SKIN DISEASE DETECTION USING MACHINE LEARNING TECHNIQUE

A Project

Submitted fulfillment for the

Award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE & ENGINEERING

Submitted by

SIDDHARTHA MAJUMDER	15500121027
RAHUL KUMAR SINGH	15500121016
PUSKAR SARKAR	15500121017
ARYAN RAJ	15500121031
DEBKUMAR MANDAL	15500121004

Under the Guidance of

Prof. Kaustav Sanyal

DURGAPUR INSTITUTE OF ADVANCED TECHNOLOGY & MANAGEMENT

Department of Computer Sciences & Engineering

Acknowledgements

We would like to express our sincere gratitude to all the contributors to this work on skin lesion detection through machine learning. First of all, we would like to thank our project mentor [Prof. Kaustav Sanyal] for his valuable guidance, support and encouragement of our project. Special thanks to our classmates and class teachers who provided constructive feedback and shared their expertise, which greatly improved our work. Finally, we are grateful to our families and friends for their unwavering support and understanding in this project. Their encouragement kept us happy and focused on our goals. Thank you all for your contributions and support.

SIDDHARTHA MAJUMDER	15500121027	Siddhartha Majumder
RAHUL KUMAR SINGH	15500121016	Rahul Kumar singh
PUSKAR SARKAR	15500121017	Puskar Sarkar
ARYAN RAJ	15500121031	Aryan raj
DEBKUMAR MANDAL	15500121004	Debkumar mandal

Date: 21/05/2024

Contents:

ABSTRACT 4
INTRODUCTION 5
BACKGROUND THEORY 6
LITERATURE SURVEY 7
METHODOLOGY 8
  DATA COLLECTION: 8
  SYSTEM REQUIREMENTS & LIBRARIES: 10
  MODULES: 13
  WORKING: 15
RESULT & DISCUSSION 16
  SAMPLE SCREEN: 17
  SAMPLE CODE: 19
CONCLUSION 29
References 30

ABSTRACT

This project is a web-based application designed to help people in remote areas diagnose skin diseases. It is especially useful for those living in rural areas without a nearby hospital. Using this website, people can quickly and accurately diagnose skin conditions, which is essential for early diagnosis and treatment. Here's how it works: The user uploads an image of the affected skin to the web page. The system then analyses the image and uses the imaging techniques to detect the skin condition. The results are displayed on the screen, along with recommendations from dermatologists and treatment precautions. The application is user-friendly and accessible to physicians and the general public. One of the main advantages is that it saves people the time and effort of going to hospitals or clinics. In addition, it helps reduce the risk of contracting those infections through contact with other patients.

Keywords: Web-based application, Remote areas, Diagnose skin diseases, Rural areas, Early diagnosis, Image analysis, Dermatologist recommendations, User-friendly;

INTRODUCTION

Skin is the largest and most sensitive part of the human body which protects our inner vital parts and organs from the outside environment, hence avoiding contact with bacteria and viruses. Skin also helps in body temperature regulation. The skin consists of cells, pigmentation, blood vessels, and other components. It is comprised of 3 main layers, namely, the epidermis, the dermis, and the hypodermis. Epidermis, being the outermost skin layer, forms a waterproof and protective sheath around the body's surface. The dermis, found beneath the epidermis, comprises of connective tissues and protects the body from stress and strain. A basement membrane tightly joins the dermis with the epidermis. The hypodermis, also called subcutaneous tissue, is not actually a part of the skin and lies below the dermis. It attaches the skin to the underlying bone and muscle and also supplies blood vessels and nerves to it. Skin diseases occur commonly among humans. They are usually caused by factors like different organism's cells, a different diet, and internal and external factors, such as the hierarchical genetic group of cells, hormones, and immune system of conditions. These factors may act together or in a sequence of skin disease. There are chronic and incurable diseases, like eczema and psoriasis, and malignant diseases like malignant melanoma. Recent researchers have found the availability of cures for these diseases if they are detected in the early stages. Atopic dermatitis, commonly called eczema, is a long-term skin disease whose common symptoms are dry and itchy skin, rashes on the face, inside the elbows, behind the knees, and on the hands and feet.

BACKGROUND THEORY

Skin diseases are a common health issue affecting millions of people worldwide. Early detection is essential for effective treatment, but access to dermatologists and primary care is often limited, especially in rural areas. Machine learning, artificial intelligence, offers a promising solution to this problem. Machine learning is about training computer programs to recognize patterns in data. For skin disease detection, these algorithms are trained with thousands of images of different skin conditions. By analyzing these images and the unique features associated with each condition, the system learns to diagnose diseases. Once trained, the machine learning system can quickly and accurately analyze new images uploaded by users. By comparing new images with learned ones, the system can identify potential skin diseases and suggest possible diagnoses. This technology enables the diagnosis of skin diseases and is more efficient. This eliminates the need to see a doctor in person, saving patients time and money. Furthermore, it can provide immediate information, which is essential for early intervention and better health. Overall, machine learning could revolutionize the way we diagnose and manage skin diseases.

LITERATURE SURVEY

This project uses machine learning to diagnose skin diseases from images, offering faster and more accurate results than traditional methods. (Inthiyaz, 2023)
This paper reviews deep learning methods for diagnosing skin diseases, highlighting their effectiveness, challenges, and future research opportunities in this field.
(Li, H. Pan, & & Zhang, 2021)
Our work enhances deep learning for image recognition, addressing challenges like small objects and limited data. Achieved high rankings in Kaggle competitions. (Yang, Zeng, Teo, Wang, & Chandrasekhar, 2018)
Deep learning technique shows promise in early melanoma diagnosis, distinguishing types accurately, aiding timely treatment, and outperforming current methods in diagnostic accuracy. (Allugunti, 2022)
This study developed systems using machine learning techniques to detect skin lesions early. Results showed high accuracy rates using artificial neural networks and convolutional neural networks. (Abunadi, 2021)
This paper introduces an automated deep learning model using Dermoscopy images for early skin disease diagnosis, achieving 90% accuracy. (Anand, Gupta, Nayak, & Koundal, 2022)
This research developed a mobile-friendly deep neural network to accurately detect herpes zoster from skin images, enhancing diagnosis efficiency. (Back, Lee, & Shin, 2021)
This paper introduces a new system combining CNN and SVM to develop a mobile app for skin disease detection, achieving high accuracy. (Elngar, Kumar, Hayat, & & Churi, 2021)

METHODOLOGY

DATA COLLECTION:

- Dataset used for this are extracted from Kaggle (SINGHAL, n.d.) towards skin disease Detection.
- It consists of 27000 images of skin disease.
- The training data consists of 10000 images and testing data consists of 2000 images.

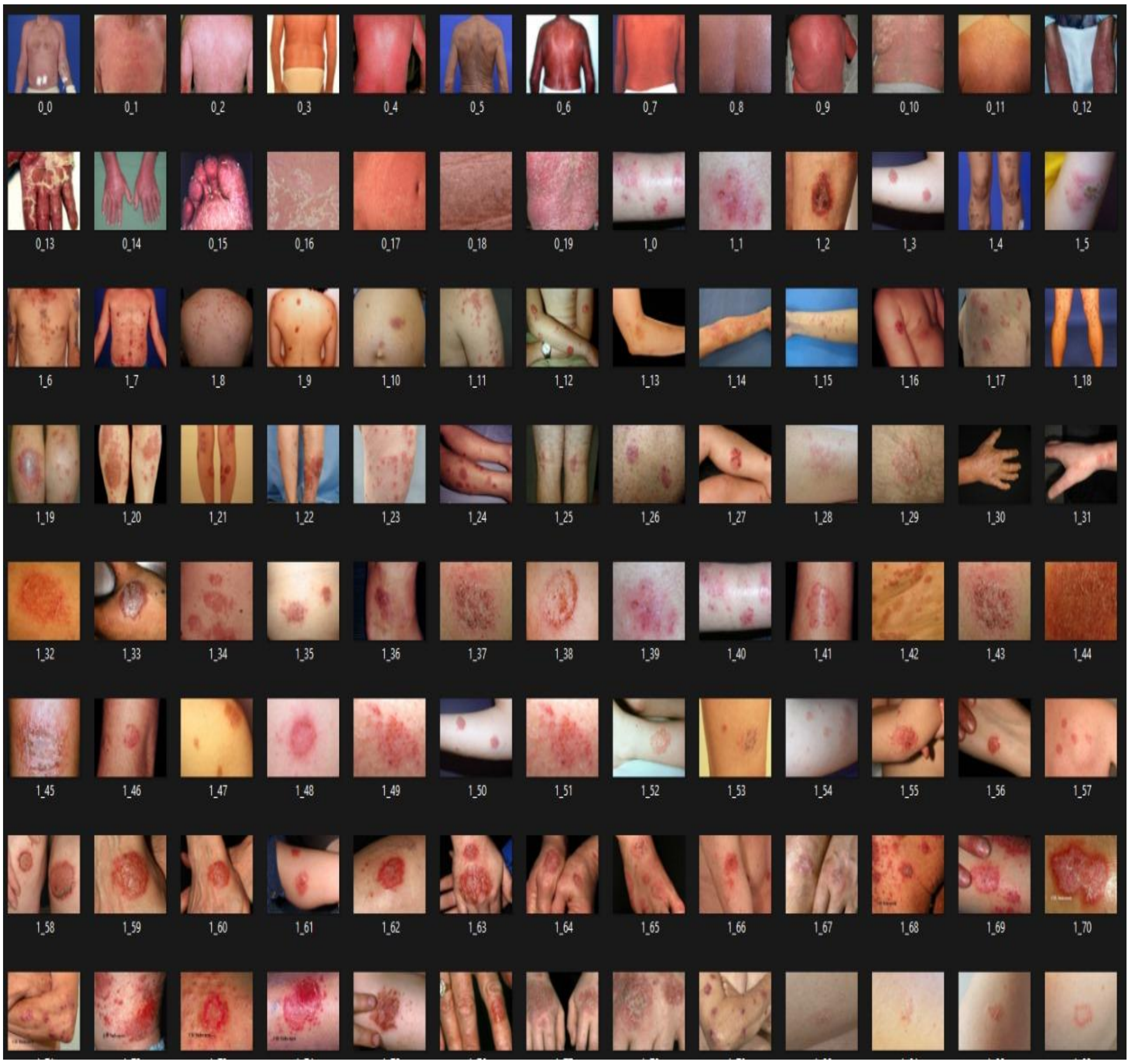


Figure 1

Fig1: IMAGES OF SKIN DISEASE DATASE

- **IMAGE PREPROCESSING:**

Image preprocessing is done by using OPEN CV and NUMPY.

- **OpenCV:**

➢ OpenCV-Python library of Python bindings in designed unravel computer vision problems.

➢ OpenCV-Python makes use Num py, by which may highly optimized library numerical operations a MATLAB-style syntax.

➢ All tin Open CV array are structures converted a and from Num py arrays.

➢ This also makes it easier to integrate other libraries are that use Num py SciPy and Matplotlib.

➢ OpenCV to be capable image analysis and processing.

- **NumPy:**

Import- numpy :as np

➢ NumPy, that stands Numerical Python, be a library consisting of multi_dimensional as array objects and set a routines for processing those arrays.

➢ Using as Num Py, mathematical and logical on operations are arrays in often performed.

➢ The array object in NumPy is named nedarray, it provides tons of supporting functions that make working with nedarray very easy.

➢ NumPy is an open-source numerical Python library. NumPy a extension o Numeric and Num array.

➢ Num py contains random number generators. NumPy may wrapper around library implemented in C.

➢ Pandas is objects reply heavily NumPy objects. Essentially, Pandas extends Numpy.

- **IMAGE SEGMENTATION & FEATURE EXTRACTION:**

➢ Image segmentation is a process of dividing image into regions or categories. In the dermoscopic images two types of fabric things first normal skin and second is lesion area so here we have done segmentation with Otsu thresholding technique. Using Texture-Based segmentation extracting the features from the image. GLCM (Gray

Level Co-occurrence Matrix) is the statistical method examining the spatial relationship between the pixel. This technique works by creating the co- occurrence matrix were to calculate the frequency of occurrence of a pixel with the grey-level value is adjacent to a pixel with grey-level value j in any given direction and selected separating distance The GLCM matrix gives four statistics Correlation, Contrast, Energy, Homogeneity. There some problem in segmentation of dermoscopic images due to the contrast of images like under segmentation and over-segmentation so we are concentrating on segmentation based on texture features.

- **<u>IMAGE CLASSIFICATION:</u>**

➢ Deep learning is one of the best techniques for image classification. Based on the texture features we are training the dataset for classification. Here first we are giving Extracted feature to the Neural network for checking performance of image classification then we are using CNN (Convolutional Neural Network) it is one of the deep learning techniques for classification, Dermoscopy images classification is done in 7 classes Melanocytic nevi', 'Melanoma', 'Benign keratosis', 'Basal cell carcinoma', 'Actinic keratoses', 'Vascular lesions', ' Dermatofibroma ' it is done by using automated extracted features by CNN images. In this step, we are passing Preprocess Images to the CNN classification.

<u>SYSTEM REQUIREMENTS & LIBRARIES:</u>

- **<u>SOFTWARE REQUIREMENTS:</u>**

Operating System: Windows, Mac OS, Linux

Application: Spyder with Python 3.0 or above

- **<u>LIBRARIES USED:</u>**

In Skin disease Detection we use some libraries in python. The list of libraries are:

➢ TensorFlow
➢ Pandas
➢ NumPy
➢ OpenCV
➢ Keras

- **TensorFlow:**

➢ TensorFlow is a free and open-source software library for machine learning.

➢ It is often across the range of tasks but features a particular specialise to training on inference of deep neural in networks.

➢ Tenso r flow may a symbolic math library supported data flow differentiable with programming.

➢ High on scalability computation across in machines and large data sets import tensorflow.

import tensorflow as tf

- **Pandas:**

➢ Pandas is a popular Python a library on data analysis.

➢ It directly in related Machine Learning. As all we have the dataset to be prepared before the training.

➢ In this case, Pandas are handy it as a developed \\to data extraction and preparation.

➢ It provides the implementation is high-level find data structures a wide variety tools for data analysis.

➢ It provides many methods for as groping, combining and filtering data.

- **NumPy:**

➢ NumPy means for Numerical Python, in a library of multidimensional array an Objects to collection of routines for processing those arrays.

➢ Using Num py mathematical and logical operations we in arrays can be performed.

➢ Num py to open-source numerical Python library.

import numpy as np

- **OpenCV:**

➢ OpenCV-Python to have library Python bindings designed and s computer vision problems.

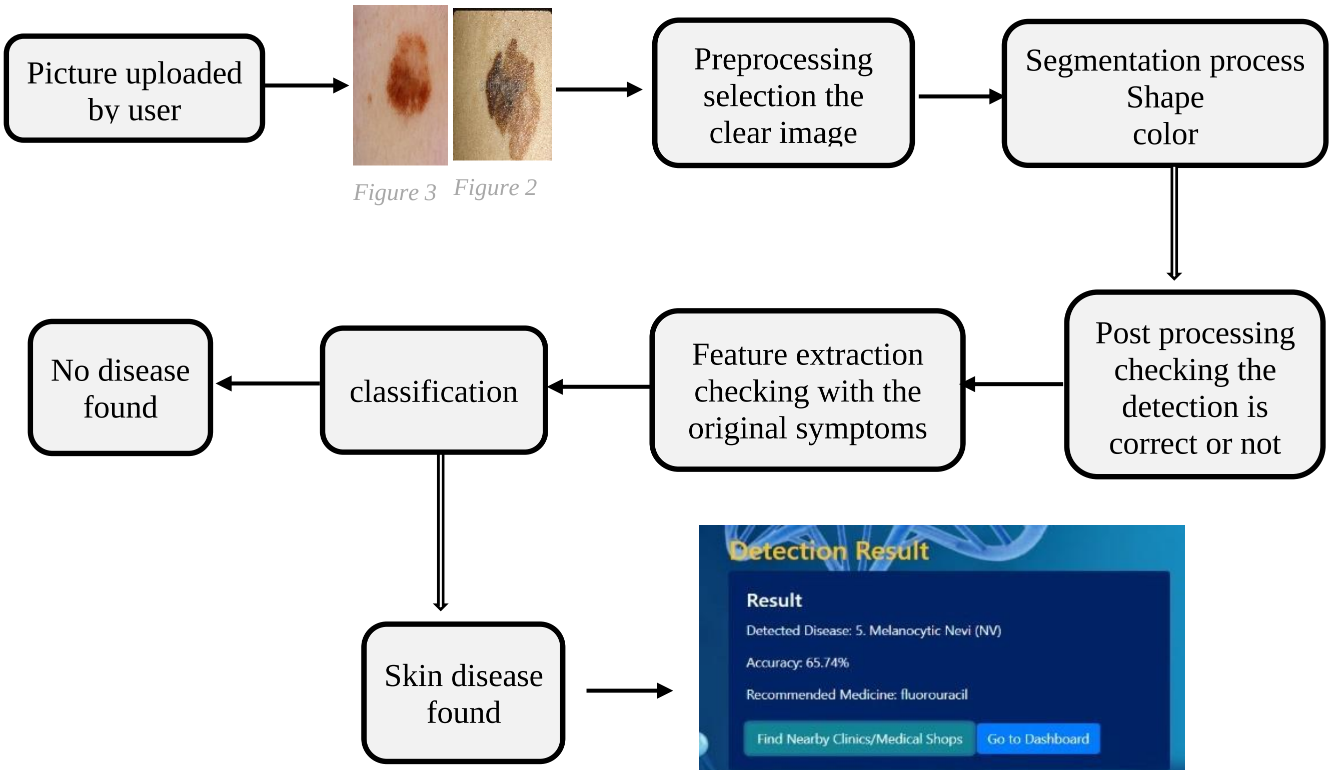
➢ Open CV is capable in image analysis and processing.

- **Keras:**

- Keras is a popular Machine Learning library for Python to do.
- It high-level neural networks (API) capable in the running top on Tensor Flow, CNTK, a Theano.
- Keras makes really to ML beginners an build a design a Neural Network.
- One in best thing about Keras that allows easy to fast prototyping.

from tensorflow import keras

- **SYSTEM ARCHITECTURE:**


Picture uploaded by user
Preprocessing selection the clear image
Segmentation process Shape color
Post processing checking the detection is correct or not
Feature extraction checking with the original symptoms
classification
No disease found
Skin disease found

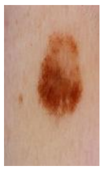
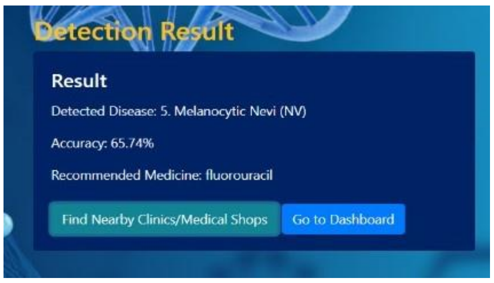


Figure 3

Figure 2


Detection Result

Result

Detected Disease: 5. Melanocytic Nevi (NV)

Accuracy: 65.74%

Recommended Medicine: fluorouracil

Figure 4

In this figure of system architecture diagram we have clearly explained the steps for detecting 10 types of skin disease. First step comes here is taking picture from the user or customer for detecting. After this next step is preprocessing which is used to convert the picture to gray scale and reshaping is also done and the next step is segmentation process in which the shape and color of the symptom or the patch will be identified. Next step is Post processing in which the detections done in the before steps are correct or not, after his feature extraction is done in which the symptoms given in the picture by user is compared with the original skin disease symptoms. Next step here comes is classification in which the website gives whether it is skin disease or not.

10 types of skin disease:

1. Eczema
2. Melanoma
3. Atopic Dermatitis
4. Basal Cell Carcinoma (BCC)
5. Melanocytic Nevi (NV)
6. Benign Keratosis-like Lesions (BKL)
7. Psoriasis pictures Lichen Planus
8. Seborrheic Keratoses
9. Tinea Ringworm Candidiasis
10. Warts Molluscum

<u>MODULES:</u>

We have 2 modules in Skin disease Detection They are:

1. Detection
2. Testing

- ### **<u>DETECTION:</u>**

➢ Detection module used them detect the image of skin cancer. In this we detect images from skin cancer by using "FEED FORWARD NEURAL NETWORK ALGORITHM".

➢ A feed forward neural network have bimologically inspired by classification which algorithm. It consist of number of simple to neuron-like as processing in units, organized layers. Every unit in a layer connected with in the units in the previous layer. This is they are called feedforward neural networks.

➢ The feed forward neural network is the in first and simplest type of artificial neural network devised. In the network, the information in one direction—forward—from a input nodes, through the hidden to nodes and to the output nodes. There non cycles in loops in the network.

➢ Two basic feed-forward neural networks (FFNNs) created using TensorFlow in deep learning library in Python.

➢ Steps required build an simple feed-forward neural network to Tenso r Flow by explaining each step details. For before actual building an neural network, some preliminary steps recommended to discussed.

The summarized steps are as follows:

1. Reading the training data (inputs and outputs)
2. Building to connect an neural networks layers
3. Building a loss function to assess the prediction error
4. Create the training loop for training network and updating parameters
5. Applying some testing data to assess the network prediction accuracy

This module briefly introduces the core concepts employed in modern convolutional neural networks, with an emphasis on methods that have been proven to be effective for tasks such as object detection and semantic segmentation. Basic network architectures, common components and helpful tools for constructing and training networks are described.

- **<u>TESTING:</u>**

➢ Testing module is used to test and predict the image of skin cancer. For testing we used " evaluation function from keras."

➢ Evaluation a is process during development to the model check whether this model fit for given problem and corresponding data.

➢ Keras provides a function, evaluate which does evaluation of the model.

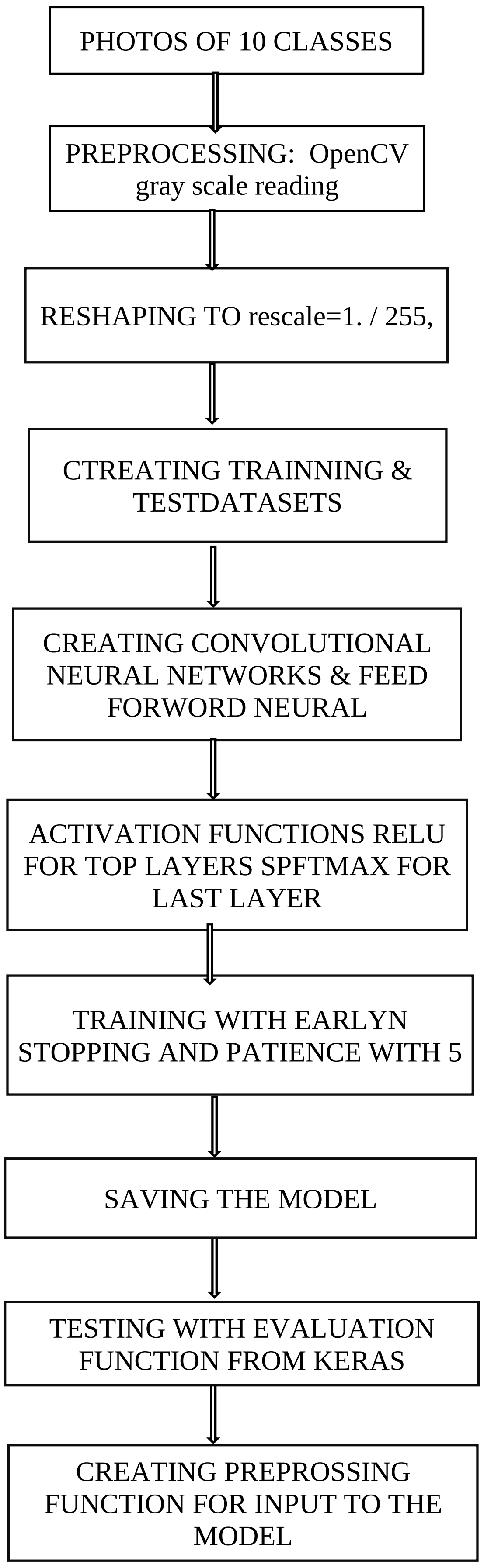
There are two main arguments,

1. Test data
2.Test data label

Keras separate a portion of your training data to validation of dataset and evaluate that performance of your model on validation dataset to each epoch. You can do this by setting the validation split argument on the fit () function to a percentage of the size of your training dataset.

WORKING:

Here comes the flowchart of this project skin disease detection:


PHOTOS OF 10 CLASSES
PREPROCESSING: OpenCV gray scale reading
RESHAPING TO rescale=1. / 255,
CTREATING TRAINNING & TESTDATASETS
CREATING CONVOLUTIONAL NEURAL NETWORKS & FEED FORWORD NEURAL
ACTIVATION FUNCTIONS RELU FOR TOP LAYERS SPFTMAX FOR LAST LAYER
TRAINING WITH EARLYN STOPPING AND PATIENCE WITH 5
SAVING THE MODEL
TESTING WITH EVALUATION FUNCTION FROM KERAS
CREATING PREPROSSING FUNCTION FOR INPUT TO THE MODEL

RESULT & DISCUSSION

Here is the output screenshot where we can know whether a person has skin disease or not. This picture is for detecting melanocytic nevi which is one of the type of skin disease.

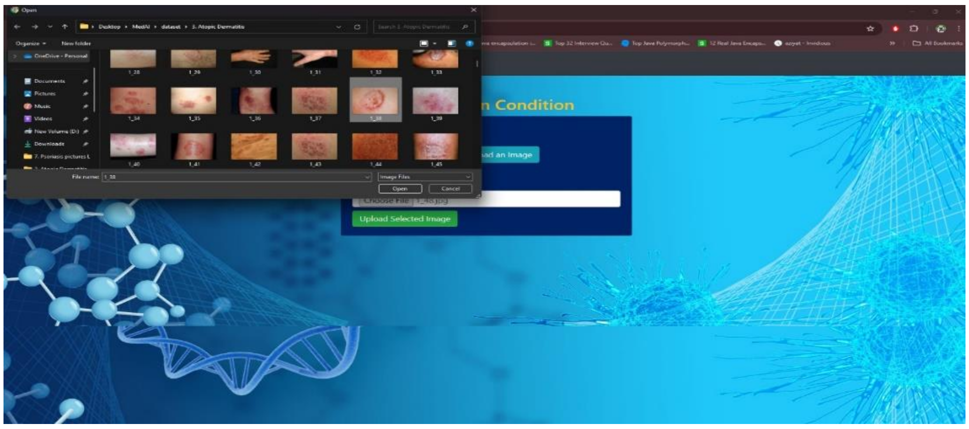
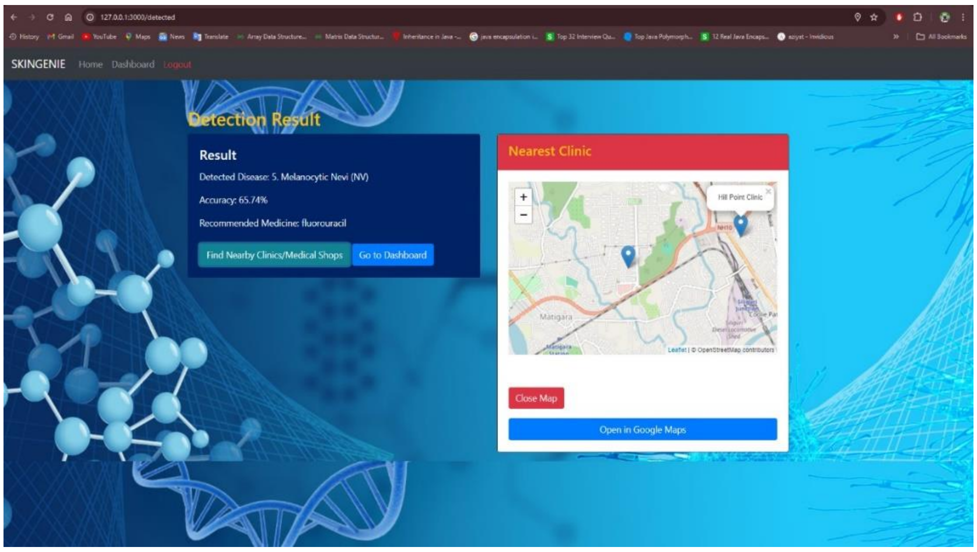


Figure 5



Detected Disease: 5. Melanocytic Nevi (NV)

Accuracy: 65.74%

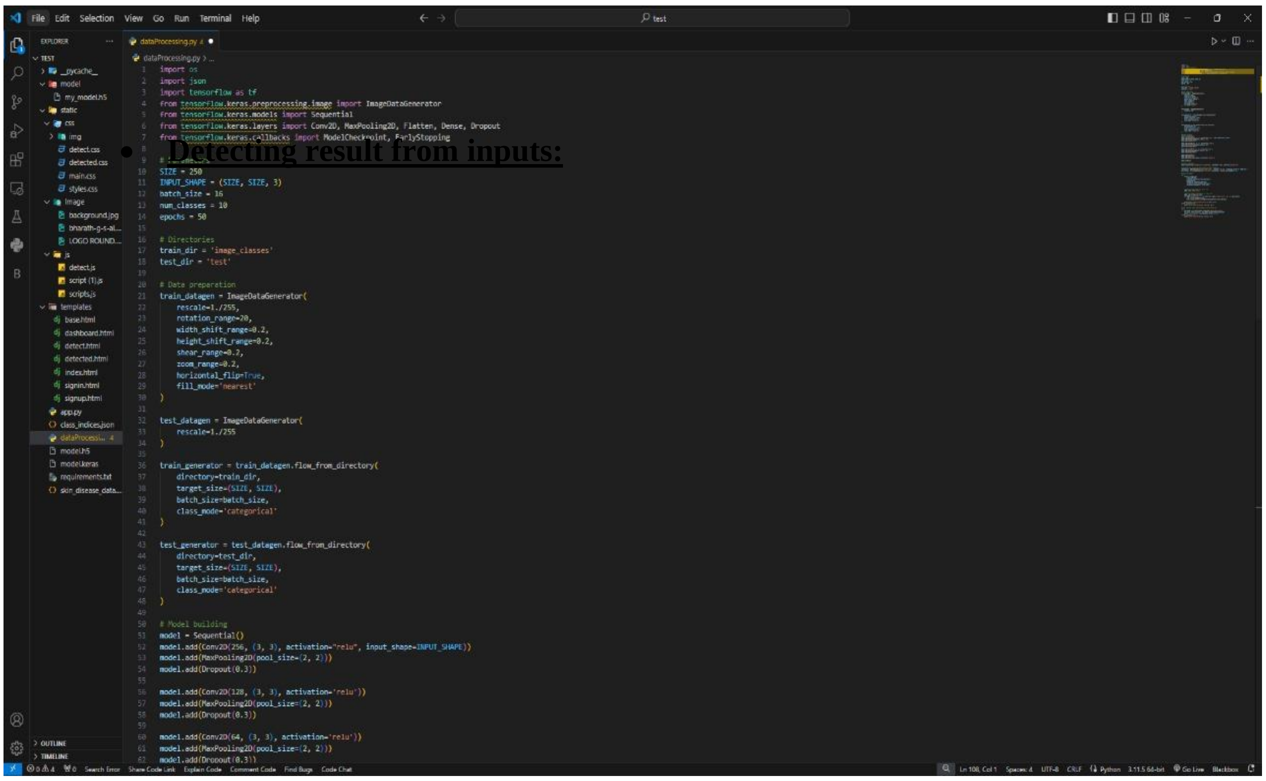
Recommended Medicine: fluorouracil

Figure 6 **We diagnosed that is melanocytic nevi.**

- This picture is for detecting ***melanocytic nevi*** which is one of the type of skin disease.

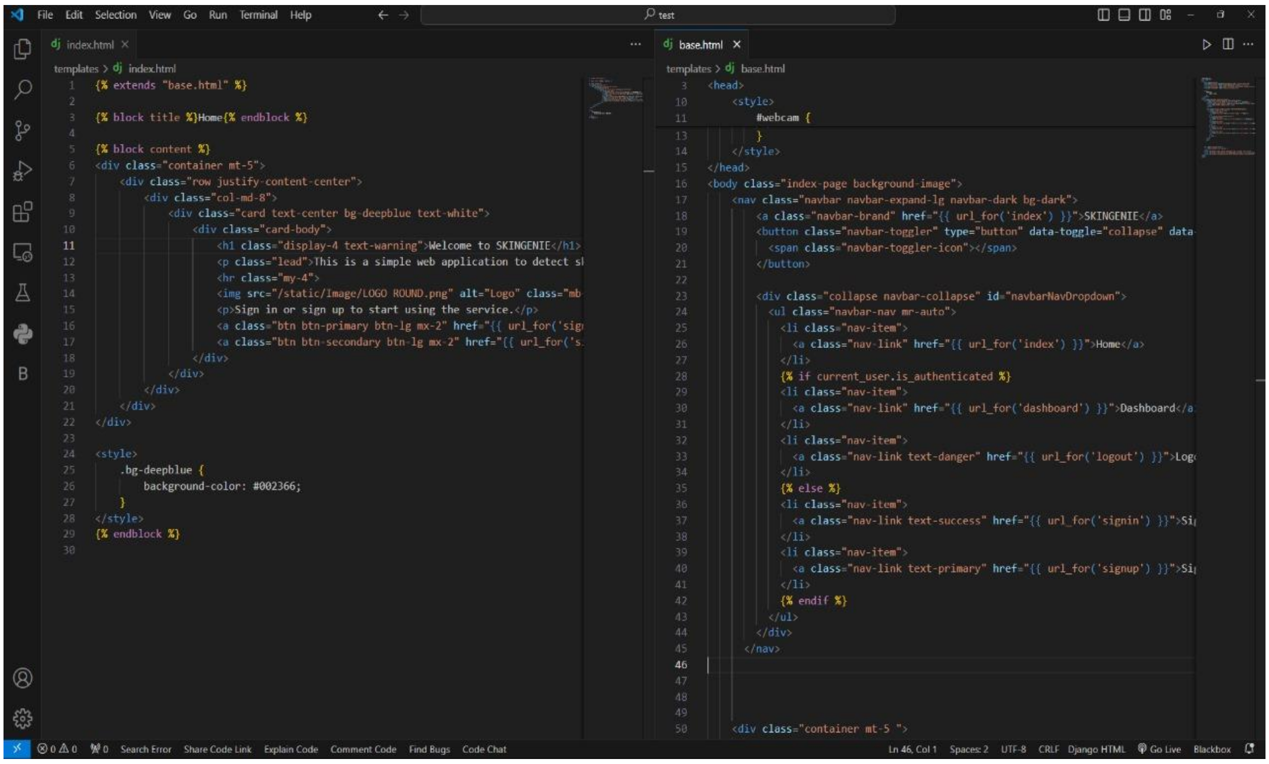
SAMPLE SCREEN:

- **Preparing train model:**



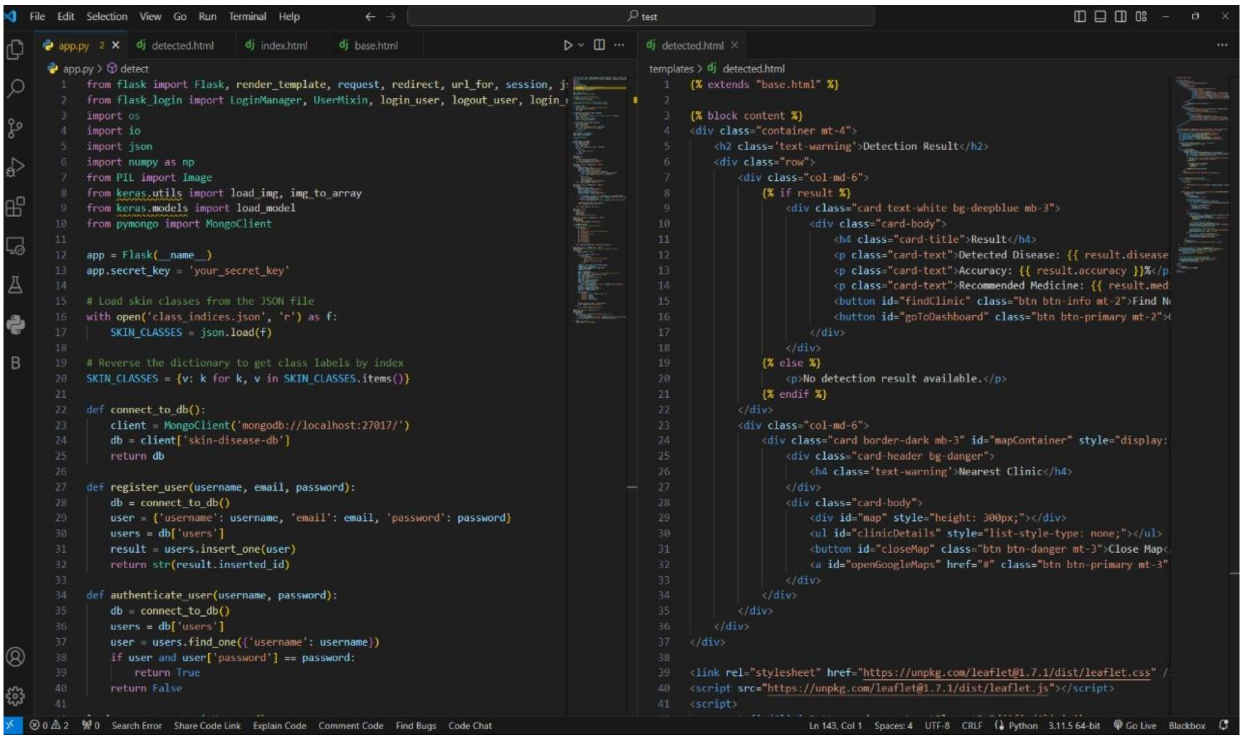
- **Detecting result from inputs:**

```
import os
import json
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

# Parameters
SIZE = 250
INPUT_SHAPE = (SIZE, SIZE, 3)
batch_size = 16
num_classes = 10
epochs = 50

# Directories
train_dir = 'image_classes'
test_dir = 'test'

# Data preperation
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

test_datagen = ImageDataGenerator(
    rescale=1./255
)

train_generator = train_datagen.flow_from_directory(
    directory=train_dir,
    target_size=(SIZE, SIZE),
    batch_size=batch_size,
    class_mode='categorical'
)

test_generator = test_datagen.flow_from_directory(
    directory=test_dir,
    target_size=(SIZE, SIZE),
    batch_size=batch_size,
    class_mode='categorical'
)

# Model building
model = Sequential()
model.add(Conv2D(256, (3, 3), activation="relu", input_shape=INPUT_SHAPE))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.3))

model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.3))

model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.3))
```



```
{% extends "base.html" %}

{% block title %}Home{% endblock %}

{% block content %}
<div class="container mt-5">
    <div class="row justify-content-center">
        <div class="col-md-8">
            <div class="card text-center bg-deepblue text-white">
                <div class="card-body">
                    <h1 class="display-4 text-warning">Welcome to SKINGENIE</h1>
                    <p class="lead">This is a simple web application to detect sl
                    <hr class="my-4">
                    <img src="/static/Image/LOGO ROUND.png" alt="Logo" class="mb
                    <p>Sign in or sign up to start using the service.</p>
                    <a class="btn btn-primary btn-lg mx-2" href="{{ url_for('sig
                    <a class="btn btn-secondary btn-lg mx-2" href="{{ url_for('s
                </div>
            </div>
        </div>
    </div>
</div>

<style>
    .bg-deepblue {
        background-color: #002366;
    }
</style>
{% endblock %}
```

```
<head>
    <style>
        #webcam {
        }
    </style>
</head>
<body class="index-page background-image">
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <a class="navbar-brand" href="{{ url_for('index') }}">SKINGENIE</a>
        <button class="navbar-toggler" type="button" data-toggle="collapse" data
          <span class="navbar-toggler-icon"></span>
        </button>

        <div class="collapse navbar-collapse" id="navbarNavDropdown">
          <ul class="navbar-nav mr-auto">
            <li class="nav-item">
              <a class="nav-link" href="{{ url_for('index') }}">Home</a>
            </li>
            {% if current_user.is_authenticated %}
            <li class="nav-item">
              <a class="nav-link" href="{{ url_for('dashboard') }}">Dashboard</a
            </li>
            <li class="nav-item">
              <a class="nav-link text-danger" href="{{ url_for('logout') }}">Log
            </li>
            {% else %}
            <li class="nav-item">
              <a class="nav-link text-success" href="{{ url_for('signin') }}">Si
            </li>
            <li class="nav-item">
              <a class="nav-link text-primary" href="{{ url_for('signup') }}">Si
            </li>
            {% endif %}
          </ul>
        </div>
      </nav>




    <div class="container mt-5 ">
```



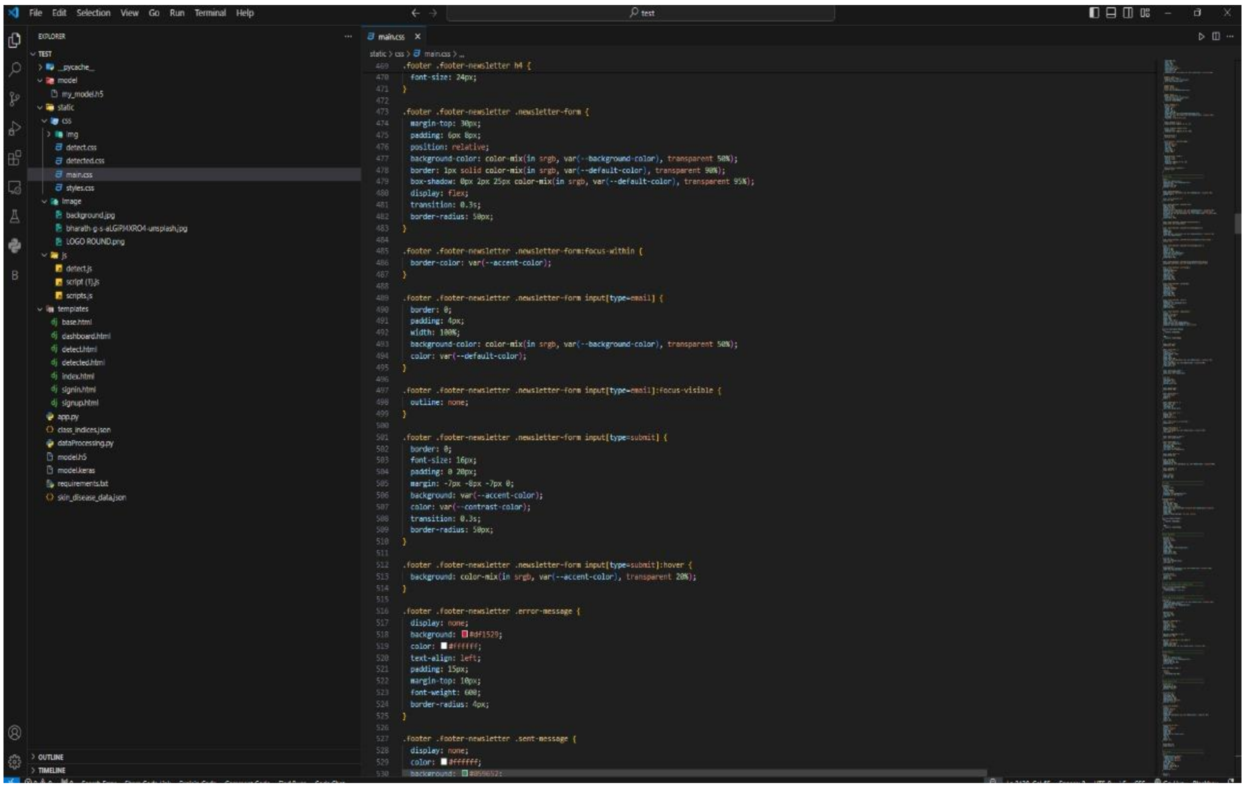
app.py > detect

```python
from flask import Flask, render_template, request, redirect, url_for, session, j
from flask_login import LoginManager, UserMixin, login_user, logout_user, login_
import os
import io
import json
import numpy as np
from PIL import Image
from keras.utils import load_img, img_to_array
from keras.models import load_model
from pymongo import MongoClient

app = Flask(__name__)
app.secret_key = 'your_secret_key'

# Load skin classes from the JSON file
with open('class_indices.json', 'r') as f:
    SKIN_CLASSES = json.load(f)

# Reverse the dictionary to get class labels by index
SKIN_CLASSES = {v: k for k, v in SKIN_CLASSES.items()}

def connect_to_db():
    client = MongoClient('mongodb://localhost:27017/')
    db = client['skin-disease-db']
    return db

def register_user(username, email, password):
    db = connect_to_db()
    user = {'username': username, 'email': email, 'password': password}
    users = db['users']
    result = users.insert_one(user)
    return str(result.inserted_id)

def authenticate_user(username, password):
    db = connect_to_db()
    users = db['users']
    user = users.find_one({'username': username})
    if user and user['password'] == password:
        return True
    return False
```

templates > detected.html

```html
{% extends "base.html" %}

{% block content %}
<div class="container mt-4">
    <h2 class='text-warning'>Detection Result</h2>
    <div class="row">
        <div class="col-md-6">
            {% if result %}
                <div class="card text-white bg-deepblue mb-3">
                    <div class="card-body">
                        <h4 class="card-title">Result</h4>
                        <p class="card-text">Detected Disease: {{ result.disease
                        <p class="card-text">Accuracy: {{ result.accuracy }}%</p
                        <p class="card-text">Recommended Medicine: {{ result.med
                        <button id="findClinic" class="btn btn-info mt-2">Find N
                        <button id="goToDashboard" class="btn btn-primary mt-2">
                    </div>
                </div>
            {% else %}
                <p>No detection result available.</p>
            {% endif %}
        </div>
        <div class="col-md-6">
            <div class="card border-dark mb-3" id="mapContainer" style="display:
                <div class="card-header bg-danger">
                    <h4 class='text-warning'>Nearest Clinic</h4>
                </div>
                <div class="card-body">
                    <div id="map" style="height: 300px;"></div>
                    <ul id="clinicDetails" style="list-style-type: none;"></ul>
                    <button id="closeMap" class="btn btn-danger mt-3">Close Map<
                    <a id="openGoogleMaps" href="#" class="btn btn-primary mt-3"
                </div>
            </div>
        </div>
    </div>
</div>

<link rel="stylesheet" href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css" /
<script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"></script>
<script>
```

Ln 143, Col 1   Spaces: 4   UTF-8   CRLF   Python   3.11.5 64-bit



```
TEST
> __pycache__
v model
    my_model.h5
v static
  v css
    > img
      detect.css
      detected.css
      main.css
      styles.css
  v image
      background.jpg
      bharath-g-s-aLGiPJ4XRO4-unsplash.jpg
      LOGO ROUND.png
  v js
      detect.js
      script (1).js
      scripts.js
v templates
    base.html
    dashboard.html
    detect.html
    detected.html
    index.html
    signin.html
    signup.html
  app.py
  class_indices.json
  dataProcessing.py
  model.h5
  model.keras
  requirements.txt
  skin_disease_data.json
```

static > css > main.css > ...

```css
.footer .footer-newsletter h4 {
  font-size: 24px;
}

.footer .footer-newsletter .newsletter-form {
  margin-top: 30px;
  padding: 6px 8px;
  position: relative;
  background-color: color-mix(in srgb, var(--background-color), transparent 50%);
  border: 1px solid color-mix(in srgb, var(--default-color), transparent 90%);
  box-shadow: 0px 2px 25px color-mix(in srgb, var(--default-color), transparent 95%);
  display: flex;
  transition: 0.3s;
  border-radius: 50px;
}

.footer .footer-newsletter .newsletter-form:focus-within {
  border-color: var(--accent-color);
}

.footer .footer-newsletter .newsletter-form input[type=email] {
  border: 0;
  padding: 4px;
  width: 100%;
  background-color: color-mix(in srgb, var(--background-color), transparent 50%);
  color: var(--default-color);
}

.footer .footer-newsletter .newsletter-form input[type=email]:focus-visible {
  outline: none;
}

.footer .footer-newsletter .newsletter-form input[type=submit] {
  border: 0;
  font-size: 16px;
  padding: 0 20px;
  margin: -7px -8px -7px 0;
  background: var(--accent-color);
  color: var(--contrast-color);
  transition: 0.3s;
  border-radius: 50px;
}

.footer .footer-newsletter .newsletter-form input[type=submit]:hover {
  background: color-mix(in srgb, var(--accent-color), transparent 20%);
}

.footer .footer-newsletter .error-message {
  display: none;
  background: #df1529;
  color: #ffffff;
  text-align: left;
  padding: 15px;
  margin-top: 10px;
  font-weight: 600;
  border-radius: 4px;
}

.footer .footer-newsletter .sent-message {
  display: none;
  color: #ffffff;
  background: #059652;
```

SAMPLE CODE:

CODE OF DATA PROCESSING:

```
import os

import json

import tensorflow as tf

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

# Parameters

SIZE = 250

INPUT_SHAPE = (SIZE, SIZE, 3)

batch_size = 16

num_classes = 10

epochs = 50

# Directories

train_dir = 'image_classes'

test_dir = 'test'

# Data preparation

train_datagen = ImageDataGenerator(

    rescale=1./255,

    rotation_range=20,

    width_shift_range=0.2,

    height_shift_range=0.2,

    shear_range=0.2,

    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
test_datagen = ImageDataGenerator(
    rescale=1./255
)
train_generator = train_datagen.flow_from_directory(
    directory=train_dir,
    target_size=(SIZE, SIZE),
    batch_size=batch_size,
    class_mode='categorical'
)
test_generator = test_datagen.flow_from_directory(
    directory=test_dir,
    target_size=(SIZE, SIZE),
    batch_size=batch_size,
    class_mode='categorical'
)
# Model building
model = Sequential()
model.add(Conv2D(256, (3, 3), activation="relu", input_shape=INPUT_SHAPE))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.3))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.3))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.3))
model.add(Flatten())
model.add(Dense(32))
model.add(Dense(num_classes, activation='softmax'))
model.summary()
# Compile the model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
# Callbacks for saving the best model and early stopping
checkpoint = ModelCheckpoint('best_model.keras', monitor='val_loss',
save_best_only=True, mode='min')
early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
# Train the model
try:
    history = model.fit(
        train_generator,
        steps_per_epoch=len(train_generator),
        epochs=epochs,
        validation_data=test_generator,
        validation_steps=len(test_generator),
        callbacks=[checkpoint, early_stop]
    )
    # Save the final model to .keras file
    model.save('model.keras')
```

```
        # Save the model architecture to .json file
          model_json = model.to_json()
          with open('model.json', 'w') as json_file, open('classes.json', 'w') as json_classes:
            json_file.write(model_json)
            json_classes.write(json.dumps(train_generator.class_indices))
          print("Model saved to model.keras and model.json")
        except Exception as e:
          print(f"Error occurred during training: {e}")
        # Load the best model and evaluate it on the test set
        try:
          best_model = tf.keras.models.load_model('best_model.keras')
          test_loss, test_accuracy = best_model.evaluate(test_generator)
          print(f"Test Accuracy: {test_accuracy * 100:.2f}%")
        except Exception as e:
  print(f"Error occurred during testing: {e}")
```

SAMPLE CODE FOR BACKEND:

```
        from flask import Flask, render_template, request, redirect, url_for, session, jsonify
        from flask_login import LoginManager, UserMixin, login_user, logout_user,
login_required, current_user
        import os
        import io
        import json
        import numpy as np
        from PIL import Image
        from keras.utils import load_img, img_to_array
        from keras.models import load_model
        from pymongo import MongoClient

        app = Flask(_name_)

        app.secret_key = 'your_secret_key'

        # Load skin classes from the JSON file

        with open('class_indices.json', 'r') as f:

           SKIN_CLASSES = json.load(f)

        # Reverse the dictionary to get class labels by index

        SKIN_CLASSES = {v: k for k, v in SKIN_CLASSES.items()}

        def connect_to_db():

           client = MongoClient('mongodb://localhost:27017/')

           db = client['skin-disease-db']

           return db

        def register_user(username, email, password):

           db = connect_to_db()

           user = {'username': username, 'email': email, 'password': password}

           users = db['users']

           result = users.insert_one(user)

           return str(result.inserted_id)

        def authenticate_user(username, password):

           db = connect_to_db()

           users = db['users']

           user = users.find_one({'username': username})

           if user and user['password'] == password:

              return True

           return False

        login_manager = LoginManager()
```

```
login_manager.init_app(app)
class User(UserMixin):
    pass
@login_manager.user_loader
def user_loader(username):
    db = connect_to_db()
    users = db['users']
    user = users.find_one({'username': username})
    if user:
        u = User()
        u.id = user['username']
        return u
    return None
@app.route('/')
def index():
    if current_user.is_authenticated:
        return redirect(url_for('dashboard'))
    return render_template('index.html')
@app.route('/signin', methods=['GET', 'POST'])
def signin():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        if not authenticate_user(username, password):
            error = 'Invalid username or password. Please try again.'
            return render_template('signin.html', error=error)
        else:
            user = User()
            user.id = username
            login_user(user)
            return redirect(url_for('dashboard'))
    return render_template('signin.html')
@app.route('/signup', methods=['GET', 'POST'])
def signup():
    if request.method == 'POST':
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']
        confirm_password = request.form['confirm']
        if not username or not email or not password or not confirm_password:
            return render_template('error.html', message='Please fill in all fields.')
        if password != confirm_password:
            return render_template('error.html', message='Passwords do not match.')
        register_user(username, email, password)
        return redirect(url_for('signin'))
    return render_template('signup.html')
@app.route('/logout')
def logout():
    session.clear()
    logout_user()
    return redirect(url_for('index'))
@app.route('/dashboard', methods=['GET', 'POST'])
```

```
@login_required
def dashboard():
    return render_template('dashboard.html')
def findMedicine(pred):
    # Add your medicine mapping logic here
    medicine_map = {
      0: "fluorouracil",
       1: "Aldara",
       2: "Tetracycline, Minocycline and Doxycycline",
       3: "fluorouracil",
       4: "fluorouracil (5-FU)",
       5: "fluorouracil",
       6: "fluorouracil",
       7: "fluorouracil",
       8: "fluorouracil",
       9: "fluorouracil",
       10: "fluorouracil"
    }
    return medicine_map.get(pred, "No medicine available")
@app.route('/detect', methods=['GET', 'POST'])
@login_required  # Ensure the user is logged in to access this route
def detect():
    if request.method == 'POST':
        try:
            file = request.files['file']
        except KeyError:
```

```
    return jsonify({
        'error': 'No file part in the request',
        'code': 'FILE',
        'message': 'file is not valid'
    }), 400

imagePil = Image.open(io.BytesIO(file.read()))
imageBytesIO = io.BytesIO()
imagePil.save(imageBytesIO, format='JPEG')
imageBytesIO.seek(0)
path = imageBytesIO
model = load_model('model.keras')  # Load model from .keras file
img = load_img(path, target_size=(224, 224))
img = img_to_array(img)
img = img.reshape((1, 224, 224, 3))
img = img / 255
prediction = model.predict(img)
pred = np.argmax(prediction)
if pred not in SKIN_CLASSES:
    return render_template('error.html', message='Invalid prediction')
disease = SKIN_CLASSES[pred]
accuracy = prediction[0][pred]
accuracy = round(accuracy * 100, 2)
medicine = findMedicine(pred)
json_response = {
    "detected": False if pred == 2 else True,
                "disease": disease,

                "accuracy": accuracy,

                "medicine": medicine,

                "img_path": file.filename,

            }

          session['detection_result'] = json_response

            return redirect(url_for('detected'))

        else:

            return render_template('detect.html')

    @app.route('/detected', methods=['GET'])

    @login_required

    def detected():

        detection_result = session.get('detection_result', None)

        if not detection_result:

            return render_template('error.html', message='No detection result available.')

        return render_template('detected.html', result=detection_result)

    if _name_ == "_main_":

app.run(debug=True, port=3000)
```

CONCLUSION

Development and implementation of an advanced skin disease detection system for health care. Access to these search tools, whether through web-based applications. You can be the first from the comfort of your own home, assessing their skin condition, health promotion and. It reduces the burden on the healthcare system. As technology advances into the future, these systems have the ability to provide better and faster detection, Improve overall skin health. We are working on the plan, as far as I know we are already making future improvements, this website is not quite done, but I'll finish it in a few days soon.

References

[1] Abunadi, I. &. (2021). Deep learning and machine learning techniques of diagnosis dermoscopy images for early detection of skin diseases. . *Electronics,* .

[2] Allugunti, V. R. (2022). A machine learning model for skin disease classification using convolution neural network. *International Journal of Computing, Programming and Database Management,* , 141-147.

[3] Anand, V., Gupta, S., Nayak, S. R., & Koundal, D. P. (2022). An automated deep learning models for classification of skin disease using Dermoscopy images:. *A comprehensive study. Multimedia Tools and Applications,*.

[4] Back, S., Lee, S., & Shin, S. Y. (2021). Robust skin disease classification by distilling deep neural network ensemble for the mobile diagnosis of herpes zoster. *IEEE Access*.

[5] Elngar, A. A., Kumar, R., Hayat, A., & & Churi, P. (2021). Intelligent system for skin disease prediction using machine learning. *IOP Publishing.*

[6] Inthiyaz, S. A. (2023). Skin disease detection using deep learning. *Advances in Engineering Software,* , 175.

[7] Li, H. Pan, Y. Z., & & Zhang, L. (2021). Skin disease diagnosis with deep learning:. *A review. Neurocomputing,*, 364-393.

[8] SINGHAL, S. (n.d.). Retrieved from kaggle: https://www.kaggle.com/code/smitisinghal/skin-disease-classification

[9] Yang, X., Zeng, Z., Teo, S. G., Wang, L., & Chandrasekhar, V. &. (2018). Deep learning for practical image recognition: Case study on kaggle competitions. . *In Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*.