title text-muted">WE CLASSIFY</h3>
86  <div class="line"></div>
87  <div class="testimonials">
88      <div class="card" style="width: 25rem;">
89          <img
    src="https://upload.wikimedia.org/wikipedia/commons/6/69/Erythema_multif
    orme_minor_of_the_hand.jpg" class="card-img-top" alt="Erythema
    multiforme (EM)" style="height: 300px">
90          <div class="card-body text-muted">
91              <h5 class="card-title text-muted">Erythema multiforme
    (EM)</h5>
92          </div>
93        </div>
94      <div class="card" style="width: 25rem;">
95          <img
    src="https://i.pinimg.com/736x/7e/ec/2e/7eec2ee1c2e592aa94dac11036fad68
    d--lyme-disease-nursing.jpg" class="card-img-top" alt="Erythema
    chronicum migrans" style="height:300px">
96          <div class="card-body text-muted">
97              <h5 class="card-title text-muted">Erythema chronicum
    migrans</h5>
```

```
98
99              </div>
100         </div>
101      <div class="card" style="width: 25rem;">
102          <img src="https://phil.cdc.gov//PHIL_Images/9875/9875_lores.jpg"
    class="card-img-top" alt="Erythema migrans" style="height:300px">
103          <div class="card-body">
104              <h5 class="card-title text-muted">Erythema migrans</h5>
105
106          </div>
107          <div class="card" style="width: 25rem;">
108              <img
    src="https://www.researchgate.net/publication/304555872/figure/fig1/AS:3
    79660283203584@1467529473118/Erythema-marginatum-on-the-arm-of-a-
    patient-with-hereditary-angioedema-Note-that-lesions.png" class="card-
    img-top" alt="Erythema marginatum" style="height:300px">
109              <div class="card-body">
110                      <h5 class="card-title text-muted">Erythema
    marginatum</h5>
111
112              </div>
113          <div class="card" style="width: 25rem;">
114                  <img
    src="https://images.medicinenet.com/images/image_collection/skin/erythem
    a-infectiosum.jpg" class="card-img-top" alt="Erythema infectiosum"
    style="height:300px">
115                  <div class="card-body">
116                          <h5 class="card-title text-muted">Erythema
    infectiosum</h5>
117
118                  </div>
119          <div class="card" style="width: 25rem;">
120                  <img
    src="https://media.healthdirect.org.au/images/inline/original/erythma_no
    dosum_a4gprc-ccee8a.jpg" class="card-img-top" alt="Erythema nodosum"
    style="height:300px">
121                  <div class="card-body">
122                          <h5 class="card-title text-muted">Erythema
    nodosum</h5>
123
124              </div>
125
126         </div>
127
```

```
128</div>
129</section>
130
131
132<section id="footer">
133
134    <div class="social">
135      <a href="#" target="_blank"><i class="fab fa-2x fa-twitter-
   square"></i></a>
136        <a href="#" target="_blank">
137          <i class="fab fa-2x  fa-linkedin"></i></a>
138          <a href="#">
139            <i class="#"></i>
140          </a>
141    </div>
142</section>
143</body>
144</html>
```

➤ Index page of our UI consists of basic idea of our application.
➤ It consists of options to navigate from the home page to other pages.

register.html

```
1  <!DOCTYPE html>
2  <html >
3
4  <head>
5    <meta charset="UTF-8">
6    <meta name="viewport" content="width=device-width, initial-
   scale=1">
7    <title>AI-based localization and classification of skin disease
   with erythema</title>
8    <link href='https://fonts.googleapis.com/css?family=Pacifico'
   rel='stylesheet' type='text/css'>
9  <link href='https://fonts.googleapis.com/css?family=Arimo'
   rel='stylesheet' type='text/css'>
10 <link href='https://fonts.googleapis.com/css?family=Hind:300'
   rel='stylesheet' type='text/css'>
11 <link
   href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed
   :300' rel='stylesheet' type='text/css'>
```

```
12 <link rel="stylesheet" href="{{ url_for('static',
   filename='css/style.css') }}">
13
14 <link href='https://fonts.googleapis.com/css?family=Merriweather'
   rel='stylesheet'>
15 <link href='https://fonts.googleapis.com/css?family=Josefin Sans'
   rel='stylesheet'>
16 <link href='https://fonts.googleapis.com/css?family=Montserrat'
   rel='stylesheet'>
17
18 <style>
19 .header {
20             top:0;
21             margin:0px;
22             left: 0px;
23             right: 0px;
24             position: fixed;
25             background-color: #28272c;
26             color: white;
27             box-shadow: 0px 8px 4px grey;
28             overflow: hidden;
29             padding-left:20px;
30             font-family: 'Josefin Sans';
31             font-size: 2vw;
32             width: 100%;
33             height:8%;
34             text-align: center;
35         }
36         .topnav {
37   overflow: hidden;
38   background-color: #333;
39 }
40
41 .topnav-right a {
42   float: left;
43   color: #f2f2f2;
44   text-align: center;
45   padding: 14px 16px;
46   text-decoration: none;
```

```css
47    font-size: 18px;
48 }
49
50 .topnav-right a:hover {
51    background-color: #ddd;
52    color: black;
53 }
54
55 .topnav-right a.active {
56    background-color: #565961;
57    color: white;
58 }
59
60 .topnav-right {
61    float: right;
62    padding-right:100px;
63 }
64
65 .login{
66 margin-top:-70px;
67 }
68 body {
69
70    background-color:#ffffff;
71    background-repeat: no-repeat;
72    background-size:cover;
73    background-position: 0px 0px;
74    }
75 .login{
76   margin-top:100px;
77 }
78 form {border: 3px solid #f1f1f1; margin-left:400px;margin-
   right:400px;}
79
80 input[type=text],
   input[type=email],input[type=number],input[type=password] {
81    width: 100%;
82    padding: 12px 20px;
83    display: inline-block;
84    margin-bottom:18px;
```

```css
85    border: 1px solid #ccc;
86    box-sizing: border-box;
87 }
88
89 button {
90    background-color: #28272c;
91    color: white;
92    padding: 14px 20px;
93    margin-bottom:8px;
94    border: none;
95    cursor: pointer;
96    width: 100%;
97 }
98
99 button:hover {
100    opacity: 0.8;
101 }
102
103 .cancelbtn {
104    width: auto;
105    padding: 10px 18px;
106    background-color: #f44336;
107 }
108
109 .imgcontainer {
110    text-align: center;
111    margin: 24px 0 12px 0;
112 }
113
114 img.avatar {
115    width: 30%;
116    border-radius: 50%;
117 }
118
119 .container {
120    padding: 16px;
121 }
122
123 span.psw {
124    float: right;
```

```
125    padding-top: 16px;
126
127 }
128
129 /* Change styles for span and cancel button on extra small
    screens */
130 @media screen and (max-width: 300px) {
131    span.psw {
132       display: block;
133        float: none;
134    }
135    .cancelbtn {
136       width: 100%;
137    }
138 }
139
140 </style>
141 </head>
142
143 <body style="font-family:Montserrat;">
144
145 <div class="header">
146    <div style="width:50%;float:left;font-size:1.5vw;text-
    align:left;color:white; padding-top:1%">AI-based localization and
    classification of skin disease with erythema</div>
147    <div class="topnav-right" style="padding-top:0.5%;">
148
149       <a  href="index.html">Home</a>
150       <a href="login.html">Login</a>
151       <a class="active"  href="register.html">Register</a>
152
153    </div>
154 </div>
155 <div id="login" class="login">
156
157
158       <form action="https://formspree.io/f/xdojbzrq"
    method="post">
159
160
```

```
161            <div class="container">
162                <input type="text" placeholder="Enter Name"
    name="name" required><br>
163                <input type="email" placeholder="Enter Email ID"
    name="_id" required><br>
164                <input type="password" placeholder="Enter
    Password" name="psw" required>
165
166                <button type="submit">Register</button><br>
167            </div>
168            <div class="container" style="background-
    color:#f1f1f1">
169        <div class="psw">Already have an account?   <a
    href="login.html">Login</a></div        >
170    </div>
171      </form>
172
173 </div>
174
175
176 </body>
177 </html>
```

➤ This page helps the user to register themselves into our application.
➤ Their basic information such as, mail id and name are saved into our database,
  so that the user can login in directly from the next time.

login.html

```
1  <!DOCTYPE html>
2  <html >
3
4  <head>
5    <meta charset="UTF-8">
6    <meta name="viewport" content="width=device-width, initial-
    scale=1">
7    <title> AI-based localization and classification of skin
    disease with erythema </title>
8    <link href='https://fonts.googleapis.com/css?family=Pacifico'
    rel='stylesheet' type='text/css'>
```

```html
9  <link href='https://fonts.googleapis.com/css?family=Arimo'
   rel='stylesheet' type='text/css'>
10 <link href='https://fonts.googleapis.com/css?family=Hind:300'
   rel='stylesheet' type='text/css'>
11 <link
   href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed
   :300' rel='stylesheet' type='text/css'>
12 <!link rel="stylesheet" href="{{ url_for('static',
   filename='css/style.css') }}">
13 <link href='https://fonts.googleapis.com/css?family=Merriweather'
   rel='stylesheet'>
14 <link href='https://fonts.googleapis.com/css?family=Josefin Sans'
   rel='stylesheet'>
15 <link href='https://fonts.googleapis.com/css?family=Montserrat'
   rel='stylesheet'>
16
17
18 <style>
19 .header {
20             top:0;
21             margin:0px;
22             left: 0px;
23             right: 0px;
24             position: fixed;
25             background-color: #28272c;
26             color: white;
27             box-shadow: 0px 8px 4px grey;
28             overflow: hidden;
29             padding-left:20px;
30             font-family: 'Josefin Sans';
31             font-size: 2vw;
32             width: 100%;
33             height:8%;
34             text-align: center;
35     }
36 .topnav {
37   overflow: hidden;
38   background-color: #333;
39 }
```

```css
40
41 .topnav-right a {
42    float: left;
43    color: #f2f2f2;
44    text-align: center;
45    padding: 14px 16px;
46    text-decoration: none;
47    font-size: 18px;
48 }
49
50 .topnav-right a:hover {
51    background-color: #ddd;
52    color: black;
53 }
54
55 .topnav-right a.active {
56    background-color: #565961;
57    color: white;
58 }
59
60 .topnav-right {
61    float: right;
62    padding-right:100px;
63 }
64
65 .login{
66 margin-top:-70px;
67 }
68 body {
69
70    background-color:#ffffff;
71    background-repeat: no-repeat;
72    background-size:cover;
73    background-position: 0px 0px;
74    }
75 .login{
76   margin-top:100px;
77 }
78 form {border: 3px solid #f1f1f1; margin-left:400px;margin-
   right:400px;}
```

```css
79
80 input[type=text],
   input[type=email],input[type=number],input[type=password] {
81   width: 100%;
82   padding: 12px 20px;
83   display: inline-block;
84   margin-bottom:18px;
85   border: 1px solid #ccc;
86   box-sizing: border-box;
87 }
88
89 button {
90   background-color: #28272c;
91   color: white;
92   padding: 14px 20px;
93   margin-bottom:8px;
94   border: none;
95   cursor: pointer;
96   width: 100%;
97   font-weight:bold;
98 }
99
100 button:hover {
101   opacity: 0.8;
102 }
103
104 .cancelbtn {
105   width: auto;
106   padding: 10px 18px;
107   background-color: #f44336;
108 }
109
110 .imgcontainer {
111   text-align: center;
112   margin: 24px 0 12px 0;
113 }
114
115 img.avatar {
116   width: 30%;
117   border-radius: 50%;
```

```
118 }
119
120 .container {
121    padding: 16px;
122 }
123
124 span.psw {
125    float: right;
126    padding-top: 16px;
127
128 }
129
130 /* Change styles for span and cancel button on extra small
    screens */
131 @media screen and (max-width: 300px) {
132    span.psw {
133       display: block;
134       float: none;
135    }
136    .cancelbtn {
137       width: 100%;
138    }
139 }
140
141 </style>
142 </head>
143
144 <body style="font-family:Montserrat;">
145
146 <div class="header">
147   <div style="width:50%;float:left;font-size:1.5vw;text-
    align:left;color:white; padding-top:1%">AI-based localization and
    classification of skin disease with erythema</div>
148    <div class="topnav-right" style="padding-top:0.5%;">
149
150      <a  href="index.html">Home</a>
151      <a class="active" href="login.html">Login</a>
152      <a  href="register.html">Register</a>
153
154    </div>
```

```
155 </div>
156 <div id="login" class="login">
157
158
159     <form action="prediction.html" method="post">
160
161
162         <div class="container">
163             <input type="email" placeholder="Enter registered
    email ID" name="_id" required><br>
164
165             <input type="password" placeholder="Enter
    Password" name="psw" required>
166
167             <button type="submit">Login</button><br>
168
169         </div>
170     </form>
171
172 </div>
173
174
175 </body>
176 </html>
```

➤ After registering the information, the user can login our application directly.
➤ After logging in, the prediction page is visible.

prediction.html

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-
   scale=1.0">
7      <!--Bootstrap -->
8      <link rel="stylesheet"
   href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstra
```

```html
p.min.css" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
9    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
integrity="sha384-
KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
crossorigin="anonymous"></script>
10   <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/p
opper.min.js" integrity="sha384-
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
crossorigin="anonymous"></script>
11   <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.
min.js" integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
crossorigin="anonymous"></script>
12
13
14   <script src="https://kit.fontawesome.com/8b9cdc2059.js"
crossorigin="anonymous"></script>
15   <link
href="https://fonts.googleapis.com/css2?family=Akronim&family=Robo
to&display=swap" rel="stylesheet">
16   <link rel="stylesheet" href="../static/style.css">
17
18   <script defer src="../static/js/JScript.js"></script>
19   <title>Prediction</title>
20 </head>
21 <body>
22   <header id="head" class="header">
23       <section  id="navbar">
24               <h1 class="nav-heading"></i>Skin Disease
Detection</h1>
25           <div class="nav--items">
26               <ul>
27                   <li><a  href="index.html">Home</a></li>
28                   <li><a
href="file:///C:/Users/kiran/OneDrive/Desktop/SI-GuidedProject-
89669-1658213465-
main/project/templates/logout.html">Logout</a></li>
```

```html
29                 <!-- <li><a href="#about">About</a></li>
30                 <li><a href="#services">Services</a></li> -->
31
32                  </ul>
33              </div>
34          </section>
35      </header>
36      <!-- dataset/Training/metal/metal326.jpg -->
37          </br>
38      <section id="prediction">
39      <h2 class="title text-muted">AI-based localization and
    classification of skin disease with erythema</h1>
40      <div class="line" style="width: 1000px;"></div>
41          </section>
42          </br>
43    <section id="about">
44
45 <div class="body">
46 <div class="left">
47     <p>
48         Nowadays people are suffering from skin diseases, More than
   125 million people suffering from Psoriasis also skin cancer rate
   is rapidly increasing over the last few decades especially
   Melanoma is most diversifying skin cancer. If skin diseases are
   not treated at an earlier stage, then it may lead to complications
   in the body including spreading of the infection from one
   individual to the other. The skin diseases can be prevented by
   investigating the infected region at an early stage. The
   characteristic of the skin images is diversified so that it is a
   challenging job to devise an efficient and robust algorithm for
   automatic detection of skin disease and its severity. Skin tone
   and skin colour play an important role in skin disease detection.
   Colour and coarseness of skin are visually different. Automatic
   processing of such images for skin analysis requires quantitative
   discriminator to differentiate the diseases.
49
50          </p>
51 </div>
52 <div class="left">
53
```

```html
54      <div class="prediction-input">
55          <img class="d-block" src="../static/img/6623.jpg"
    alt="Second slide" style="width:80%!important; align:center;
    padding-left:100px">
56          </br>
57                  <form id="form" action="/result" method="post"
    enctype="multipart/form-data">
58
59                      <input type="submit" class="submitbtn"
    value="Click here for Prediction">
60                  </form>
61              </div>
62              <h5 style="text-color:Red">
63              <b style="text-color:Red" /b>
64          </h5>
65 </div>
66 </div>
67 </section>
68      <section id="footer">
69
70
71      </section>
72 </body>
73
74 </html>
```

➤ Image can be uploaded in the prediction page and the result is displayes here
   aswell.

final.css

```css
1  .img-preview {
2      width: 256px;
3      height: 256px;
4      position: relative;
5      border: 5px solid #F8F8F8;
6      box-shadow: 0px 2px 4px 0px rgba(0, 0, 0, 0.1);
7      margin-top: 1em;
8      margin-bottom: 1em;
```

```css
 9  }
10
11 .img-preview>div {
12     width: 100%;
13     height: 100%;
14     background-size: 256px 256px;
15     background-repeat: no-repeat;
16     background-position: center;
17 }
18
19 input[type="file"] {
20     display: none;
21 }
22
23 .upload-label{
24     display: inline-block;
25     padding: 12px 30px;
26     background: #28272c;
27     color: #fff;
28     font-size: 1em;
29     transition: all .4s;
30     cursor: pointer;
31 }
32
33 .upload-label:hover{
34     background: #C2C5A8;
35     color: #39D2B4;
36 }
37
38 .loader {
39     border: 8px solid #f3f3f3; /* Light grey */
40     border-top: 8px solid #28272c; /* Blue */
41     border-radius: 50%;
42     width: 50px;
43     height: 50px;
44     animation: spin 1s linear infinite;
45 }
46
47 @keyframes spin {
48     0% { transform: rotate(0deg); }
```

```
49    100% { transform: rotate(360deg); }
50 }
```

➤ Used to style the website.

style.css

```css
1  :root{
2      --main-bg-color: #fff;
3      --text-color:#ced4da;
4      --bs-font-sans-serif: Poppins, system-ui, -apple-system,
   "Segoe UI", Roboto, "Helvetica Neue", Arial, "Noto Sans", sans-
   serif, "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol",
   "Noto Color Emoji";
5      --navbar-bg:#333;
6      --hover-color:#228B22;
7      --yellow:#FFD700;
8      --box-shadow:rgba(100, 100, 111, 0.2) 0px 7px 29px 0px
9
10 }
11
12 /* reset */
13 *{
14     margin: 0;
15     padding: 0;
16     box-sizing: border-box;
17 }
18 body{
19     background: var(--main-bg-color);
20     font-family: var(--bs-font-sans-serif);
21     color: #333;
22     line-height: 1.6;
23 }
24 ul{
25     list-style:none;
26 }
27 a{
28     text-decoration: none;
29     color: var(--text-color);
```

```css
30 }
31
32 h1,h2{
33     font-weight: 360;
34     line-height: 1.2;
35 }
36 p{
37 margin: 10px 0px;
38
39 }
40
41 .m2{
42     margin-right: 10px;
43 }
44
45 /* utility */
46 .title{
47     margin-top: 10px;
48     text-align: center;
49 }
50 html {
51     scroll-behavior: smooth;
52   }
53
54 /* Header */
55 #head #navbar{
56     height: 100px;
57      width: auto;
58     background-color: var(--navbar-bg);
59     color: #fff;
60     padding: 10px;
61 }
62 #navbar{
63     display: flex;
64     justify-content: space-between;
65     align-items: center;
66}
67#navbar .nav--items ul{
```

```css
68     display: flex;
69     align-items: center;
70 }
71
72 #navbar .nav--items ul li a{
73 margin: 10px;
74 text-decoration: none;
75 }
76 #navbar .nav--items ul li a:hover{
77     color:var(--hover-color) ;
78 }
79
80 /* header carousel */
81 #head #slider .carousel-item img{
82     display: block;
83     width:100%;
84     height: 50vh;
85 }
86
87 .font{
88     font-size: 50px;
89     font-weight: bold;
90     color: #fff;
91 }
92
93 /* About */
94 #about .top{
95     margin-top: 20px;
96 }
97
98  .line{
99     background-color: var(--yellow);
100     width: 200px;
101     height: 2px;
102     margin: auto;
103     margin-top: 10px;
104 }
105 #about .body{
106     margin-top: 20px;
```

```css
107     display: grid;
108     grid-template-columns: 1fr 1fr;
109     text-align: center;
110 }
111
112 #about .body .right,#about .body .left
113 {
114     box-shadow: rgba(0, 0, 0, 0.15) 0px 3px 3px 0px;
115 margin: 0.5rem;
116 }
117
118 #about .body .right p{
119     justify-self: center;
120     margin-top: 50px;
121 }
122 /* Services */
123 #services .testimonials{
124     display: grid;
125     grid-template-columns: 1fr ;
126     grid-column-gap: 10px;
127     grid-row-gap: 20px;
128     margin: 40px;
129   justify-items: center;
130
131 }
132 #services .testimonials .card{
133  box-shadow: rgba(0, 0, 0, 0.35) 0px 5px 15px;
134  text-align: center;
135 }
136
137 #services .testimonials .card h5{
138 text-transform: uppercase;
139 }
140
141 /* Contcat form */
142 #contact .contact-container{
143     display: grid;
144
145     grid-template-columns: repeat(3,1fr);
```

```css
146        justify-items: center;
147        margin: 3rem;
148 }
149 #contact .contact-container .conatct-left .items h3{
150        display: inline;
151        margin-left: 10px;
152 }
153
154 #contact .contact-container .conatct-left .items{
155        margin: 10px;
156        margin-bottom: 30px;
157
158 }
159
160 #contact .contact-container .contact-right form input,
161 #contact .contact-container .contact-right form button
162  {
163        display: block;
164        margin: 20px
165        }
166
167 /* footer */
168 #footer {
169   width: auto;
170   height: 80px;
171     background-color: var(--navbar-bg);
172     color: #fff;
173     display: flex;
174     align-items: center;
175     justify-content: space-around;
176 }
177 #footer .social a{
178   margin-left: 20px;
179   text-decoration: none;
180 }
181 #footer .social a:hover{
182    color: var(--hover-color);
183 }
184 /* prediction.html */
185
186 #prediction .prediction-input{
187   display: flex;
```

```css
188    align-items: center;
189    justify-content: center;
190    margin-top: 1.5rem;
191 }
192 #prediction .prediction-input form{
193    margin-left: 1.2rem;
194 }
195 #prediction .circle {
196    width: 150px;
197    height: 150px;
198    border-radius: 50%;
199    margin-bottom: 5px;
200    box-shadow:var(--box-shadow);
201    transition:all ease-in 1s;
202 }
203
204 .output{
205    width: 200px;
206    margin: 10rem 1.5rem;
207    padding: 6px;
208    text-align: center;
209    box-shadow: rgba(0, 0, 0, 0.35) 0px 5px 15px;
210 }
211    .output-container{
212     display: grid;
213     row-gap: 10px;
214     grid-template-areas: 'img1 img2 img3 img4 img5 img6';
215 }
216
217 /* Hidden class */
218 .hidden{
219    visibility: hidden;
220 }
221 .hide{
222    visibility: hidden;
223 }
```

JScript.js

```javascript
1    'use strict'
2    const demo = document.querySelector('#demo');
3    const imageUpload = document.getElementById('imageupload');
```

```
4    const dataAttributeEL = document.querySelectorAll(`div[data-type]`);
5    const displayAll = function () {
6        dataAttributeEL.forEach(el => {
7            el.classList.remove('hidden')
8        })
9    }
10
11
12   imageUpload.addEventListener('change', (event) => {
13       const fileList = event.target.files[0];
14
15       //console.log(URL.createObjectURL(fileList));
16       if (fileList) {
17           demo.src =URL.createObjectURL(fileList);
18       }
19       displayAll();
20
21   });
22
23   const prediction = document.querySelector('#result')
24   dataAttributeEL.forEach(el => {
25       if (el.dataset.type !== prediction.innerHTML.trim()) {
26           el.classList.add('hidden')
27       };
28   })
```

main.js

```
1    $(document).ready(function () {
2        // Init
3        $('.image-section').hide();
4        $('.loader').hide();
5        $('#result').hide();
6
7        // Upload Preview
8        function readURL(input) {
9            if (input.files && input.files[0]) {
10               var reader = new FileReader();
11               reader.onload = function (e) {
12                   $('#imagePreview').css('background-image', 'url(' + e.target.result + ')');
13                   $('#imagePreview').hide();
14                   $('#imagePreview').fadeIn(650);
15               }
16               reader.readAsDataURL(input.files[0]);
```

```
17        }
18      }
19      $("#imageUpload").change(function () {
20        $('.image-section').show();
21        $('#btn-predict').show();
22        $('#result').text('');
23        $('#result').hide();
24        readURL(this);
25      });
26
27      // Predict
28      $('#btn-predict').click(function () {
29        var form_data = new FormData($('#upload-file')[0]);
30
31        // Show loading animation
32        $(this).hide();
33        $('.loader').show();
34
35        // Make prediction by calling api /predict
36        $.ajax({
37          type: 'POST',
38          url: '/predict',
39          data: form_data,
40          contentType: false,
41          cache: false,
42          processData: false,
43          async: true,
44          success: function (data) {
45            // Get and display the result
46            $('.loader').hide();
47            $('#result').fadeIn(600);
48            $('#result').text('Prediction: '+data);
49            console.log('Success!');
50          },
51        });
52      });
53
54  });
```

7.2 FEATURE 2

app.py

```
1  import re
2  import numpy as np
3  import os
```

```python
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\manik\Desktop\yolo_structure\yolo_structure-master\2_Training\src'
print(src_path)
utils_path = r'C:\Users\manik\Desktop\yolo_structure\yolo_structure-master\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
```

```python
data_folder = os.path.join(get_parent_dir(n=1), "Skin Disease-Flask", "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('2eb40045-a8d6-450d-9d24-52cc7cbb2810-
bluemix','Ud0wunTPOI_8h5ZtEqi1IXk1gIKeYLmpUsCn0EeO8T4z', 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")

```

```python
91  #registration page
92  @app.route('/register')
93  def register():
94      return render_template('register.html')
95
96  @app.route('/afterreg', methods=['POST'])
97  def afterreg():
98      x = [x for x in request.form.values()]
99      print(x)
100     data = {
101     '_id': x[1], # Setting _id is optional
102     'name': x[0],
103     'psw':x[2]
104     }
105     print(data)
106
107     query = {'_id': {'$eq': data['_id']}}
108
109     docs = my_database.get_query_result(query)
110     print(docs)
111
112     print(len(docs.all()))
113
114     if(len(docs.all())==0):
115         url = my_database.create_document(data)
116         #response = requests.get(url)
117         return render_template('register.html', pred="Registration Successful, please login using
    your details")
118     else:
119         return render_template('register.html', pred="You are already a member, please login using
    your details")
120
121 #login page
122 @app.route('/login')
123 def login():
124     return render_template('login.html')
125
126 @app.route('/afterlogin',methods=['POST'])
127 def afterlogin():
128     user = request.form['_id']
129     passw = request.form['psw']
130     print(user,passw)
131
132     query = {'_id': {'$eq': user}}
```

```python
133
134    docs = my_database.get_query_result(query)
135    print(docs)
136
137    print(len(docs.all()))
138
139
140    if(len(docs.all())==0):
141        return render_template('login.html', pred="The username is not found.")
142    else:
143        if((user==docs[0][0]['_id'] and passw==docs[0][0]['psw'])):
144            return redirect(url_for('prediction'))
145        else:
146            print('Invalid User')
147
148
149 @app.route('/logout')
150 def logout():
151    return render_template('logout.html')
152
153 @app.route('/prediction')
154 def prediction():
155    return render_template('prediction.html')
156
157
158 @app.route('/result',methods=["GET","POST"])
159 def res():
160    # Delete all default flags
161    parser = argparse.ArgumentParser(argument_default=argparse.SUPPRESS)
162    """
163    Command line options
164    """
165
166    parser.add_argument(
167        "--input_path",
168        type=str,
169        default=image_test_folder,
170        help="Path to image/video directory. All subdirectories will be included. Default is "
171        + image_test_folder,
172    )
173
174    parser.add_argument(
175        "--output",
176        type=str,
```

```
177        default=detection_results_folder,
178        help="Output path for detection results. Default is "
179        + detection_results_folder,
180    )
181
182    parser.add_argument(
183        "--no_save_img",
184        default=False,
185        action="store_true",
186        help="Only save bounding box coordinates but do not save output images with annotated
    boxes. Default is False.",
187    )
188
189    parser.add_argument(
190        "--file_types",
191        "--names-list",
192        nargs="*",
193        default=[],
194        help="Specify list of file types to include. Default is --file_types .jpg .jpeg .png .mp4",
195    )
196
197    parser.add_argument(
198        "--yolo_model",
199        type=str,
200        dest="model_path",
201        default=model_weights,
202        help="Path to pre-trained weight files. Default is " + model_weights,
203    )
204
205    parser.add_argument(
206        "--anchors",
207        type=str,
208        dest="anchors_path",
209        default=anchors_path,
210        help="Path to YOLO anchors. Default is " + anchors_path,
211    )
212
213    parser.add_argument(
214        "--classes",
215        type=str,
216        dest="classes_path",
217        default=model_classes,
218        help="Path to YOLO class specifications. Default is " + model_classes,
219    )
```

```python
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)
```

```python
264    # Split images and videos
265    img_endings = (".jpg", ".jpeg", ".png")
266    vid_endings = (".mp4", ".mpeg", ".mpg", ".avi")
267
268    input_image_paths = []
269    input_video_paths = []
270    for item in input_paths:
271        if item.endswith(img_endings):
272            input_image_paths.append(item)
273        elif item.endswith(vid_endings):
274            input_video_paths.append(item)
275
276    output_path = FLAGS.output
277    if not os.path.exists(output_path):
278        os.makedirs(output_path)
279
280    # define YOLO detector
281    yolo = YOLO(
282        **{
283            "model_path": FLAGS.model_path,
284            "anchors_path": FLAGS.anchors_path,
285            "classes_path": FLAGS.classes_path,
286            "score": FLAGS.score,
287            "gpu_num": FLAGS.gpu_num,
288            "model_image_size": (416, 416),
289        }
290    )
291
292    # Make a dataframe for the prediction outputs
293    out_df = pd.DataFrame(
294        columns=[
295            "image",
296            "image_path",
297            "xmin",
298            "ymin",
299            "xmax",
300            "ymax",
301            "label",
302            "confidence",
303            "x_size",
304            "y_size",
305        ]
306    )
307
```

```python
308    # labels to draw on images
309    class_file = open(FLAGS.classes_path, "r")
310    input_labels = [line.rstrip("\n") for line in class_file.readlines()]
311    print("Found {} input labels: {} ...".format(len(input_labels), input_labels))
312
313    if input_image_paths:
314        print(
315            "Found {} input images: {} ...".format(
316                len(input_image_paths),
317                [os.path.basename(f) for f in input_image_paths[:5]],
318            )
319        )
320        start = timer()
321        text_out = ""
322
323        # This is for images
324        for i, img_path in enumerate(input_image_paths):
325            print(img_path)
326            prediction, image,lat,lon= detect_object(
327                yolo,
328                img_path,
329                save_img=save_img,
330                save_img_path=FLAGS.output,
331                postfix=FLAGS.postfix,
332            )
333            print(lat,lon)
334            y_size, x_size, _ = np.array(image).shape
335            for single_prediction in prediction:
336                out_df = out_df.append(
337                    pd.DataFrame(
338                        [
339                            [
340                                os.path.basename(img_path.rstrip("\n")),
341                                img_path.rstrip("\n"),
342                            ]
343                            + single_prediction
344                            + [x_size, y_size]
345                        ],
346                        columns=[
347                            "image",
348                            "image_path",
349                            "xmin",
350                            "ymin",
351                            "xmax",
```

```python
                    "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')
```

```
396
397 """ Running our application """
398 if __name__ == "__main__":
399     app.run(debug=True)
```

➤ All the html files are attached in this file and the yolo model is deployed and attached here aswell.

Convert_Format.py

```python
1   import os
2   import re
3   from os import makedirs, path
4
5   import numpy as np
6   import pandas as pd
7   from PIL import Image
8
9   from Get_File_Paths import ChangeToOtherMachine, GetFileList
10
11
12  def convert_vott_csv_to_yolo(
13      vott_df,
14      labeldict=dict(zip(["Cat_Face"], [0,])),
15      path="",
16      target_name="data_train.txt",
17      abs_path=False,
18  ):
19
20      # Encode labels according to labeldict if code's don't exist
21      if not "code" in vott_df.columns:
22          vott_df["code"] = vott_df["label"].apply(lambda x: labeldict[x])
23      # Round float to ints
24      for col in vott_df[["xmin", "ymin", "xmax", "ymax"]]:
25          vott_df[col] = (vott_df[col]).apply(lambda x: round(x))
26
27      # Create Yolo Text file
28      last_image = ""
29      txt_file = ""
30
31      for index, row in vott_df.iterrows():
32          if not last_image == row["image"]:
33              if abs_path:
```

```python
34              txt_file += "\n" + row["image_path"] + " "
35          else:
36              txt_file += "\n" + os.path.join(path, row["image"]) + " "
37          txt_file += ",".join(
38              [
39                  str(x)
40                  for x in (row[["xmin", "ymin", "xmax", "ymax", "code"]].tolist())
41              ]
42          )
43      else:
44          txt_file += " "
45          txt_file += ",".join(
46              [
47                  str(x)
48                  for x in (row[["xmin", "ymin", "xmax", "ymax", "code"]].tolist())
49              ]
50          )
51      last_image = row["image"]
52   file = open(target_name, "w")
53   file.write(txt_file[1:])
54   file.close()
55   return True


def csv_from_xml(directory, path_name=""):
    # First get all images and xml files from path and its subfolders
    image_paths = GetFileList(directory, ".jpg")
    xml_paths = GetFileList(directory, ".xml")
    result_df = pd.DataFrame()
    if not len(image_paths) == len(xml_paths):
        print("number of annotations doesnt match number of images")
        return False
    for image in image_paths:
        target_filename = os.path.join(path_name, image) if path_name else image
        source_filename = os.path.join(directory, image)
        y_size, x_size, _ = np.array(Image.open(source_filename)).shape
        source_xml = image.replace(".jpg", ".xml")
        txt = open(source_xml, "r").read()
        y_vals = re.findall(r"(?:x>\n)(.*)(?:\n</)", txt)
        ymin_vals = y_vals[::2]
        ymax_vals = y_vals[1::2]
        x_vals = re.findall(r"(?:y>\n)(.*)(?:\n</)", txt)
        xmin_vals = x_vals[::2]
```

```python
77          xmax_vals = x_vals[1::2]
78          label_vals = re.findall(r"(?:label>\n)(.*)(?:\n</)", txt)
79          label_name_vals =
  re.findall(r"(?:labelname>\n)(.*)(?:\n</)", txt)
80          df = pd.DataFrame()
81          df["xmin"] = xmin_vals
82          df["xmin"] = df["xmin"].astype(float) * x_size
83          df["ymin"] = ymin_vals
84          df["ymin"] = df["ymin"].astype(float) * y_size
85          df["xmax"] = xmax_vals
86          df["xmax"] = df["xmax"].astype(float) * x_size
87          df["ymax"] = ymax_vals
88          df["ymax"] = df["ymax"].astype(float) * y_size
89          df["label"] = label_name_vals
90          df["code"] = label_vals
91          df["image_path"] = target_filename
92          df["image"] = os.path.basename(target_filename)
93          result_df = result_df.append(df)
94      #       Bring image column first
95      cols = list(df.columns)
96      cols = [cols[-1]] + cols[:-1]
97      result_df = result_df[cols]
98      return result_df
99
100
101 def crop_and_save(
102     image_df,
103     target_path,
104     target_file,
105     one=True,
106     label_dict={0: "house"},
107     postfix="cropped",
108 ):
109     """Takes a vott_csv file with image names, labels and
  crop_boxes
110     and crops the images accordingly
111
112     Input csv file format:
113
114     image    xmin ymin xmax ymax label
```

```
115      im.jpg  0    10   100  500  house
116
117      Parameters
118      ----------
119      df : pd.Dataframe
120          The input dataframe with file_names, bounding box info
121          and label
122      source_path : str
123          Path of source images
124      target_path : str, optional
125          Path to save cropped images
126      one : boolean, optional
127          if True, only the most central house will be returned
128      Returns
129      -------
130      True if completed succesfully
131      """
132      if not path.isdir(target_path):
133          makedirs(target_path)
134
135      previous_name = ""
136      counter = 0
137      image_df.dropna(inplace=True)
138      image_df["image_path"] =
    ChangeToOtherMachine(image_df["image_path"].values)
139
140      def find_rel_position(row):
141          current_name = row["image_path"]
142          x_size, _ = Image.open(current_name).size
143          x_centrality = abs((row["xmin"] + row["xmax"]) / 2 /
    x_size - 0.5)
144          return x_centrality
145
146      if one:
147          centrality = []
148          for index, row in image_df.iterrows():
149              centrality.append(find_rel_position(row))
150          image_df["x_centrality"] = pd.Series(centrality)
151          image_df.sort_values(["image", "x_centrality"],
    inplace=True)
```

```python
152          image_df.drop_duplicates(subset="image", keep="first",
    inplace=True)
153      new_paths = []
154      for index, row in image_df.iterrows():
155          current_name = row["image_path"]
156          if current_name == previous_name:
157              counter += 1
158          else:
159              counter = 0
160          imageObject = Image.open(current_name)
161          cropped = imageObject.crop((row["xmin"], row["ymin"],
    row["xmax"], row["ymax"]))
162          label = row["label"]
163          if type(label) == int:
164              label = label_dict[label]
165          image_name_cropped = (
166              "_".join([row["image"][:-4], postfix, label,
    str(counter)]) + ".jpg"
167          )
168          new_path = os.path.join(target_path, image_name_cropped)
169          cropped.save(new_path)
170          new_paths.append(new_path.replace("\\", "/"))
171          previous_name = current_name
172      pd.DataFrame(new_paths,
    columns=["image_path"]).to_csv(target_file)
173      return True
174
175
176 if __name__ == "__main__":
177      # Prepare the houses dataset for YOLO
178      labeldict = dict(zip(["house"], [0,]))
179
180      multi_df =
    r"C:\Users\Admin\Desktop\yolo_structure\Data\Source_Images\Traini
    ng_Images\vott-csv-export\Annotations-export.csv"
181
182      convert_vott_csv_to_yolo(
183          multi_df,
184          labeldict,
185          path=r"C:\Users\Admin\Desktop\data\skin",
```

```
186            target_name= "data_train.txt"
187     )
188
189      # Prepare the windows dataset for YOLO
190      path =
   r"C:\Users\Admin\Desktop\yolo_structure\Data\Source_Images\base"
191      csv_from_xml(path,
   r"C:\Users\Admin\Desktop\data\windows").to_csv(r"C:\Users\Admin\D
   esktop\yolo_structure\Data\Source_Images\base/annotations.csv")
192
193      label_names = [
194          "Erythema multiforme (EM)",
195          "Erythema chronicum migrans",
196          "Erythema migrans",
197          "Erythema marginatum",
198          "Erythema infectiosum",
199          "Erythema nodosum"
200      ]
201      labeldict = dict(zip(label_names, list(range(6))))
202      convert_vott_csv_to_yolo(
203          csv_from_xml(path,
   r"C:\Users\Admin\Desktop\data\windows"), labeldict
204      )
```

Convert_to_YOLO_format.py

```
1  from PIL import Image
2  from os import path, makedirs
3  import os
4  import re
5  import pandas as pd
6  import sys
7  import argparse
8
9
10 from Convert_Format import convert_vott_csv_to_yolo
11
12
13 def get_parent_dir(n=1):
```

```python
14      """ returns the n-th parent dicrectory of the current
15      working directory """
16      current_path = os.path.dirname(os.path.abspath(__file__))
17      for k in range(n):
18          current_path = os.path.dirname(current_path)
19      return current_path
20
21
22 sys.path.append(os.path.join(get_parent_dir(1), "Utils"))
23
24
25 Data_Folder = os.path.join(get_parent_dir(1), "Data")
26 VoTT_Folder = os.path.join(
27      Data_Folder, "Source_Images", "Training_Images", "vott-csv-
    export"
28 )
29 VoTT_csv = os.path.join(VoTT_Folder, "Annotations-export.csv")
30 YOLO_filename = os.path.join(VoTT_Folder, "data_train.txt")
31
32 model_folder = os.path.join(Data_Folder, "Model_Weights")
33 classes_filename = os.path.join(model_folder, "data_classes.txt")
34
35 if __name__ == "__main__":
36      # surpress any inhereted default values
37      parser =
    argparse.ArgumentParser(argument_default=argparse.SUPPRESS)
38      """
39      Command line options
40      """
41      parser.add_argument(
42          "--VoTT_Folder",
43          type=str,
44          default=VoTT_Folder,
45          help="Absolute path to the exported files from the image
    tagging step with VoTT. Default is "
46          + VoTT_Folder,
47      )
48
49      parser.add_argument(
50          "--VoTT_csv",
```

```python
51             type=str,
52             default=VoTT_csv,
53             help="Absolute path to the *.csv file exported from VoTT.
   Default is "
54             + VoTT_csv,
55         )
56     parser.add_argument(
57         "--YOLO_filename",
58         type=str,
59         default=YOLO_filename,
60         help="Absolute path to the file where the annotations in
   YOLO format should be saved. Default is "
61         + YOLO_filename,
62     )
63
64     FLAGS = parser.parse_args()
65
66     # Prepare the dataset for YOLO
67     multi_df = pd.read_csv(FLAGS.VoTT_csv)
68     labels = multi_df["label"].unique()
69     labeldict = dict(zip(labels, range(len(labels))))
70     multi_df.drop_duplicates(subset=None, keep="first",
   inplace=True)
71     train_path = FLAGS.VoTT_Folder
72     convert_vott_csv_to_yolo(
73         multi_df, labeldict, path=train_path,
   target_name=FLAGS.YOLO_filename
74     )
75
76     # Make classes file
77     file = open(classes_filename, "w")
78
79     # Sort Dict by Values
80     SortedLabelDict = sorted(labeldict.items(), key=lambda x:
   x[1])
81     for elem in SortedLabelDict:
82         file.write(elem[0] + "\n")
83     file.close()
```

Get_File_Paths.py

```python
1  from os import path, makedirs
2  import os
3
4  """
5  For the given path, get the List of all files in the directory
   tree
6  https://thispointer.com/python-how-to-get-list-of-files-in-
   directory-and-sub-directories/
7  """
8
9
10 def GetFileList(dirName, endings=[".jpg", ".jpeg", ".png",
   ".mp4"]):
11     # create a list of file and sub directories
12     # names in the given directory
13     listOfFile = os.listdir(dirName)
14     allFiles = list()
15     # Make sure all file endings start with a '.'
16
17     for i, ending in enumerate(endings):
18         if ending[0] != ".":
19             endings[i] = "." + ending
20     # Iterate over all the entries
21     for entry in listOfFile:
22         # Create full path
23         fullPath = os.path.join(dirName, entry)
24         # If entry is a directory then get the list of files in
   this directory
25         if os.path.isdir(fullPath):
26             allFiles = allFiles + GetFileList(fullPath, endings)
27         else:
28             for ending in endings:
29                 if entry.endswith(ending):
30                     allFiles.append(fullPath)
31     return allFiles
32
33
34 def ChangeToOtherMachine(filelist, repo="TrainYourOwnYOLO",
```

```python
35      remote_machine=""):
36          """
            Takes a list of file_names located in a repo and changes it
        to the local machines file names. File must be executed from
        withing the repository
37          Example:
38
        '/home/ubuntu/TrainYourOwnYOLO/Data/Street_View_Images/vulnerable
        /test.jpg'
39          Get's converted to
40
41
        'C:/Users/Anton/TrainYourOwnYOLO/Data/Street_View_Images/vulnerab
        le/test.jpg'
42          """
43          filelist = [x.replace("\\", "/") for x in filelist]
44          if repo[-1] == "/":
45              repo = repo[:-1]
46          if remote_machine:
47              prefix = remote_machine.replace("\\", "/")
48          else:
49              prefix =
        ((os.path.dirname(os.path.abspath(__file__)).split(repo))[0]).rep
        lace(
50                  "\\", "/"
51              )
52          new_list = []
53
54          for file in filelist:
55              suffix = (file.split(repo))[1]
56              if suffix[0] == "/":
57                  suffix = suffix[1:]
58              new_list.append(os.path.join(prefix, repo + "/",
        suffix).replace("\\", "/"))
59          return new_list
```

Download_Weights.py

```python
1  import requests
```

```python
import os
import progressbar


def download_file_from_google_drive(id, destination):
    def get_confirm_token(response):
        for key, value in response.cookies.items():
            if key.startswith("download_warning"):
                return value

        return None

    def save_response_content(response, destination):
        CHUNK_SIZE = 32768

        with open(destination, "wb") as f:
            bar =
  progressbar.ProgressBar(max_value=progressbar.UnknownLength)
            i = 0
            for chunk in response.iter_content(CHUNK_SIZE):
                if chunk:  # filter out keep-alive new chunks
                    bar.update(i)
                    i += 1
                    f.write(chunk)

    URL = "https://docs.google.com/uc?export=download"

    session = requests.Session()

    response = session.get(URL, params={"id": id}, stream=True)
    token = get_confirm_token(response)

    if token:
        params = {"id": id, "confirm": token}
        response = session.get(URL, params=params, stream=True)

    save_response_content(response, destination)


if __name__ == "__main__":
```

```
41      import sys
42
43      if len(sys.argv) is not 3:
44          print("Usage: python google_drive.py drive_file_id
    destination_file_path")
45      else:
46          # TAKE ID FROM SHAREABLE LINK
47          file_id = sys.argv[1]
48          # DESTINATION FILE ON YOUR DISK
49          destination = os.path.join(os.getcwd(), sys.argv[2])
50          download_file_from_google_drive(file_id, destination)
```

utils.py

```
1  import colorsys
2
3  import cv2 as cv
4  import h5py
5  from keras import Model
6  import numpy as np
7  import os
8  from matplotlib.colors import rgb_to_hsv, hsv_to_rgb
9  from PIL import Image, ImageFont, ImageDraw
10 from timeit import default_timer as timer
11 import random
12 # import readline
13 # readline.parse_and_bind("tab: complete")
14
15 min_logo_size = (10, 10)
16
17
18 def detect_object(yolo, img_path, save_img, save_img_path="./",
   postfix=""):
19     """
20     Call YOLO logo detector on input image, optionally save
   resulting image.
21     Args:
22       yolo: keras-yolo3 initialized YOLO instance
23       img_path: path to image file
24       save_img: bool to save annotated image
```

```python
25          save_img_path: path to directory where to save image
26          postfix: string to add to filenames
27      Returns:
28          prediction: list of bounding boxes in format
    (xmin,ymin,xmax,ymax,class_id,confidence)
29          image: unaltered input image as (H,W,C) array
30      """
31      try:
32          image = Image.open(img_path)
33          if image.mode != "RGB":
34              image = image.convert("RGB")
35          image_array = np.array(image)
36      except:
37          print("File Open Error! Try again!")
38          return None, None
39
40      prediction, new_image = yolo.detect_image(image)
41
42      img_out =
    postfix.join(os.path.splitext(os.path.basename(img_path)))
43      lat= random.uniform(-90.00, 90.00)
44      lon= random.uniform(-180.00,180.00)
45      if save_img:
46          new_image.save(os.path.join(save_img_path, img_out))
47
48      return prediction, image_array,lat,lon
49
50
51 def parse_input():
52      """
53      Ask user input for input images: pass path to individual
    images, directory
54      """
55      out = []
56      while True:
57          ins = input("Enter path (q to quit):").strip()
58          if ins in ["q", "quit"]:
59              break
60          if not os.path.exists(ins):
61              print("Error: file not found!")
```

```python
62              elif os.path.isdir(ins):
63                  out = [
64                      os.path.abspath(os.path.join(ins, f))
65                      for f in os.listdir(ins)
66                      if f.endswith((".jpg", ".png"))
67                  ]
68                  break
69              elif ins.endswith((".jpg", ".png")):
70                  out.append(os.path.abspath(ins))
71          print(out)
72      return out
73
74
75  def load_extractor_model(model_name="InceptionV3", flavor=1):
76      """Load variant of InceptionV3 or VGG16 model specified.
77      Args:
78        model_name: string, either InceptionV3 or VGG16
79        flavor: int specifying the model variant and input_shape.
80          For InceptionV3, the map is {0: default, 1: 200*200,
    truncate last Inception block,
81          2: 200*200, truncate last 2 blocks, 3: 200*200, truncate
    last 3 blocks, 4: 200*200}
82          For VGG16, it only changes the input size, {0: 224
    (default), 1: 128, 2: 64}.
83  """
84      start = timer()
85      if model_name == "InceptionV3":
86          from keras.applications.inception_v3 import InceptionV3
87          from keras.applications.inception_v3 import
    preprocess_input
88
89          model = InceptionV3(weights="imagenet",
    include_top=False)
90
91          trunc_layer = [-1, 279, 248, 228, -1]
92          i_layer = flavor
93          model_out = Model(
94              inputs=model.inputs,
    outputs=model.layers[trunc_layer[i_layer]].output
95          )
```

```python
96          input_shape = (299, 299, 3) if flavor == 0 else (200,
   200, 3)
97
98      elif model_name == "VGG16":
99          from keras.applications.vgg16 import VGG16
100          from keras.applications.vgg16 import preprocess_input
101
102          model_out = VGG16(weights="imagenet", include_top=False)
103          input_length = [224, 128, 64][flavor]
104          input_shape = (input_length, input_length, 3)
105
106      end = timer()
107      print("Loaded {} feature extractor in
   {:.2f}sec".format(model_name, end - start))
108      return model_out, preprocess_input, input_shape
109
110
111 def chunks(l, n, preprocessing_function=None):
112      """Yield successive n-sized chunks from l.
113      General purpose function modified for Keras: made infinite
   loop,
114      add preprocessing, returns np.array instead of list
115      Args:
116        l: iterable
117        n: number of items to take for each chunk
118        preprocessing_function: function that processes image (3D
   array)
119      Returns:
120        generator with n-sized np.array preprocessed chunks of the
   input
121      """
122
123      func = (lambda x: x) if (preprocessing_function is None)
   else preprocessing_function
124
125      # in predict_generator, steps argument sets how many times
   looped through "while True"
126      while True:
127          for i in range(0, len(l), n):
128              yield np.array([func(el) for el in l[i : i + n]])
```

```python
129
130
131  def load_features(filename):
132      """
133      Load pre-saved HDF5 features for all logos in the
   LogosInTheWild database
134      """
135
136      start = timer()
137      # get database features
138      with h5py.File(filename, "r") as hf:
139          brand_map = list(hf.get("brand_map"))
140          input_shape = list(hf.get("input_shape"))
141          features = hf.get("features")
142          features = np.array(features)
143      end = timer()
144      print(
145          "Loaded {} features from {} in {:.2f}sec".format(
146              features.shape, filename, end - start
147          )
148      )
149
150      return features, brand_map, input_shape
151
152
153  def save_features(filename, features, brand_map, input_shape):
154      """
155      Save features to compressed HDF5 file for later use
156      """
157
158      print("Saving {} features into {}...
   ".format(features.shape, filename), end="")
159      # reduce file size by saving as float16
160      features = features.astype(np.float16)
161      start = timer()
162      with h5py.File(filename, "w") as hf:
163          hf.create_dataset("features", data=features,
   compression="lzf")
164          hf.create_dataset("brand_map", data=brand_map)
165          hf.create_dataset("input_shape", data=input_shape)
```

```python
166
167     end = timer()
168     print("done in {:.2f}sec".format(end - start))
169
170     return None
171
172
173 def features_from_image(img_array, model, preprocess,
    batch_size=100):
174     """
175     Extract features from image array given a decapitated keras
    model.
176     Use a generator to avoid running out of memory for large
    inputs.
177     Args:
178         img_array: (N, H, W, C) list/array of input images
179         model: keras model, outputs
180     Returns:
181         features: (N, F) array of 1D features
182     """
183
184     if len(img_array) == 0:
185         return np.array([])
186
187     steps = len(img_array) // batch_size + 1
188     img_gen = chunks(img_array, batch_size,
    preprocessing_function=preprocess)
189     features = model.predict_generator(img_gen, steps=steps)
190
191     # if the generator has looped past end of array, cut it down
192     features = features[: len(img_array)]
193
194     # reshape features: flatten last three dimensions to one
195     features = features.reshape(features.shape[0],
    np.prod(features.shape[1:]))
196     return features
197
198
199 #################################################
200 # image processing and bounding box functions
```

```python
201 ###################################################
202
203
204 def pad_image(img, shape, mode="constant_mean"):
205     """
206     Resize and pad image to given size.
207     Args:
208       img: (H, W, C) input numpy array
209       shape: (H', W') destination size
210       mode: filling mode for new padded pixels. Default =
   'constant_mean' returns
211         grayscale padding with pixel intensity equal to mean of
   the array. Other
212         options include np.pad() options, such as 'edge', 'mean'
   (by row/column)...
213     Returns:
214       new_im: (H', W', C) padded numpy array
215     """
216     if mode == "constant_mean":
217         mode_args = {"mode": "constant", "constant_values":
   np.mean(img)}
218     else:
219         mode_args = {"mode": mode}
220
221     ih, iw = img.shape[:2]
222     h, w = shape[:2]
223
224     # first rescale image so that largest dimension matches
   target
225     scale = min(w / iw, h / ih)
226     nw, nh = int(iw * scale), int(ih * scale)
227     img = cv.resize(img, (nw, nh))
228
229     # center-pad rest of image: compute padding and split in two
230     xpad, ypad = shape[1] - nw, shape[0] - nh
231     xpad = (xpad // 2, xpad // 2 + xpad % 2)
232     ypad = (ypad // 2, ypad // 2 + ypad % 2)
233
234     new_im = np.pad(img, pad_width=(ypad, xpad, (0, 0)),
   **mode_args)
```

```python
235
236        return new_im
237
238
239  def bbox_colors(n):
240        """
241        Define n distinct bounding box colors
242        Args:
243          n: number of colors
244        Returns:
245          colors: (n, 3) np.array with RGB integer values in [0-255]
  range
246        """
247        hsv_tuples = [(x / n, 1.0, 1.0) for x in range(n)]
248        colors = 255 * np.array([colorsys.hsv_to_rgb(*x) for x in
  hsv_tuples])
249
250        np.random.seed(10101)  # Fixed seed for consistent colors
  across runs.
251        np.random.shuffle(colors)  # Shuffle colors to decorrelate
  adjacent classes.
252        np.random.seed(None)  # Reset seed to default.
253
254        return colors.astype(int)
255
256
257  def contents_of_bbox(img, bbox_list, expand=1.0):
258        """
259        Extract portions of image inside  bounding boxes list.
260        Args:
261          img: (H,W,C) image array
262          bbox_list: list of bounding box specifications, with first
  4 elements
263          specifying box corners in (xmin, ymin, xmax, ymax) format.
264        Returns:
265          candidates: list of 3D image arrays
266          i_candidates_too_small: list of indices of small
  candidates dropped
267        """
268
```

```python
269        candidates = []
270        i_candidates_too_small = []
271        for i, (xmin, ymin, xmax, ymax, *_) in enumerate(bbox_list):
272
273            # for very low confidence sometimes logos found outside
    of the image
274            if ymin > img.shape[0] or xmin > img.shape[1]:
275                continue
276
277            xmin, ymin = int(xmin // expand), int(ymin // expand)
278            xmax, ymax = int(np.round(xmax // expand)),
    int(np.round(ymax // expand))
279
280            # do not even consider tiny logos
281            if xmax - xmin > min_logo_size[1] and ymax - ymin >
    min_logo_size[0]:
282                candidates.append(img[ymin:ymax, xmin:xmax])
283            else:
284                i_candidates_too_small.append(i)
285
286        return candidates, i_candidates_too_small
287
288
289 def draw_annotated_box(image, box_list_list, label_list,
    color_list):
290        """
291        Draw box and overhead label on image.
292        Args:
293            image: PIL image object
294            box_list_list: list of lists of bounding boxes, one for
    each label, each box in
295                (xmin, ymin, xmax, ymax [, score]) format (where score
    is an optional float)
296            label_list: list of  string to go above box
297            color_list: list of RGB tuples
298        Returns:
299            image: annotated PIL image object
300        """
301
302        font_path = os.path.join(
```

```python
303            os.path.dirname(__file__), "keras_yolo3/font/FiraMono-
    Medium.otf"
304        )
305        font = ImageFont.truetype(
306            font=font_path, size=np.floor(3e-2 * image.size[1] +
    0.5).astype("int32")
307        )
308        thickness = (image.size[0] + image.size[1]) // 300
309
310        draw = ImageDraw.Draw(image)
311
312        for box_list, label, color in zip(box_list_list, label_list,
    color_list):
313            if not isinstance(color, tuple):
314                color = tuple(color)
315            for box in box_list:
316                # deal with empty predictions
317                if len(box) < 4:
318                    continue
319
320                # if score is also passed, append to label
321                thelabel = "{}".format(label)
322                if len(box) > 4:
323                    thelabel += " {:.2f}".format(box[-1])
324                label_size = draw.textsize(thelabel, font)
325
326                xmin, ymin, xmax, ymax = box[:4]
327                ymin = max(0, np.floor(ymin + 0.5).astype("int32"))
328                xmin = max(0, np.floor(xmin + 0.5).astype("int32"))
329                ymax = min(image.size[1], np.floor(ymax +
    0.5).astype("int32"))
330                xmax = min(image.size[0], np.floor(xmax +
    0.5).astype("int32"))
331
332                if ymin - label_size[1] >= 0:
333                    text_origin = np.array([xmin, ymin -
    label_size[1]])
334                else:
335                    text_origin = np.array([xmin, ymax])
336
```

```
337                for i in range(thickness):
338                    draw.rectangle([xmin + i, ymin + i, xmax - i,
      ymax - i], outline=color)
339                draw.rectangle(
340                    [tuple(text_origin), tuple(text_origin +
      label_size)], fill=color
341                )
342                draw.text(text_origin, thelabel, fill=(0, 0, 0),
      font=font)
343
344        del draw
345
346        return image
```

Train_YOLO.py

```
1  import argparse
2  import os
3  import sys
4  import warnings
5
6
7  def get_parent_dir(n=1):
8      """ returns the n-th parent dicrectory of the current
9      working directory """
10     current_path = os.path.dirname(os.path.abspath(__file__))
11     for k in range(n):
12         current_path = os.path.dirname(current_path)
13     return current_path
14
15
16 src_path = os.path.join(get_parent_dir(0), "src")
17 sys.path.append(src_path)
18
19 utils_path = os.path.join(get_parent_dir(1), "Utils")
20 sys.path.append(utils_path)
21
22 import pickle
23 from time import time
24
```

```python
25 import keras.backend as K
26 import numpy as np
27 import tensorflow.compat.v1 as tf
28 from keras.callbacks import (EarlyStopping, ModelCheckpoint,
   ReduceLROnPlateau,
29                              TensorBoard)
30 from keras.layers import Input, Lambda
31 from keras.models import Model
32 from keras.optimizers import Adam
33 from keras_yolo3.yolo3.model import (preprocess_true_boxes,
   tiny_yolo_body,
34                                      yolo_body, yolo_loss)
35 from keras_yolo3.yolo3.utils import get_random_data
36 from Train_Utils import (ChangeToOtherMachine, create_model,
   create_tiny_model,
37                         data_generator, data_generator_wrapper,
   get_anchors,
38                         get_classes)
39
40 keras_path = os.path.join(src_path, "keras_yolo3")
41 Data_Folder = os.path.join(get_parent_dir(1), "Data")
42 Image_Folder = os.path.join(Data_Folder, "Source_Images",
   "Training_Images")
43 VoTT_Folder = os.path.join(Image_Folder, "vott-csv-export")
44 YOLO_filename = os.path.join(VoTT_Folder, "data_train.txt")
45
46 Model_Folder = os.path.join(Data_Folder, "Model_Weights")
47 YOLO_classname = os.path.join(Model_Folder, "data_classes.txt")
48
49 log_dir = Model_Folder
50 anchors_path = os.path.join(keras_path, "model_data",
   "yolo_anchors.txt")
51 weights_path = os.path.join(keras_path, "yolo.h5")
52
53 FLAGS = None
54
55 if __name__ == "__main__":
56     # Delete all default flags
57     parser =
   argparse.ArgumentParser(argument_default=argparse.SUPPRESS)
```

```python
58      """
59      Command line options
60      """
61
62      parser.add_argument(
63          "--annotation_file",
64          type=str,
65          default=YOLO_filename,
66          help="Path to annotation file for Yolo. Default is " +
   YOLO_filename,
67      )
68      parser.add_argument(
69          "--classes_file",
70          type=str,
71          default=YOLO_classname,
72          help="Path to YOLO classnames. Default is " +
   YOLO_classname,
73      )
74
75      parser.add_argument(
76          "--log_dir",
77          type=str,
78          default=log_dir,
79          help="Folder to save training logs and trained weights
   to. Default is "
80          + log_dir,
81      )
82
83      parser.add_argument(
84          "--anchors_path",
85          type=str,
86          default=anchors_path,
87          help="Path to YOLO anchors. Default is " + anchors_path,
88      )
89
90      parser.add_argument(
91          "--weights_path",
92          type=str,
93          default=weights_path,
94          help="Path to pre-trained YOLO weights. Default is " +
```

```python
        weights_path,
95      )
96      parser.add_argument(
97          "--val_split",
98          type=float,
99          default=0.1,
100         help="Percentage of training set to be used for
    validation. Default is 10%.",
101     )
102     parser.add_argument(
103         "--is_tiny",
104         default=False,
105         action="store_true",
106         help="Use the tiny Yolo version for better performance
    and less accuracy. Default is False.",
107     )
108     parser.add_argument(
109         "--random_seed",
110         type=float,
111         default=None,
112         help="Random seed value to make script deterministic.
    Default is 'None', i.e. non-deterministic.",
113     )
114     parser.add_argument(
115         "--epochs",
116         type=float,
117         default=51,
118         help="Number of epochs for training last layers and
    number of epochs for fine-tuning layers. Default is 51.",
119     )
120     parser.add_argument(
121         "--warnings",
122         default=False,
123         action="store_true",
124         help="Display warning messages. Default is False.",
125     )
126
127     FLAGS = parser.parse_args()
128
129     if not FLAGS.warnings:
```

```python
            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)
```

```python
166     early_stopping = EarlyStopping(
167         monitor="val_loss", min_delta=0, patience=10, verbose=1
168     )
169
170     val_split = FLAGS.val_split
171     with open(FLAGS.annotation_file) as f:
172         lines = f.readlines()
173
174     # This step makes sure that the path names correspond to the
    local machine
175     # This is important if annotation and training are done on
    different machines (e.g. training on AWS)
176     lines = ChangeToOtherMachine(lines, remote_machine="")
177     np.random.shuffle(lines)
178     num_val = int(len(lines) * val_split)
179     num_train = len(lines) - num_val
180
181     # Train with frozen layers first, to get a stable loss.
182     # Adjust num epochs to your dataset. This step is enough to
    obtain a decent model.
183     if True:
184         model.compile(
185             optimizer=Adam(lr=1e-3),
186             loss={
187                 # use custom yolo_loss Lambda layer.
188                 "yolo_loss": lambda y_true, y_pred: y_pred
189             },
190         )
191
192         batch_size = 32
193         print(
194             "Train on {} samples, val on {} samples, with batch
    size {}.".format(
195                 num_train, num_val, batch_size
196             )
197         )
198         history = model.fit_generator(
199             data_generator_wrapper(
200                 lines[:num_train], batch_size, input_shape,
    anchors, num_classes
```

```
201                ),
202                steps_per_epoch=max(1, num_train // batch_size),
203                validation_data=data_generator_wrapper(
204                    lines[num_train:], batch_size, input_shape,
    anchors, num_classes
205                ),
206                validation_steps=max(1, num_val // batch_size),
207                epochs=epoch1,
208                initial_epoch=0,
209                callbacks=[logging, checkpoint],
210            )
211            model.save_weights(os.path.join(log_dir,
    "trained_weights_stage_1.h5"))
212
213            step1_train_loss = history.history["loss"]
214
215            file = open(os.path.join(log_dir_time,
    "step1_loss.npy"), "w")
216            with open(os.path.join(log_dir_time, "step1_loss.npy"),
    "w") as f:
217                for item in step1_train_loss:
218                    f.write("%s\n" % item)
219            file.close()
220
221            step1_val_loss = np.array(history.history["val_loss"])
222
223            file = open(os.path.join(log_dir_time,
    "step1_val_loss.npy"), "w")
224            with open(os.path.join(log_dir_time,
    "step1_val_loss.npy"), "w") as f:
225                for item in step1_val_loss:
226                    f.write("%s\n" % item)
227            file.close()
228
229        # Unfreeze and continue training, to fine-tune.
230        # Train longer if the result is unsatisfactory.
231        if True:
232            for i in range(len(model.layers)):
233                model.layers[i].trainable = True
234            model.compile(
```

```python
235            optimizer=Adam(lr=1e-4), loss={"yolo_loss": lambda
    y_true, y_pred: y_pred}
236        )  # recompile to apply the change
237        print("Unfreeze all layers.")
238
239        batch_size = (
240            4  # note that more GPU memory is required after
    unfreezing the body
241        )
242        print(
243            "Train on {} samples, val on {} samples, with batch
    size {}.".format(
244                num_train, num_val, batch_size
245            )
246        )
247        history = model.fit_generator(
248            data_generator_wrapper(
249                lines[:num_train], batch_size, input_shape,
    anchors, num_classes
250            ),
251            steps_per_epoch=max(1, num_train // batch_size),
252            validation_data=data_generator_wrapper(
253                lines[num_train:], batch_size, input_shape,
    anchors, num_classes
254            ),
255            validation_steps=max(1, num_val // batch_size),
256            epochs=epoch1 + epoch2,
257            initial_epoch=epoch1,
258            callbacks=[logging, checkpoint, reduce_lr,
    early_stopping],
259        )
260        model.save_weights(os.path.join(log_dir,
    "trained_weights_final.h5"))
261        step2_train_loss = history.history["loss"]
262
263        file = open(os.path.join(log_dir_time,
    "step2_loss.npy"), "w")
264        with open(os.path.join(log_dir_time, "step2_loss.npy"),
    "w") as f:
265            for item in step2_train_loss:
```

```
266                    f.write("%s\n" % item)
267            file.close()
268
269            step2_val_loss = np.array(history.history["val_loss"])
270
271            file = open(os.path.join(log_dir_time,
   "step2_val_loss.npy"), "w")
272            with open(os.path.join(log_dir_time,
   "step2_val_loss.npy"), "w") as f:
273                for item in step2_val_loss:
274                    f.write("%s\n" % item)
275            file.close()
```

8. TESTING

8.1 TEST CASES

| Test Case- I: Upload The picture of affected area |
|---|
| Precondition: The application should be open |
| Assumptions: All the datasets are available |
| Test steps: 1) Register and login
          2) Upload the image |
| Expected Result: The predicted result should be displayed in the prediction page |

8.2 USER ACCEPTANCE TESTING

we have created the application and the prediction process is carried out successfully.
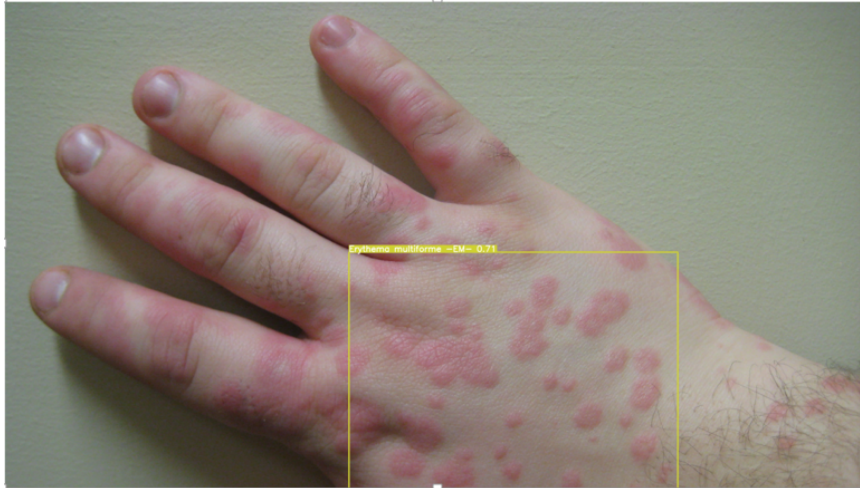
9. RESULTS

So finally when we run the python code it is going to connect the IBM Watson platform and we can able to view the predicted result.

Step 1: This is the interface of our application.
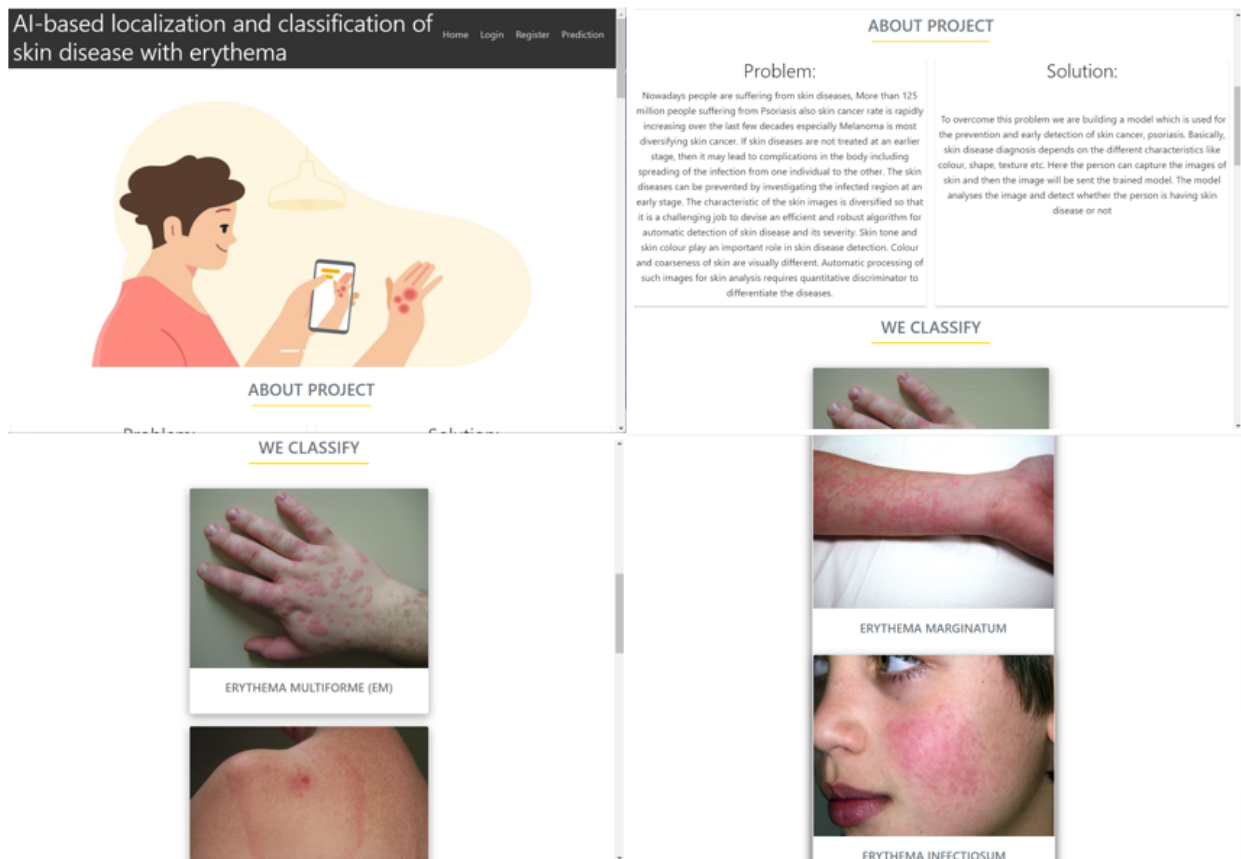
Step 2: New user should register with their information.

Step3: Login using the login page.

Step4: Upload image of the affected area

AI-based localization and classification of skin disease with erythema

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

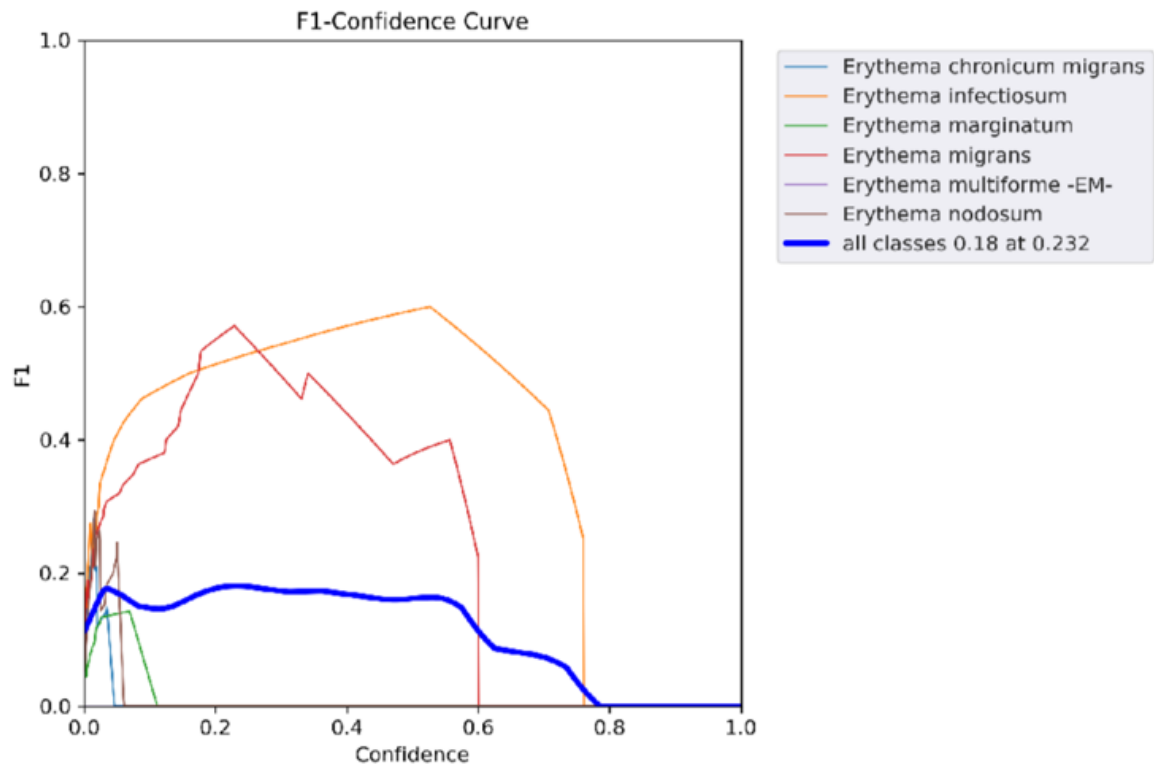Click here for Prediction

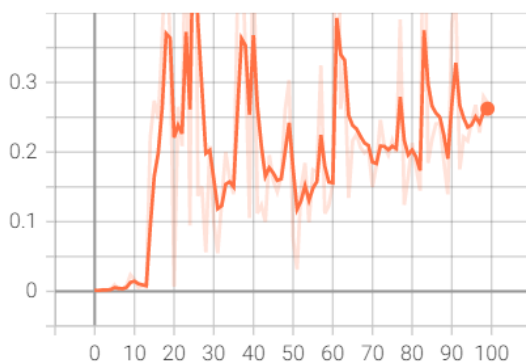**Step5**: The predicted result is displayed

9.1 PERFORMANCE METRICS

tag: F1_curve
step **99**
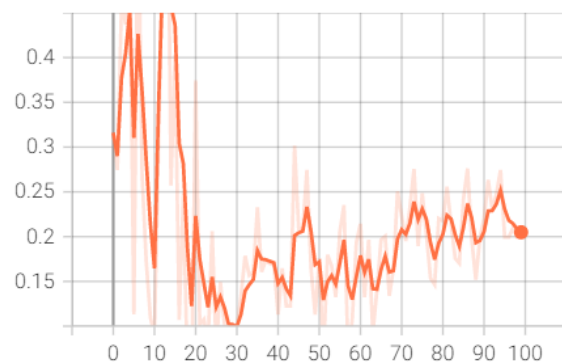
Fri Nov 18 2022 18:35:50 GMT+0530 (India Standard Time)

F1-Confidence Curve

Legend:
- Erythema chronicum migrans
- Erythema infectiosum
- Erythema marginatum
- Erythema migrans
- Erythema multiforme -EM-
- Erythema nodosum
- all classes 0.18 at 0.232

metrics/precision
tag: metrics/precision
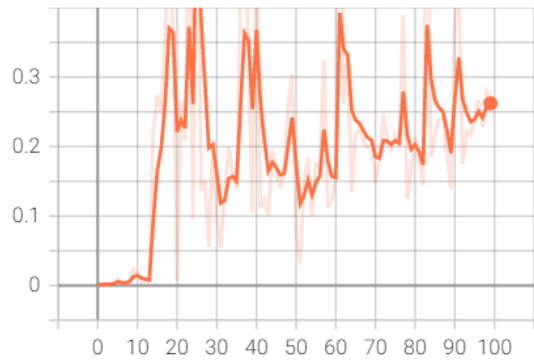

metrics/recall
tag: metrics/recall

metrics/precision
tag: metrics/precision

metrics/recall
tag: metrics/recall

Confusion Matrix

	Erythema chronicum migrans	Erythema infectiosum	Erythema marginatum	Erythema migrans	Erythema multiforme -EM-	Erythema nodosum	background
Erythema chronicum migrans	0.25						
Erythema infectiosum		0.43					
Erythema marginatum				0.12			
Erythema migrans				0.37		0.20	1.00
Erythema multiforme -EM-				0.12			
Erythema nodosum							
background	0.75	0.57	1.00	0.37	1.00	0.80	

Predicted
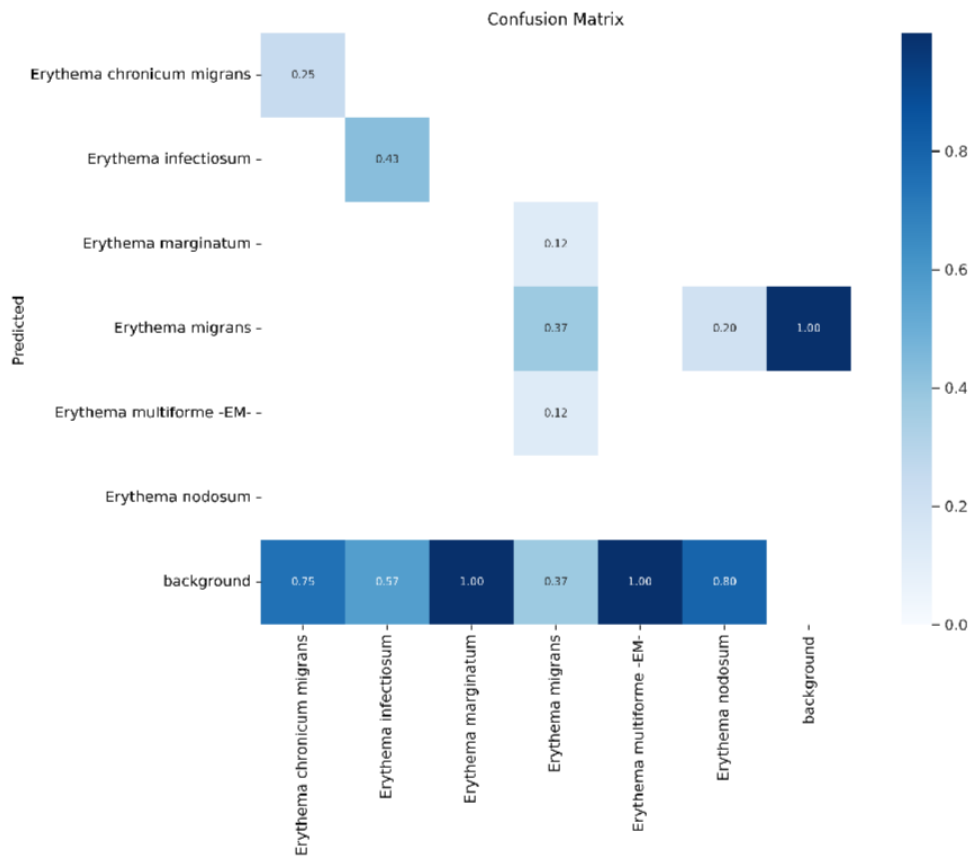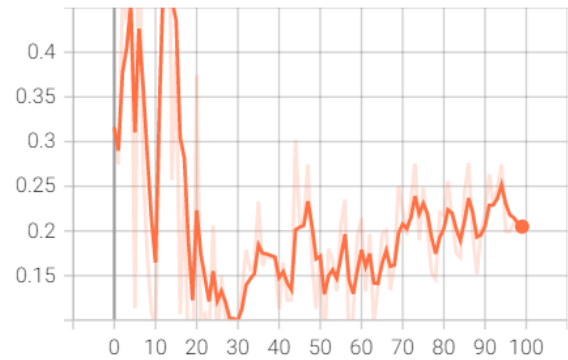
10. ADVANTAGES & DISADVANTAGES

ADVANTAGES:

- We have demonstrated that adequate accuracy rates can be attained even in the absence of a huge dataset and high-quality photos.
- Knowing the location of the disease through accurate segmentation is helpful for preparing data used for classification.

DISADVANTAGES:

- our model is most effective with camera images of skin diseases with erythema, which is a limitation of our study.
- People with no knowledge of using mobile phone will find it difficult to use.

11. CONCLUSION

The web application is successfully created, user needs to provide information, register and login, which is stored and the data is input for the trained algorithms. Here we used algorithms which will produce better accuracy . And finally displayed the output in new html page like if the patient has a condition of erythema otherwise it displays as your skin condition is not erythema . The application is tested fine and properly debugged.

12. FUTURE SCOPE

For future work, we plan tests that aim to show the medication that a patient can take for treatment. Also, we are looking to link this website to various hospitals and pharmacies for easy use.

13. APPENDIX
Source Code

```
1  import re
2  import numpy as np
3  import os
4  from flask import Flask, app,request,render_template
```

```python
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\manik\Desktop\yolo_structure\yolo_structure-master\2_Training\src'
print(src_path)
utils_path = r'C:\Users\manik\Desktop\yolo_structure\yolo_structure-master\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
```

```python
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), "Skin Disease-Flask", "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('2eb40045-a8d6-450d-9d24-52cc7cbb2810-bluemix','Ud0wunTPOI_8h5ZtEqi1IXk1gIKeYLmpUsCn0EeO8T4z', connect=True)
```

```python
73 # Create a database using an initialized client
74 my_database = client.create_database('my_database')
75
76
77 app=Flask(__name__)
78
79 #default home page or route
80 @app.route('/')
81 def index():
82     return render_template('index.html')
83
84
85
86 @app.route('/index.html')
87 def home():
88     return render_template("index.html")
89
90
91 #registration page
92 @app.route('/register')
93 def register():
94     return render_template('register.html')
95
96 @app.route('/afterreg', methods=['POST'])
97 def afterreg():
98     x = [x for x in request.form.values()]
99     print(x)
100     data = {
101     '_id': x[1], # Setting _id is optional
102     'name': x[0],
103     'psw':x[2]
104     }
105     print(data)
106
107     query = {'_id': {'$eq': data['_id']}}
108
109     docs = my_database.get_query_result(query)
110     print(docs)
111
112     print(len(docs.all()))
```

```python
113
114      if(len(docs.all())==0):
115          url = my_database.create_document(data)
116          #response = requests.get(url)
117          return render_template('register.html',
     pred="Registration Successful, please login using your details")
118      else:
119          return render_template('register.html', pred="You are
     already a member, please login using your details")
120
121  #login page
122  @app.route('/login')
123  def login():
124      return render_template('login.html')
125
126  @app.route('/afterlogin',methods=['POST'])
127  def afterlogin():
128      user = request.form['_id']
129      passw = request.form['psw']
130      print(user,passw)
131
132      query = {'_id': {'$eq': user}}
133
134      docs = my_database.get_query_result(query)
135      print(docs)
136
137      print(len(docs.all()))
138
139
140      if(len(docs.all())==0):
141          return render_template('login.html', pred="The username
     is not found.")
142      else:
143          if((user==docs[0][0]['_id'] and
     passw==docs[0][0]['psw'])):
144              return redirect(url_for('prediction'))
145          else:
146              print('Invalid User')
147
148
```

```python
149 @app.route('/logout')
150 def logout():
151     return render_template('logout.html')
152
153 @app.route('/prediction')
154 def prediction():
155     return render_template('prediction.html')
156
157
158 @app.route('/result',methods=["GET","POST"])
159 def res():
160     # Delete all default flags
161     parser =
    argparse.ArgumentParser(argument_default=argparse.SUPPRESS)
162     """
163     Command line options
164     """
165
166     parser.add_argument(
167         "--input_path",
168         type=str,
169         default=image_test_folder,
170         help="Path to image/video directory. All subdirectories
    will be included. Default is "
171         + image_test_folder,
172     )
173
174     parser.add_argument(
175         "--output",
176         type=str,
177         default=detection_results_folder,
178         help="Output path for detection results. Default is "
179         + detection_results_folder,
180     )
181
182     parser.add_argument(
183         "--no_save_img",
184         default=False,
185         action="store_true",
186         help="Only save bounding box coordinates but do not save
```

```python
            output images with annotated boxes. Default is False.",
187     )
188
189     parser.add_argument(
190         "--file_types",
191         "--names-list",
192         nargs="*",
193         default=[],
194         help="Specify list of file types to include. Default is "
    --file_types .jpg .jpeg .png .mp4",
195     )
196
197     parser.add_argument(
198         "--yolo_model",
199         type=str,
200         dest="model_path",
201         default=model_weights,
202         help="Path to pre-trained weight files. Default is " +
    model_weights,
203     )
204
205     parser.add_argument(
206         "--anchors",
207         type=str,
208         dest="anchors_path",
209         default=anchors_path,
210         help="Path to YOLO anchors. Default is " + anchors_path,
211     )
212
213     parser.add_argument(
214         "--classes",
215         type=str,
216         dest="classes_path",
217         default=model_classes,
218         help="Path to YOLO class specifications. Default is " +
    model_classes,
219     )
220
221     parser.add_argument(
222         "--gpu_num", type=int, default=1, help="Number of GPU to
    use. Default is 1"
```

```python
223          )
224
225      parser.add_argument(
226          "--confidence",
227          type=float,
228          dest="score",
229          default=0.25,
230          help="Threshold for YOLO object confidence score to show
    predictions. Default is 0.25.",
231      )
232
233      parser.add_argument(
234          "--box_file",
235          type=str,
236          dest="box",
237          default=detection_results_file,
238          help="File to save bounding box results to. Default is "
239          + detection_results_file,
240      )
241
242      parser.add_argument(
243          "--postfix",
244          type=str,
245          dest="postfix",
246          default="_disease",
247          help='Specify the postfix for images with bounding
    boxes. Default is "_disease"',
248      )
249
250      FLAGS = parser.parse_args()
251
252      save_img = not FLAGS.no_save_img
253
254      file_types = FLAGS.file_types
255      #print(input_path)
256
257      if file_types:
258          input_paths = GetFileList(FLAGS.input_path,
    endings=file_types)
259          print(input_paths)
```

```python
260        else:
261            input_paths = GetFileList(FLAGS.input_path)
262            print(input_paths)
263
264        # Split images and videos
265        img_endings = (".jpg", ".jpeg", ".png")
266        vid_endings = (".mp4", ".mpeg", ".mpg", ".avi")
267
268        input_image_paths = []
269        input_video_paths = []
270        for item in input_paths:
271            if item.endswith(img_endings):
272                input_image_paths.append(item)
273            elif item.endswith(vid_endings):
274                input_video_paths.append(item)
275
276        output_path = FLAGS.output
277        if not os.path.exists(output_path):
278            os.makedirs(output_path)
279
280        # define YOLO detector
281        yolo = YOLO(
282            **{
283                "model_path": FLAGS.model_path,
284                "anchors_path": FLAGS.anchors_path,
285                "classes_path": FLAGS.classes_path,
286                "score": FLAGS.score,
287                "gpu_num": FLAGS.gpu_num,
288                "model_image_size": (416, 416),
289            }
290        )
291
292        # Make a dataframe for the prediction outputs
293        out_df = pd.DataFrame(
294            columns=[
295                "image",
296                "image_path",
297                "xmin",
298                "ymin",
299                "xmax",
```

```python
300                "ymax",
301                "label",
302                "confidence",
303                "x_size",
304                "y_size",
305            ]
306        )
307
308        # labels to draw on images
309        class_file = open(FLAGS.classes_path, "r")
310        input_labels = [line.rstrip("\n") for line in
    class_file.readlines()]
311        print("Found {} input labels: {}
    ...".format(len(input_labels), input_labels))
312
313        if input_image_paths:
314            print(
315                "Found {} input images: {} ...".format(
316                    len(input_image_paths),
317                    [os.path.basename(f) for f in
    input_image_paths[:5]],
318                )
319            )
320            start = timer()
321            text_out = ""
322
323            # This is for images
324            for i, img_path in enumerate(input_image_paths):
325                print(img_path)
326                prediction, image,lat,lon= detect_object(
327                    yolo,
328                    img_path,
329                    save_img=save_img,
330                    save_img_path=FLAGS.output,
331                    postfix=FLAGS.postfix,
332                )
333                print(lat,lon)
334                y_size, x_size, _ = np.array(image).shape
335                for single_prediction in prediction:
336                    out_df = out_df.append(
```

```python
                            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),
```

```
375                     [os.path.basename(f) for f in
    input_video_paths[:5]],
376                 )
377             )
378         start = timer()
379         for i, vid_path in enumerate(input_video_paths):
380             output_path = os.path.join(
381                 FLAGS.output,
382                 os.path.basename(vid_path).replace(".",
    FLAGS.postfix + "."),
383             )
384             detect_video(yolo, vid_path,
    output_path=output_path)
385
386         end = timer()
387         print(
388             "Processed {} videos in {:.1f}sec".format(
389                 len(input_video_paths), end - start
390             )
391         )
392     # Close the current yolo session
393     yolo.close_session()
394     return render_template('prediction.html')
395
396
397 """ Running our application """
398 if __name__ == "__main__":
399     app.run(debug=True)
```

GitHub & Project Demo Link
 Github link:
https://github.com/IBM-EPBL/IBM-Project-1491-1658390989

Project Demo Link:

https://drive.google.com/file/d/1taI6JBWVhO6bxdSFp3EoauODUbw2IQSU/view?usp=share_link