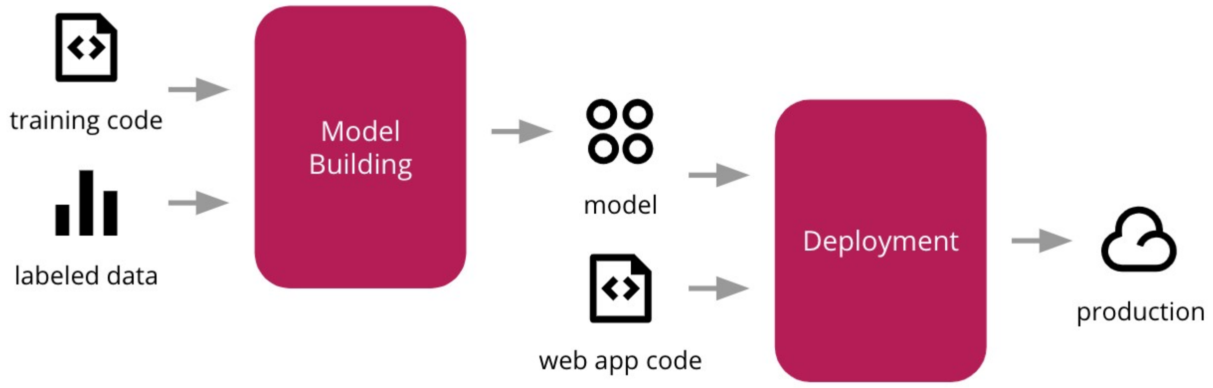
Nail Disease Prediction

Introduction:

In the healthcare domain, many diseases can be predicted by observing the color and shape of human nails. A white spot here, a rosy stain there, or some winkler projection may be an indication of disease in the body. Problems in the liver, lungs, and heart can show up in your nails. Doctors observe the nails of patients to get assistance in disease identification. Usually, pink nails indicate a healthy human. Healthy nails are smooth and consistent in color. Anything else affecting the growth and appearance of the fingernails or toenails may indicate an abnormality. A person's nails can say a lot about their health condition. The need of such systems to analyze nails for disease prediction is because the human eye is having subjectivity about colors, having limitation of resolution and small amount of color change in a few pixels on the nail not being highlighted to human eyes which may lead to wrong result, whereas computer recognize small color changes on nails.

To overcome the above problem we are building a model which is used for the prevention and early detection of Nail Disease, Basically nail disease diagnosis depends on the different characteristics like color, shape, texture etc. Here the person can capture the images of the nail and then the image will be sent to the trained model. The model analyzes the image and detects whether the person is having nail disease or not and its type.

Technical Architecture:



Prerequisites:

To complete this project, you must require the following software's, concepts, and packages

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS.Conda is an open-source, cross-platform, package management system. Anaconda comes with so very nice tools like JupyterLab, Jupyter Notebook, QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code. For this project, we will be

using Jupyter notebook and Spyder
To install Anaconda navigator and to know how to use Jupyter Notebook & Spyder using Anaconda watch the video
Link: Click here to watch the video

1. To build Machine learning models you must require the following packages

● **Numpy**:

o It is an open-source numerical Python library. It contains a multidimensional array and matrix data structures and can be used to perform mathematical operations ●

Scikit-learn:

o It is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbors, and it also supports Python numerical and scientific libraries like NumPy and SciPy

● **Streamlit:**

Web framework used for building Web applications

● **Python packages:**

o open anaconda prompt as administrator
o Type “pip install numpy” and click enter.
o Type “pip install pandas” and click enter.
o Type “pip install scikit-learn” and click enter.
o Type “pip install tensorflow==2.3.2” and click enter.
o Type “pip install keras==2.3.1” and click enter.
o Type “pip install streamlit” and click enter.

● **Deep Learning Concepts**

o **VGG16:** VGG16 is a transfer learning method. A pre-trained model trained on 1000 classes of images. VGG basic

o **Streamlit:** Streamlit is an open-source app framework for Machine Learning and Data Science teams for building web applications

If you are using Pycharm IDE, you can install the packages through the command prompt and follow the same syntax as above.

Project Objectives:

By the end of this project you will:

● Know fundamental concepts and techniques of Convolutional Neural Network.
● Gain a broad understanding of image data.
● Know how to pre-process/clean the data using different data preprocessing techniques.
● know how to build a web application using the Streamlit framework.

Project Flow:

● The user interacts with the UI (User Interface) to choose the image. ● The chosen image analyzed by the model which is integrated with streamlit application. ● VGG16

Model analyzes the image, then prediction is showcased on the Streamlit UI.

To accomplish this, we have to complete all the activities and tasks listed below

- o Data Collection.
  - o Create Train and Test Folders.
- o Model Building
  - o Importing the Model Building Libraries
  - o Loading the model
  - o Adding Flatten Layers
  - o Adding Output Layer
  - o Creating a Model object:
  - o Configure the Learning Process
  - o Import the ImageDataGenerator library
  - o Configure ImageDataGenerator class
  - o Apply ImageDataGenerator functionality to Train set and Test set
- o Training
  - o Train the Model
  - o Save the Model
- o Testing
  - o Test the model
- o Application Building
  - o Create a Streamlit UI
  - o Build Python Code
  - o Run the application
  - o Final Output

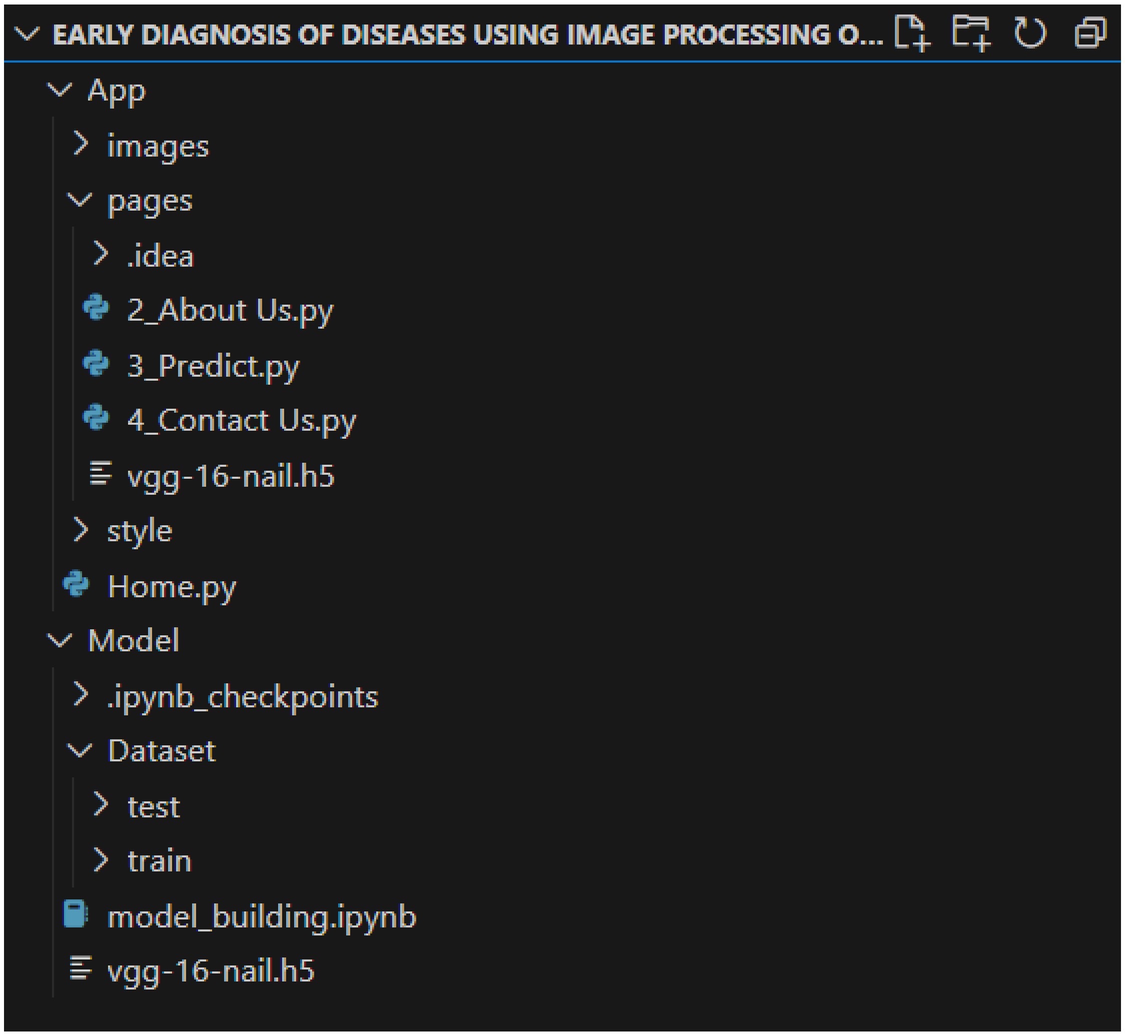
Project Structure:

Create a Project folder which contains files as shown below

● The Dataset folder contains the training and testing images for training our model. ● We are building a Streamlit Application that needs multiple pages stored in the **pages** folder and a python script **Home.py** for server side scripting

● we need the model which is saved and the saved model in this content is a **vgg-16-nail.h5**

● pages folder contains 2_About Us.py, 3_Predict.py, 4_Contact Us.py, and our model vgg-16-nail.h5. ● An IPYNB file is a notebook document created by Jupyter Notebook. **model_building.ipynb**



```
EARLY DIAGNOSIS OF DISEASES USING IMAGE PROCESSING O...
  App
    images
    pages
      .idea
      2_About Us.py
      3_Predict.py
      4_Contact Us.py
      vgg-16-nail.h5
    style
    Home.py
  Model
    .ipynb_checkpoints
    Dataset
      test
      train
    model_building.ipynb
    vgg-16-nail.h5
```

Milestone 1: Data Collection

You can download the dataset used in this project using the below link
Dataset:
https://drive.google.com/drive/folders/1AXTYsbiarS1TCAgfj0mancTSrJYYMWMs?usp=sharing

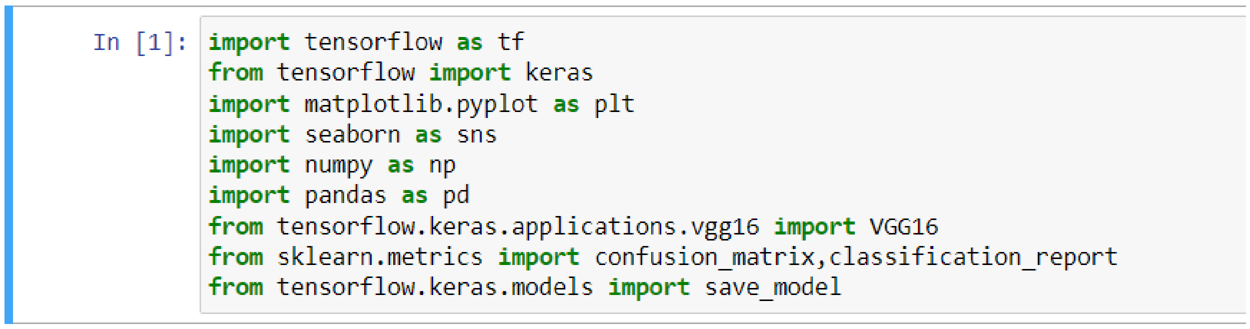
Note: For better accuracy, train on more images, which can be downloaded from google and place it in respective folders.

Milestone 2: Model Building

Now it's time to Build input and output layers for VGG16 model
Hidden layers freeze because they have trained sequence, so changing the input and output layers.

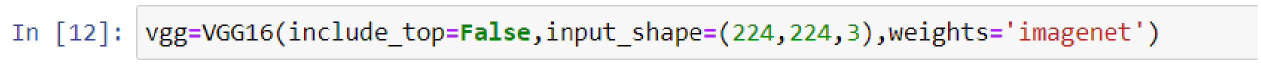
Activity 1: Importing the Model Building Libraries

Importing the necessary libraries



```
In [1]: import tensorflow as tf
        from tensorflow import keras
        import matplotlib.pyplot as plt
        import seaborn as sns
        import numpy as np
        import pandas as pd
        from tensorflow.keras.applications.vgg16 import VGG16
        from sklearn.metrics import confusion_matrix,classification_report
        from tensorflow.keras.models import save_model
```

Activity 2: Loading the model

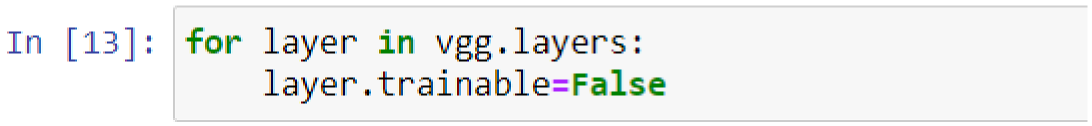


```
In [12]: vgg=VGG16(include_top=False,input_shape=(224,224,3),weights='imagenet')
```

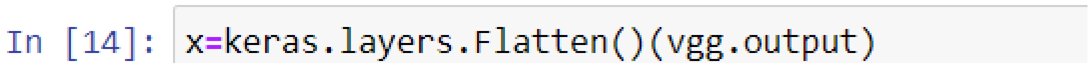
The vgg16 model need to be loaded and we are storing that into a variable called vgg

Activity 3: Adding Flatten Layers

For VGG16 model, we need to keep the Hidden layer training as false, because it has trained weights



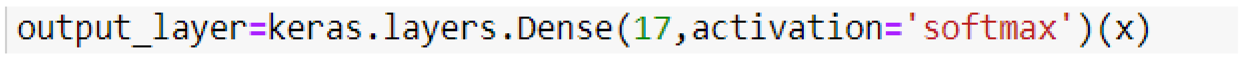
```
In [13]: for layer in vgg.layers:
             layer.trainable=False
```



```
In [14]: x=keras.layers.Flatten()(vgg.output)
```

Activity 4: Adding Output Layer

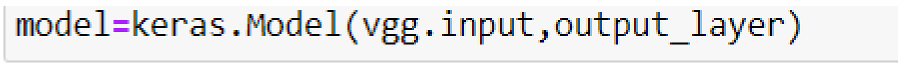
Our dataset has 17 classes, so the output layer need to be changed as per the dataset



```
output_layer=keras.layers.Dense(17,activation='softmax')(x)
```

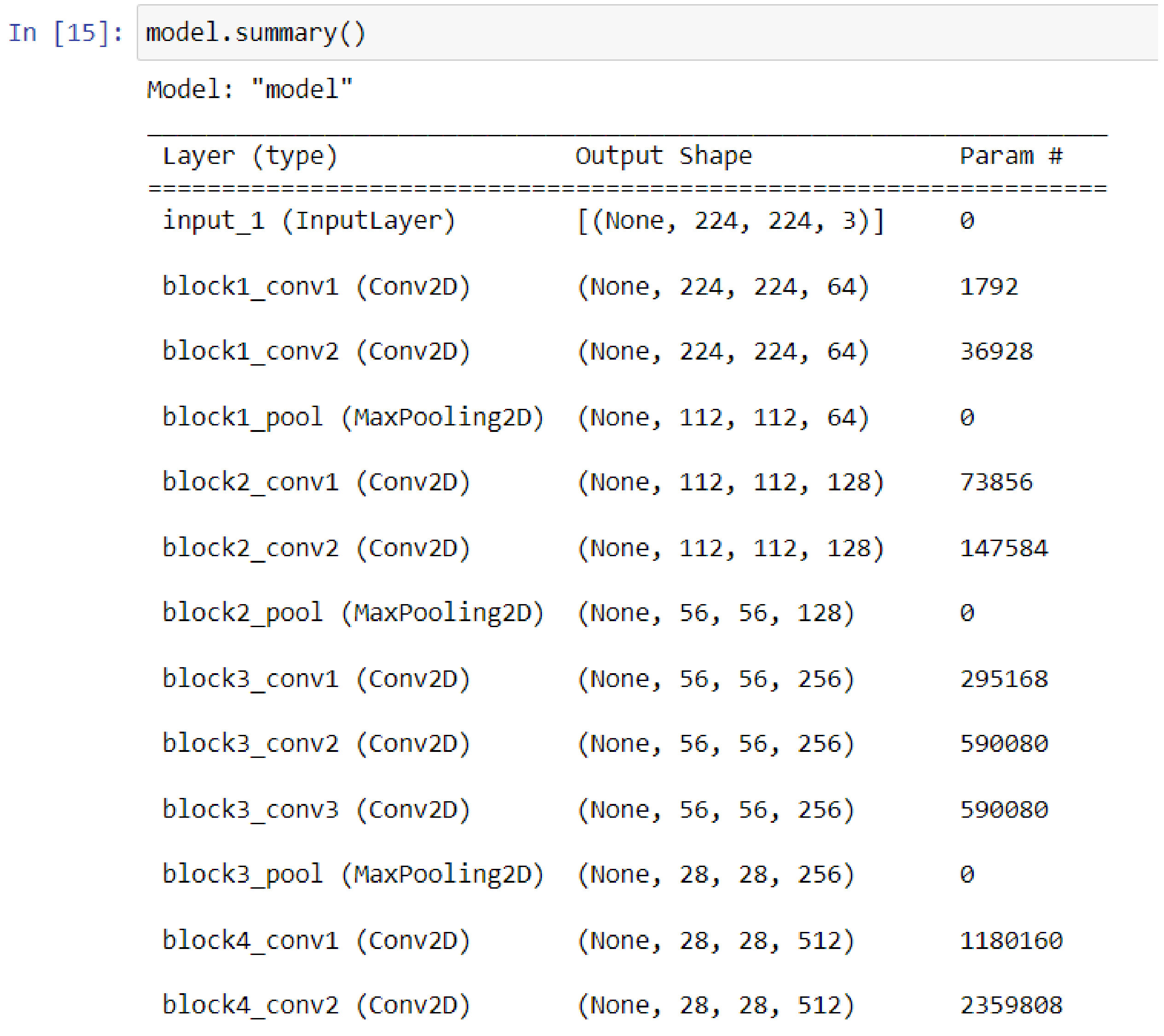
17 indicates no of classes, softmax is the activation function we use for categorical output Adding fully connected layer

Activity 5: Creating a Model object:



```
model=keras.Model(vgg.input,output_layer)
```

We have created inputs and outputs in the previous steps and we are creating a model and fitting to the vgg16 model, so that it will take inputs as per the given and displays the given no of classes



```
In [15]: model.summary()

         Model: "model"
         _________________________________________________________________
          Layer (type)                Output Shape              Param #
         =================================================================
          input_1 (InputLayer)        [(None, 224, 224, 3)]     0

          block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

          block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

          block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

          block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

          block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

          block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

          block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

          block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

          block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

          block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

          block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

          block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808
```

```
 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten (Flatten)           (None, 25088)             0

 dense (Dense)               (None, 17)                426513

=================================================================
Total params: 15,141,201
Trainable params: 426,513
Non-trainable params: 14,714,688
_________________________________________________________________
```

The vgg model has a total of 15,141,201 parameters, out of which 426,513 are trainable and the remaining are freeze.

Activity 6: Configure the Learning Process

- The compilation is the final step in creating a model. Once the compilation is done, we can move on to the training phase. The loss function is used to find errors or deviations in the learning process. Keras requires a loss function during the model compilation process.
- Optimization is an important process that optimizes the input weights by comparing the prediction and the loss function. Here we are using Adam optimizer
- Metrics are used to evaluate the performance of your model. It is similar to the loss function, but not used in the training process

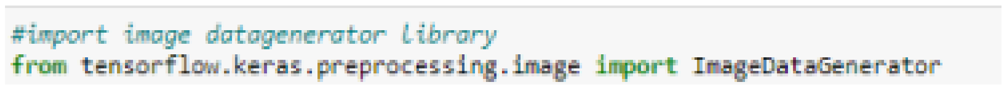
```
In [16]: model.compile(optimizer=keras.optimizers.Adam(learning_rate=0.001),
                       loss='categorical_crossentropy',metrics=['accuracy'])
```

Activity 7: Import the ImageDataGenerator library

In this we will be improving the image data that suppresses unwilling distortions or enhances some image features important for further processing, although perform some geometric transformations of images like rotation, scaling, translation, etc.

Image data augmentation is a technique that can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset.

The Keras deep learning neural network library provides the capability to fit models using image data augmentation via the ImageDataGenerator class.
Let us import the ImageDataGenerator class from tensorflow Keras

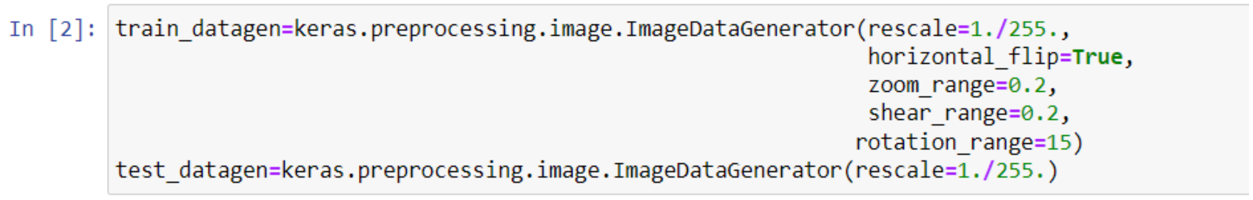


```
#import image datagenerator library
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

Activity 8: Configure ImageDataGenerator class

ImageDataGenerator class is instantiated and the configuration for the types of data augmentation
There are five main types of data augmentation techniques for image data; specifically:
Image shifts via the width_shift_range and height_shift_range arguments. The image flips via the horizontal_flip and vertical_flip arguments.
Image rotations via the rotation_range argument
Image brightness via the brightness_range argument.
Image zoom via the zoom_range argument.

An instance of the ImageDataGenerator class can be constructed for training and test.



```
In [2]: train_datagen=keras.preprocessing.image.ImageDataGenerator(rescale=1./255.,
                                                                 horizontal_flip=True,
                                                                 zoom_range=0.2,
                                                                 shear_range=0.2,
                                                                rotation_range=15)
        test_datagen=keras.preprocessing.image.ImageDataGenerator(rescale=1./255.)
```

Activity 9: Apply ImageDataGenerator functionality to Trainset and Testset

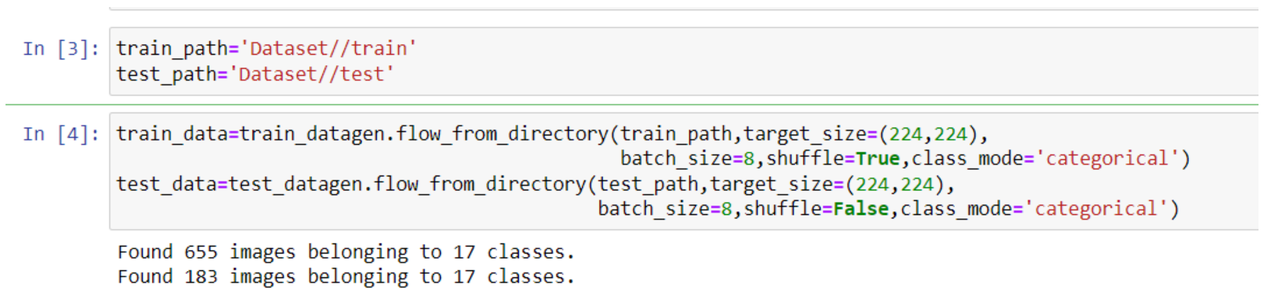
Let us apply ImageDataGenerator functionality to Trainset and Testset by using the following code. For Training set using flow_from_directory function.

This function will return batches of images from the subdirectories
Arguments:

- Directory: Directory where the data is located. If labels are "inferred", it should contain subdirectories, each containing images for a class. Otherwise, the directory structure is ignored.

- batch_size: Size of the batches of data which is 10.
- target_size: Size to resize images after they are read from disk.
- class_mode:
  - ‘int': means that the labels are encoded as integers (e.g. for sparse_categorical_crossentropy loss).
  - 'categorical' means that the labels are encoded as a categorical vector (e.g. for categorical_crossentropy loss).
  - 'binary' means that the labels (there can be only 2) are encoded as float32 scalars with values 0 or 1 (e.g. for binary_crossentropy).
  - None (no labels).

Loading our data and performing Data Augmentation

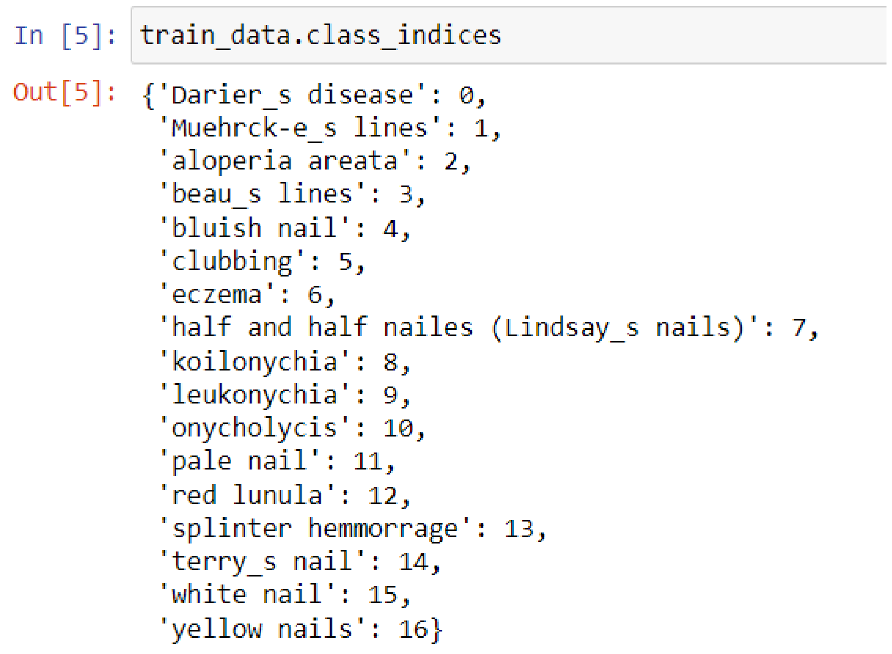


```
In [3]: train_path='Dataset//train'
        test_path='Dataset//test'
```

```
In [4]: train_data=train_datagen.flow_from_directory(train_path,target_size=(224,224),
                                                     batch_size=8,shuffle=True,class_mode='categorical')
        test_data=test_datagen.flow_from_directory(test_path,target_size=(224,224),
                                                   batch_size=8,shuffle=False,class_mode='categorical')

        Found 655 images belonging to 17 classes.
        Found 183 images belonging to 17 classes.
```

We notice that 655 images belong to 17 classes for training and 183 images belonging to 17 classes for testing purposes.

List of classes we have



```
In [5]: train_data.class_indices

Out[5]: {'Darier_s disease': 0,
         'Muehrck-e_s lines': 1,
         'aloperia areata': 2,
         'beau_s lines': 3,
         'bluish nail': 4,
         'clubbing': 5,
         'eczema': 6,
         'half and half nailes (Lindsay_s nails)': 7,
         'koilonychia': 8,
         'leukonychia': 9,
         'onycholycis': 10,
         'pale nail': 11,
         'red lunula': 12,
         'splinter hemmorrage': 13,
         'terry_s nail': 14,
         'white nail': 15,
         'yellow nails': 16}
```

Milestone 3: Training

Activity 1: Train the Model

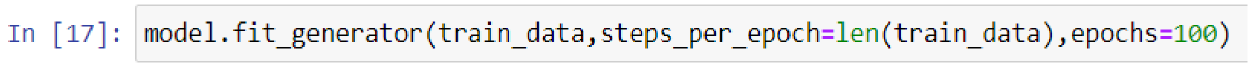
Now, let us train our model with our image dataset. The model is trained for 30 epochs and after every epoch, the current model state is saved if the model has the least loss encountered till that time. We can see that the training loss decreases in almost every epoch till 30 epochs and probably there is further scope to improve the model.

Fit_generator functions used to train a deep learning neural network

Arguments:

- steps_per_epoch: it specifies the total number of steps taken from the generator as soon as one epoch is finished and the next epoch has started. We can calculate the value of steps_per_epoch as the total number of samples in your dataset divided by the batch size.

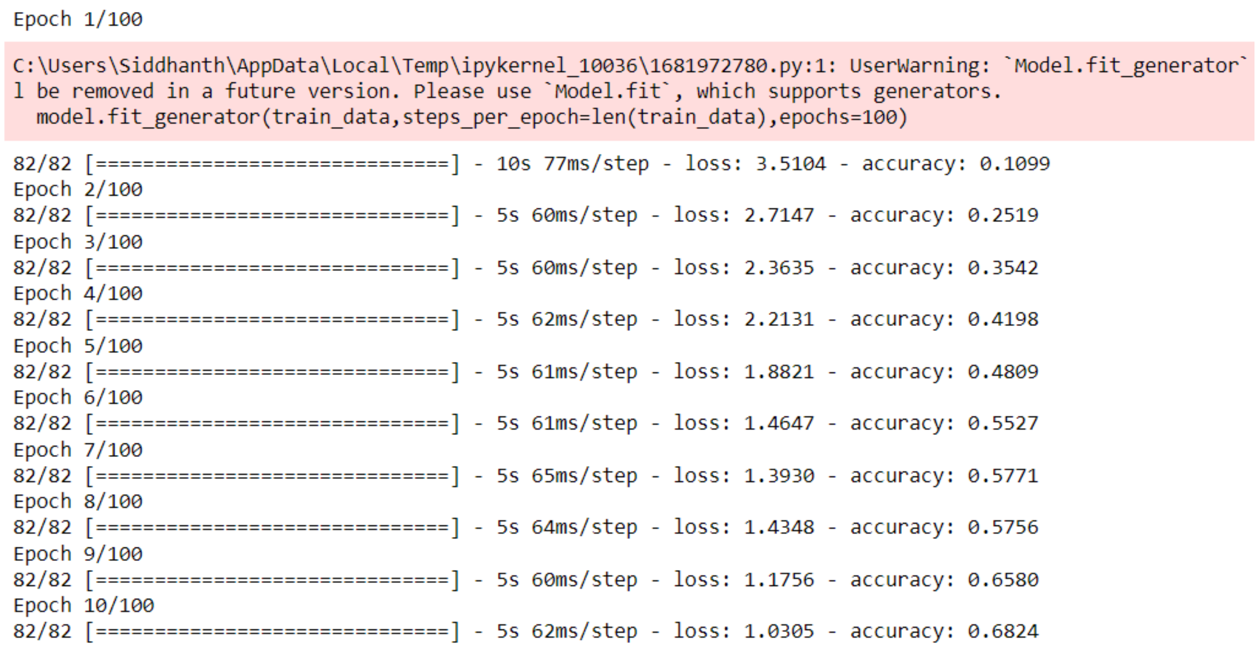
- Epochs: an integer and number of epochs we want to train our model for.



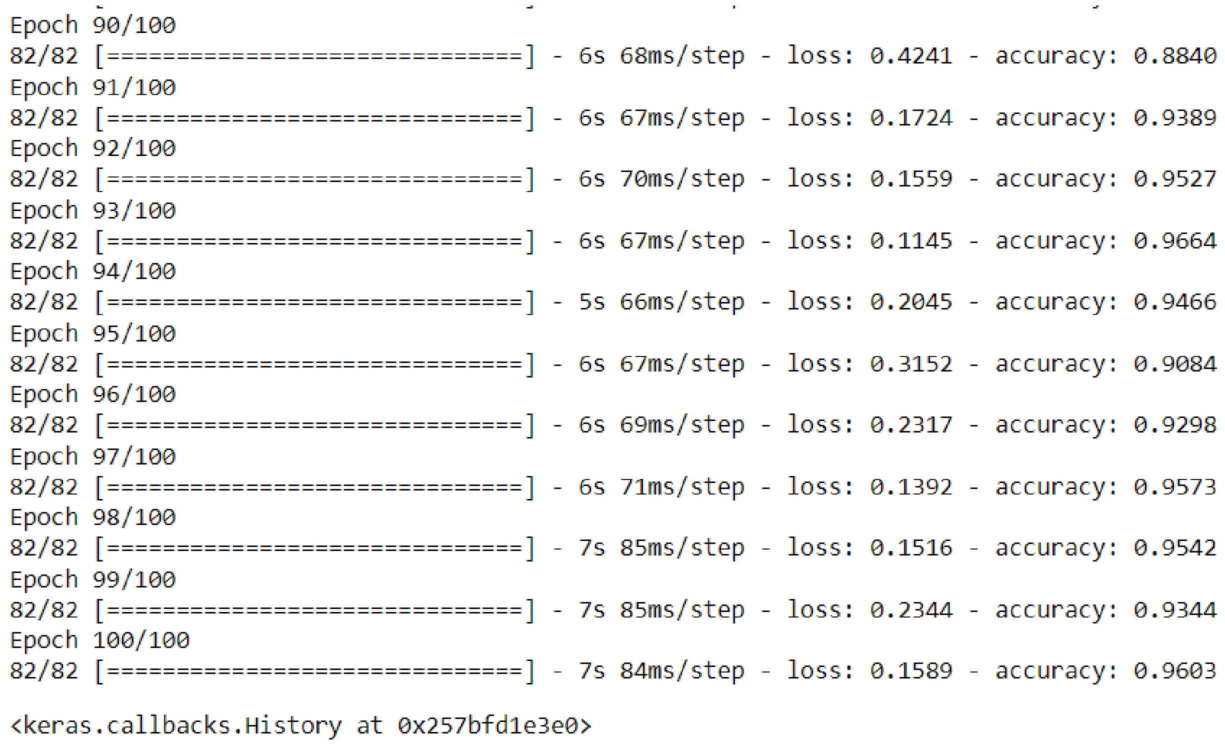
```
In [17]: model.fit_generator(train_data,steps_per_epoch=len(train_data),epochs=100)
```

Note: for model, we need to fit the data

Accuracy of the model after 100 epochs.

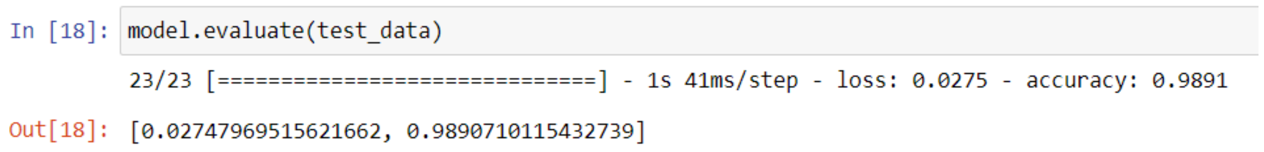


```
Epoch 1/100
C:\Users\Siddhanth\AppData\Local\Temp\ipykernel_10036\1681972780.py:1: UserWarning: `Model.fit_generator`
l be removed in a future version. Please use `Model.fit`, which supports generators.
  model.fit_generator(train_data,steps_per_epoch=len(train_data),epochs=100)
82/82 [==============================] - 10s 77ms/step - loss: 3.5104 - accuracy: 0.1099
Epoch 2/100
82/82 [==============================] - 5s 60ms/step - loss: 2.7147 - accuracy: 0.2519
Epoch 3/100
82/82 [==============================] - 5s 60ms/step - loss: 2.3635 - accuracy: 0.3542
Epoch 4/100
82/82 [==============================] - 5s 62ms/step - loss: 2.2131 - accuracy: 0.4198
Epoch 5/100
82/82 [==============================] - 5s 61ms/step - loss: 1.8821 - accuracy: 0.4809
Epoch 6/100
82/82 [==============================] - 5s 61ms/step - loss: 1.4647 - accuracy: 0.5527
Epoch 7/100
82/82 [==============================] - 5s 65ms/step - loss: 1.3930 - accuracy: 0.5771
Epoch 8/100
82/82 [==============================] - 5s 64ms/step - loss: 1.4348 - accuracy: 0.5756
Epoch 9/100
82/82 [==============================] - 5s 60ms/step - loss: 1.1756 - accuracy: 0.6580
Epoch 10/100
82/82 [==============================] - 5s 62ms/step - loss: 1.0305 - accuracy: 0.6824
```

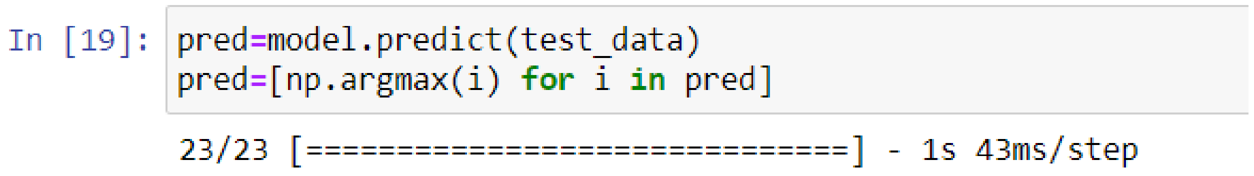


```
Epoch 90/100
82/82 [==============================] - 6s 68ms/step - loss: 0.4241 - accuracy: 0.8840
Epoch 91/100
82/82 [==============================] - 6s 67ms/step - loss: 0.1724 - accuracy: 0.9389
Epoch 92/100
82/82 [==============================] - 6s 70ms/step - loss: 0.1559 - accuracy: 0.9527
Epoch 93/100
82/82 [==============================] - 6s 67ms/step - loss: 0.1145 - accuracy: 0.9664
Epoch 94/100
82/82 [==============================] - 5s 66ms/step - loss: 0.2045 - accuracy: 0.9466
Epoch 95/100
82/82 [==============================] - 6s 67ms/step - loss: 0.3152 - accuracy: 0.9084
Epoch 96/100
82/82 [==============================] - 6s 69ms/step - loss: 0.2317 - accuracy: 0.9298
Epoch 97/100
82/82 [==============================] - 6s 71ms/step - loss: 0.1392 - accuracy: 0.9573
Epoch 98/100
82/82 [==============================] - 7s 85ms/step - loss: 0.1516 - accuracy: 0.9542
Epoch 99/100
82/82 [==============================] - 7s 85ms/step - loss: 0.2344 - accuracy: 0.9344
Epoch 100/100
82/82 [==============================] - 7s 84ms/step - loss: 0.1589 - accuracy: 0.9603

<keras.callbacks.History at 0x257bfd1e3e0>
```

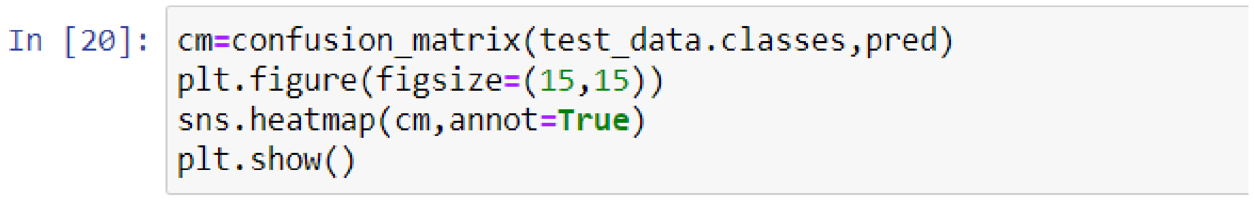
Testing model performance on Testset



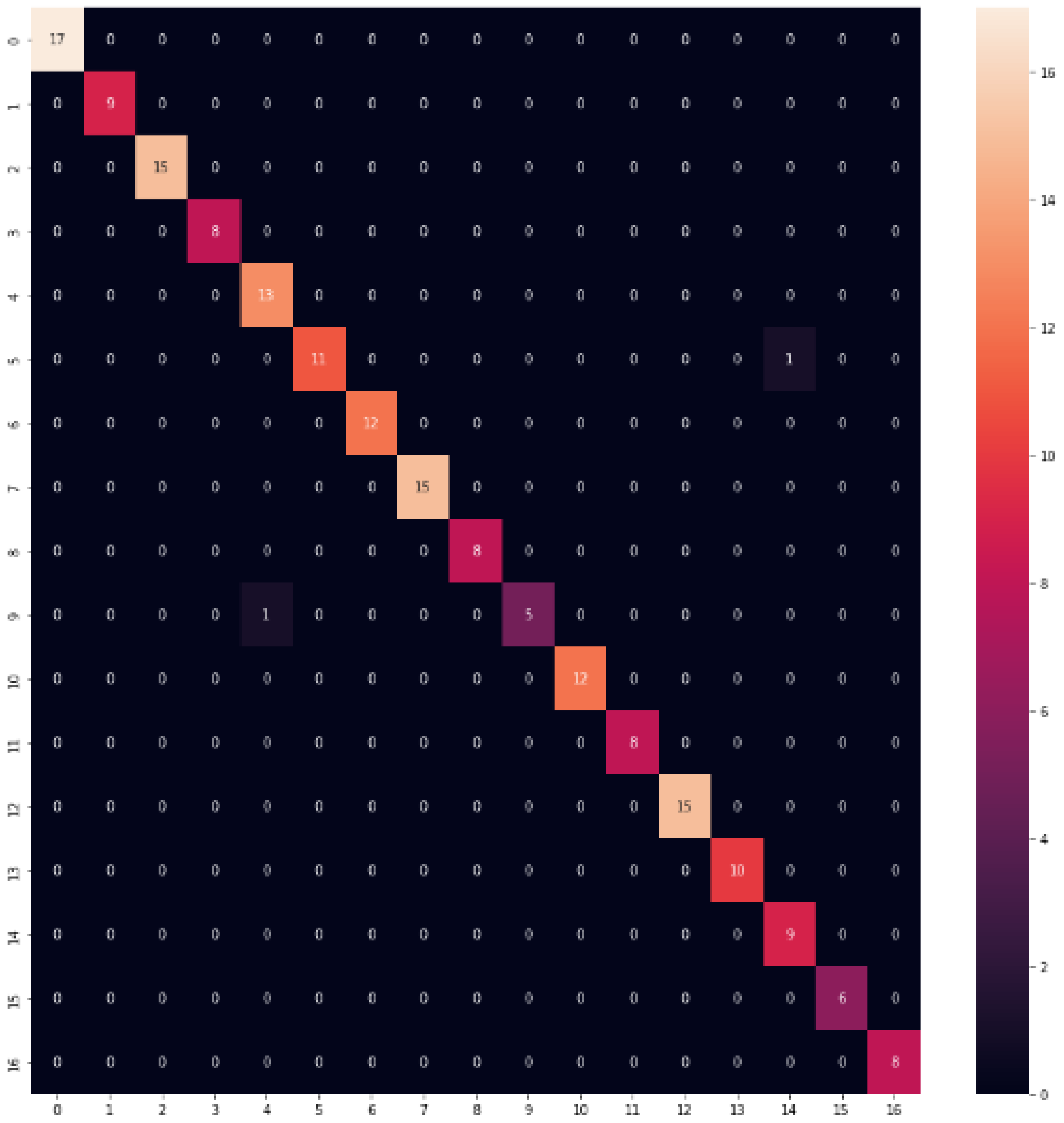
```
In [18]: model.evaluate(test_data)

         23/23 [==============================] - 1s 41ms/step - loss: 0.0275 - accuracy: 0.9891

Out[18]: [0.02747969515621662, 0.9890710115432739]
```



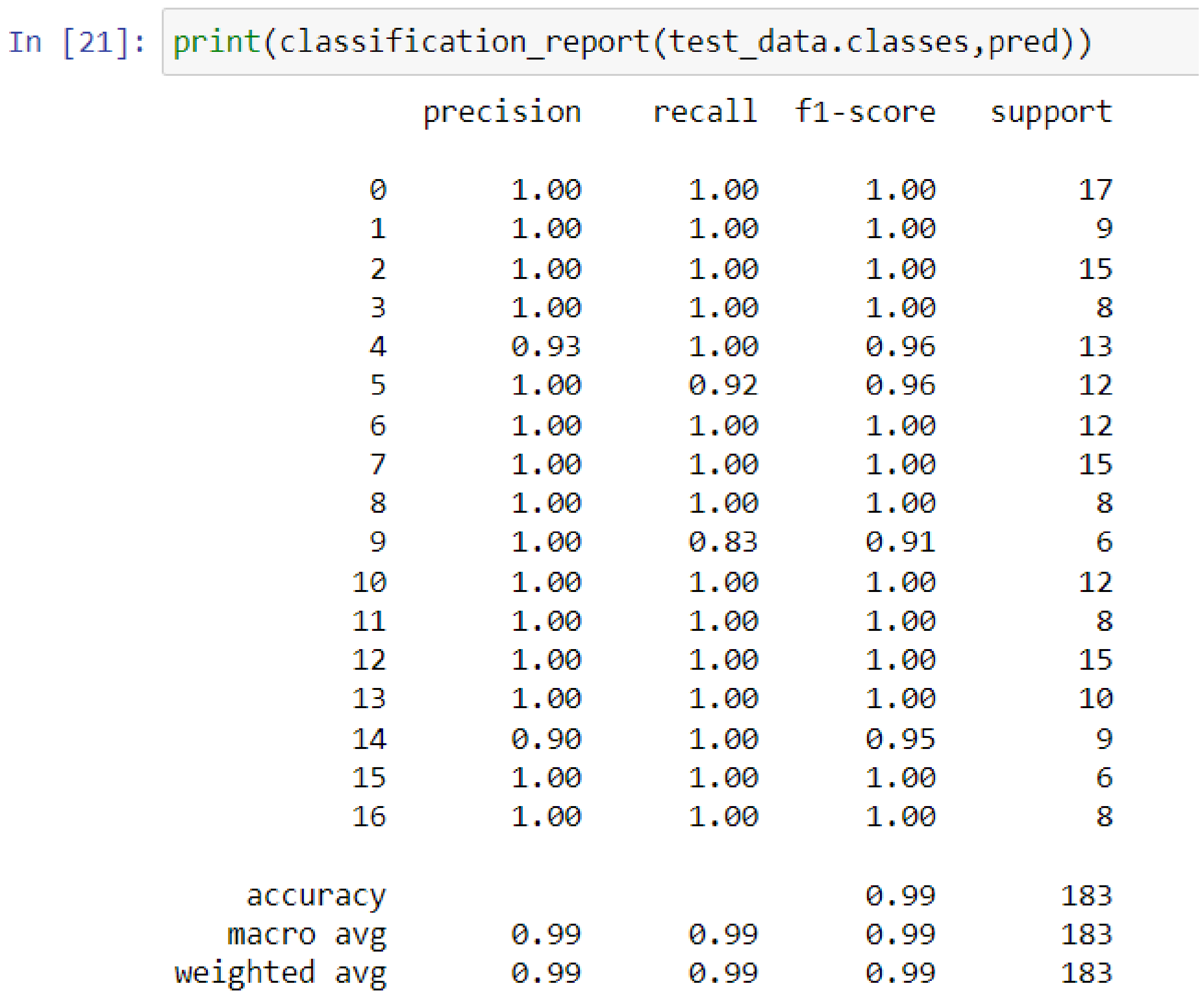
```
In [19]: pred=model.predict(test_data)
         pred=[np.argmax(i) for i in pred]

         23/23 [==============================] - 1s 43ms/step
```



```
In [20]: cm=confusion_matrix(test_data.classes,pred)
         plt.figure(figsize=(15,15))
         sns.heatmap(cm,annot=True)
         plt.show()
```



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	11	0	0	0	0	0	0	0	0	1	0	0
6	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0
9	0	0	0	0	1	0	0	0	0	5	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8

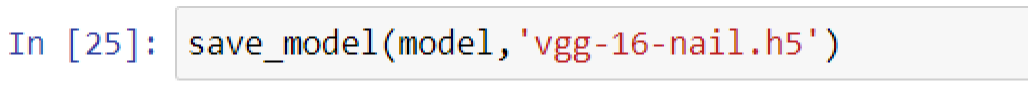


```
In [21]: print(classification_report(test_data.classes,pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        17
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        15
           3       1.00      1.00      1.00         8
           4       0.93      1.00      0.96        13
           5       1.00      0.92      0.96        12
           6       1.00      1.00      1.00        12
           7       1.00      1.00      1.00        15
           8       1.00      1.00      1.00         8
           9       1.00      0.83      0.91         6
          10       1.00      1.00      1.00        12
          11       1.00      1.00      1.00         8
          12       1.00      1.00      1.00        15
          13       1.00      1.00      1.00        10
          14       0.90      1.00      0.95         9
          15       1.00      1.00      1.00         6
          16       1.00      1.00      1.00         8

    accuracy                           0.99       183
   macro avg       0.99      0.99      0.99       183
weighted avg       0.99      0.99      0.99       183
```

Activity 2: Save the Model

The model is saved with .h5 extension as follows
An H5 file is a data file saved in the Hierarchical Data Format (HDF). It contains multidimensional arrays of scientific data.

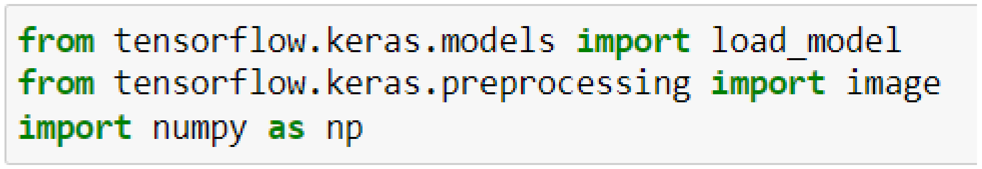


```
In [25]: save_model(model,'vgg-16-nail.h5')
```

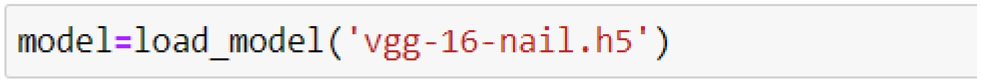
Milestone 4: Testing

Activity 1: Test the model

Evaluation is a process during the development of the model to check whether the model is the best fit for the given problem and corresponding data.
Load the saved model using load_model

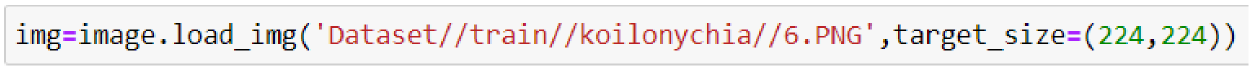


```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
```

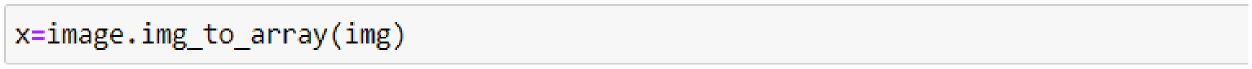


```
model=load_model('vgg-16-nail.h5')
```

Taking an image as input and checking the results



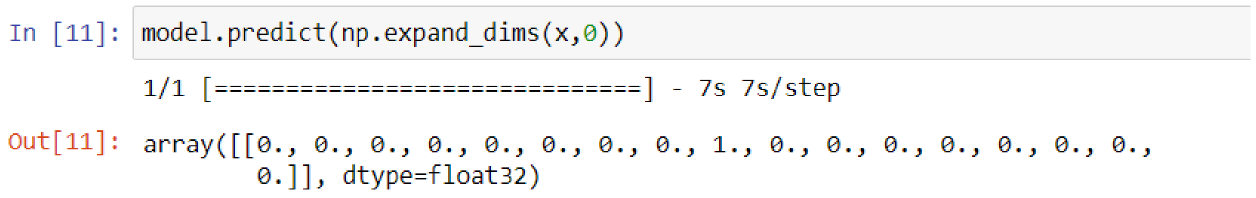
```
img=image.load_img('Dataset//train//koilonychia//6.PNG',target_size=(224,224))
```



```
x=image.img_to_array(img)
```

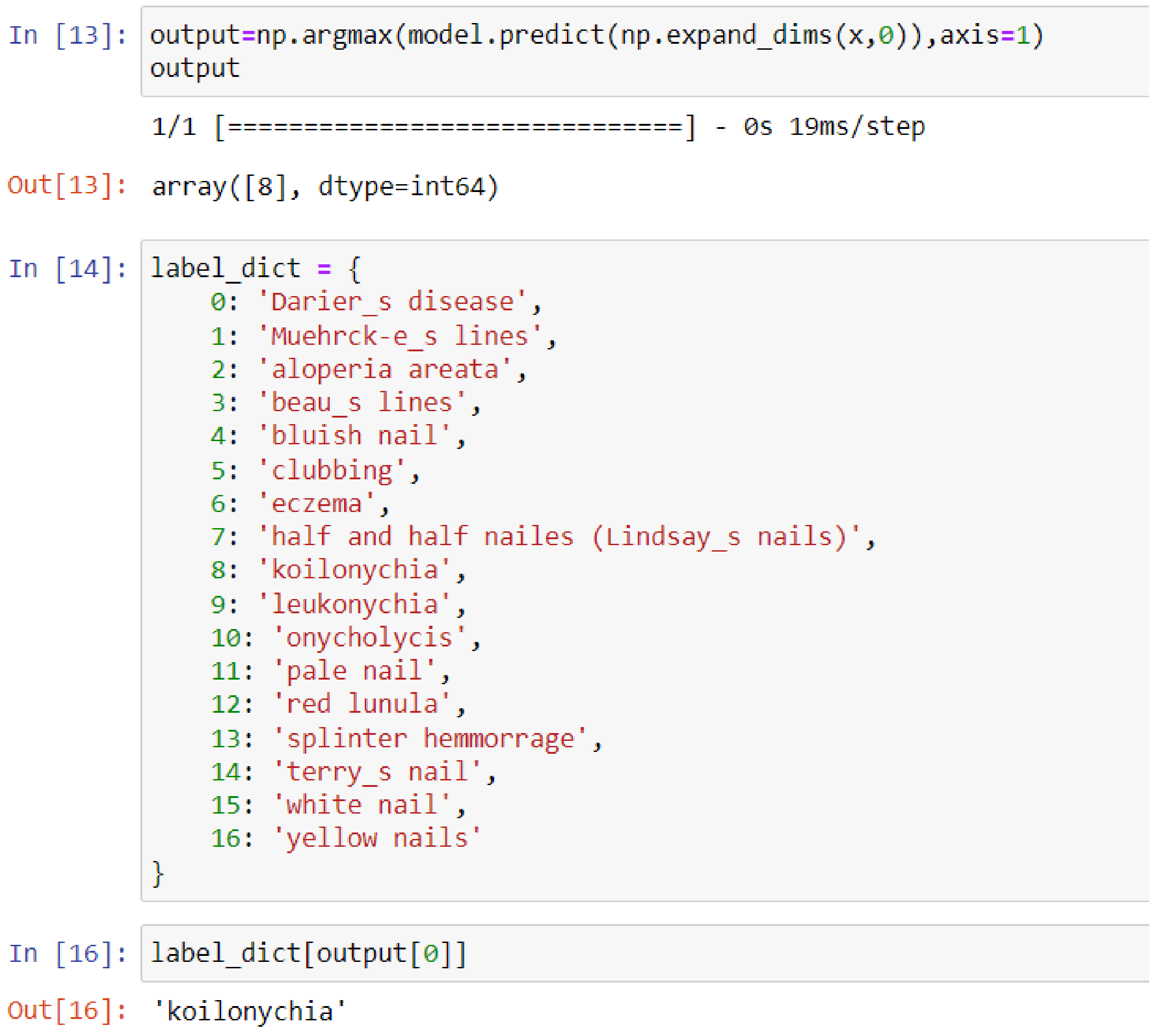
Note: We used VGG16, for image is 224,224, so change according to the model

By using the model we are predicting the output for the given input image



```
In [11]: model.predict(np.expand_dims(x,0))

         1/1 [==============================] - 7s 7s/step

Out[11]: array([[0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0.,
                 0.]], dtype=float32)
```

The predicted class index name will be printed here.



```
In [13]: output=np.argmax(model.predict(np.expand_dims(x,0)),axis=1)
         output

         1/1 [==============================] - 0s 19ms/step

Out[13]: array([8], dtype=int64)

In [14]: label_dict = {
             0: 'Darier_s disease',
             1: 'Muehrck-e_s lines',
             2: 'aloperia areata',
             3: 'beau_s lines',
             4: 'bluish nail',
             5: 'clubbing',
             6: 'eczema',
             7: 'half and half nailes (Lindsay_s nails)',
             8: 'koilonychia',
             9: 'leukonychia',
             10: 'onycholycis',
             11: 'pale nail',
             12: 'red lunula',
             13: 'splinter hemmorrage',
             14: 'terry_s nail',
             15: 'white nail',
             16: 'yellow nails'
         }

In [16]: label_dict[output[0]]

Out[16]: 'koilonychia'
```

Milestone 5: Application Building

Now that we have trained our model, let us build our Streamlit application which will be running in our local browser with a user interface.

In the Streamlit application, there are multiple pages, the image is taken on the predict page and given to model and predicts the output and give simple recommendations.

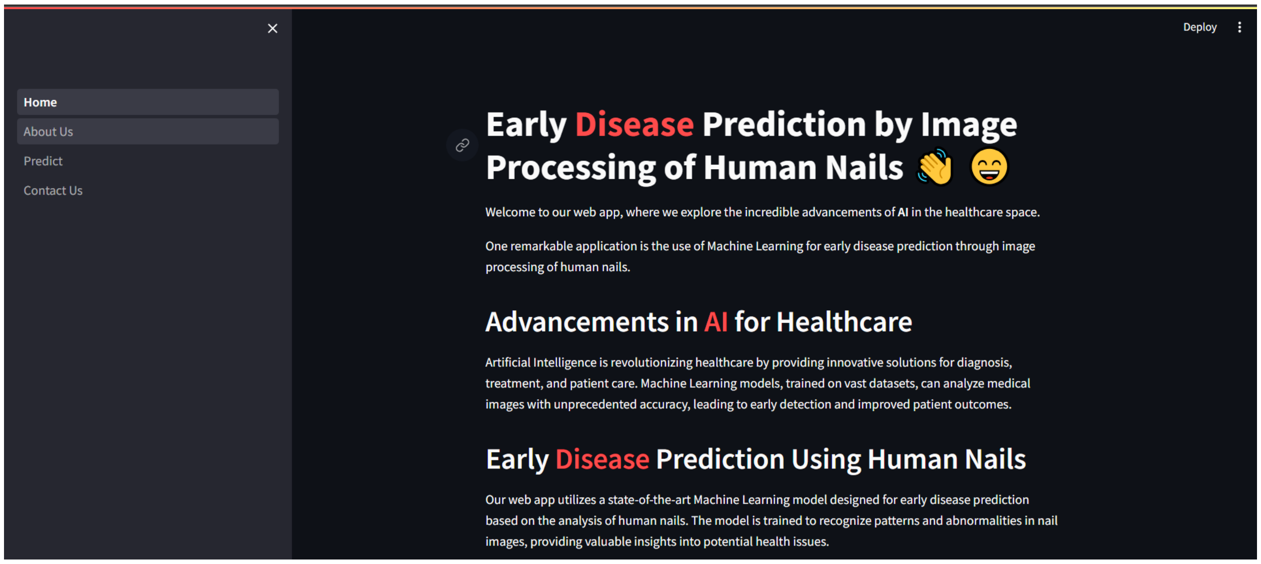
Activity 1: Create Streamlit pages

- o We use Streamlit to create the front end part of the web page.
- o Here, we have created 4 Streamlit pages- Home.py, 2_About Us.py,

3_Predict.py and 4_Contact Us.py

- Home.py displays the home page.
- 2_About Us.py display the about us and our socials
- 3_Predict.py takes the input image and displays the prediction.
- 4_Contact Us.py is a form to provide feedback.

Home page:-



Early Disease Prediction by Image Processing of Human Nails 👋 😄

Welcome to our web app, where we explore the incredible advancements of **AI** in the healthcare space.

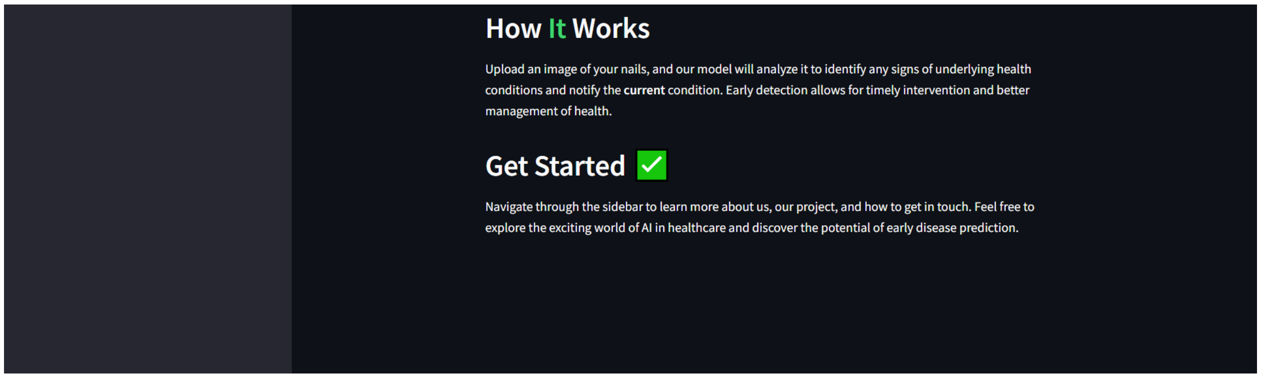
One remarkable application is the use of Machine Learning for early disease prediction through image processing of human nails.

Advancements in AI for Healthcare

Artificial Intelligence is revolutionizing healthcare by providing innovative solutions for diagnosis, treatment, and patient care. Machine Learning models, trained on vast datasets, can analyze medical images with unprecedented accuracy, leading to early detection and improved patient outcomes.

Early Disease Prediction Using Human Nails

Our web app utilizes a state-of-the-art Machine Learning model designed for early disease prediction based on the analysis of human nails. The model is trained to recognize patterns and abnormalities in nail images, providing valuable insights into potential health issues.



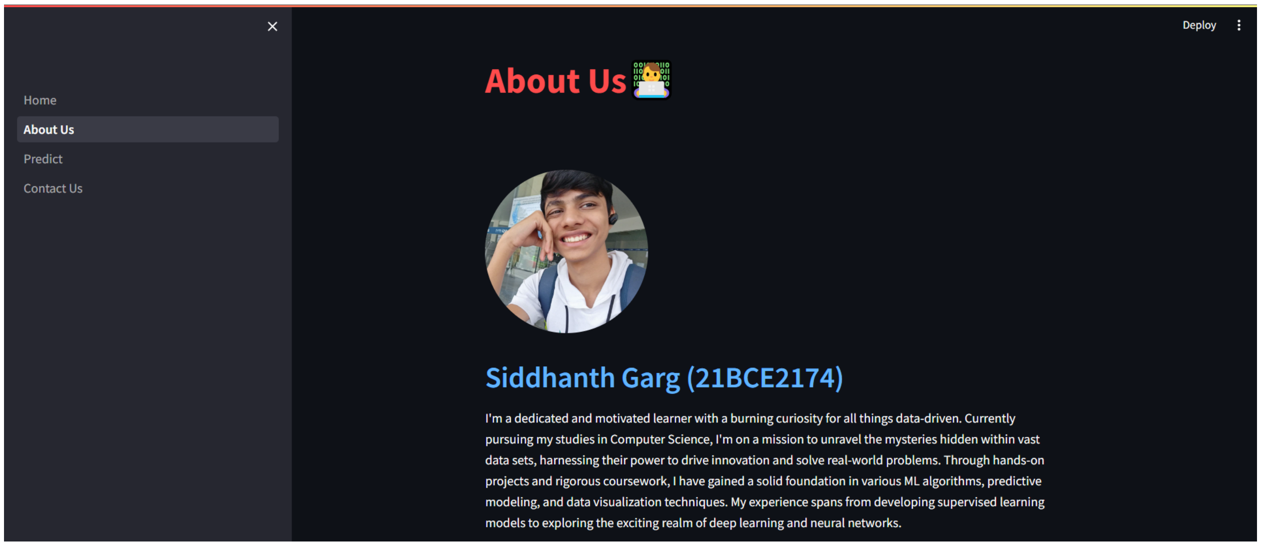
How It Works

Upload an image of your nails, and our model will analyze it to identify any signs of underlying health conditions and notify the **current** condition. Early detection allows for timely intervention and better management of health.

Get Started ✅

Navigate through the sidebar to learn more about us, our project, and how to get in touch. Feel free to explore the exciting world of AI in healthcare and discover the potential of early disease prediction.

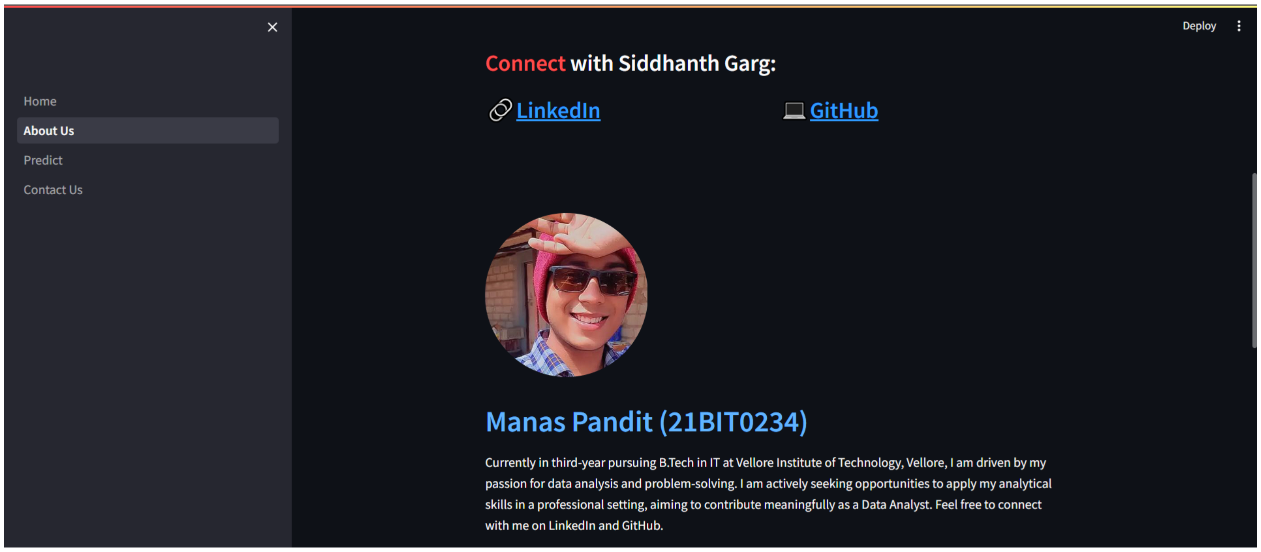
About page:-



About Us 🧑‍💻

Siddhanth Garg (21BCE2174)

I'm a dedicated and motivated learner with a burning curiosity for all things data-driven. Currently pursuing my studies in Computer Science, I'm on a mission to unravel the mysteries hidden within vast data sets, harnessing their power to drive innovation and solve real-world problems. Through hands-on projects and rigorous coursework, I have gained a solid foundation in various ML algorithms, predictive modeling, and data visualization techniques. My experience spans from developing supervised learning models to exploring the exciting realm of deep learning and neural networks.

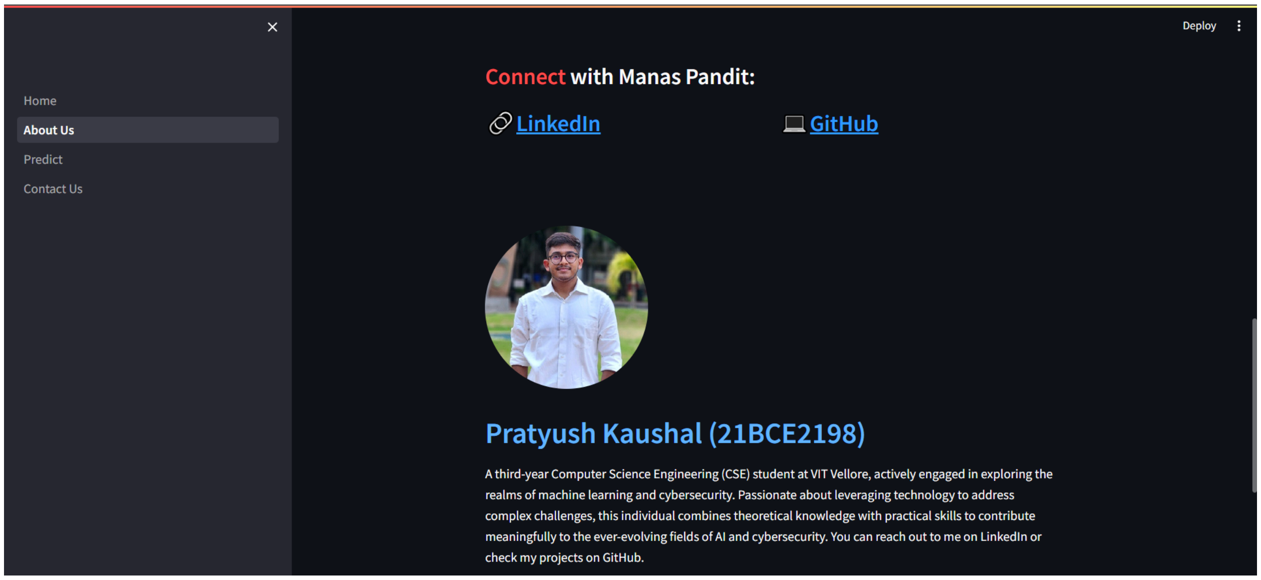


Connect with Siddhanth Garg:

🔗 LinkedIn 💻 GitHub

Manas Pandit (21BIT0234)

Currently in third-year pursuing B.Tech in IT at Vellore Institute of Technology, Vellore, I am driven by my passion for data analysis and problem-solving. I am actively seeking opportunities to apply my analytical skills in a professional setting, aiming to contribute meaningfully as a Data Analyst. Feel free to connect with me on LinkedIn and GitHub.

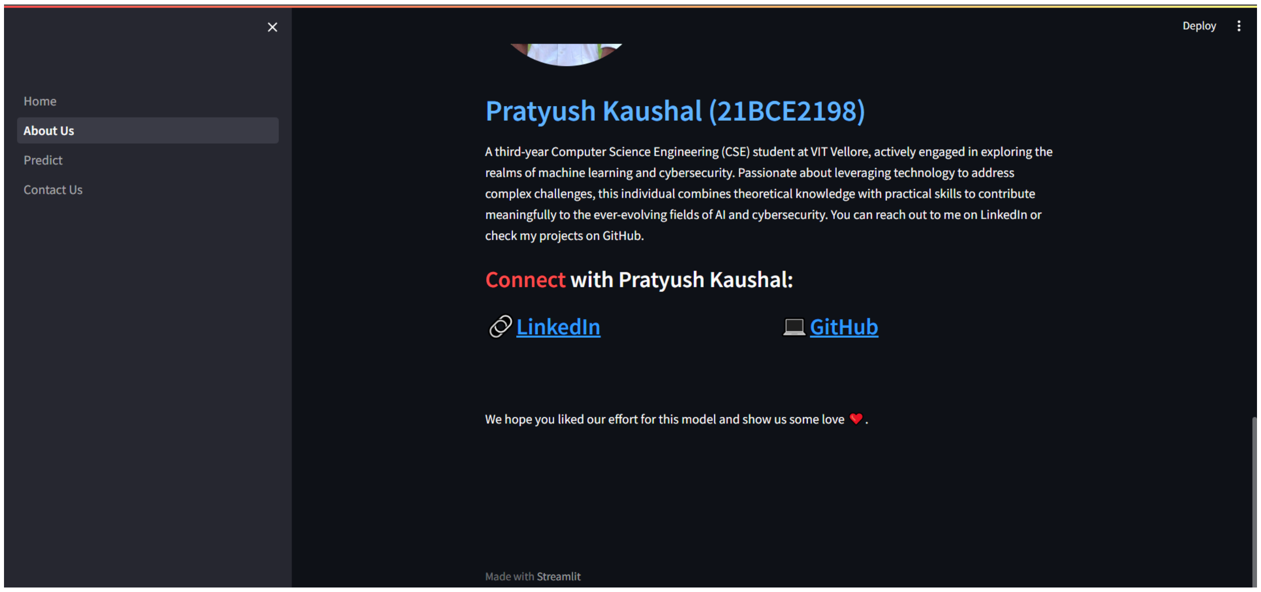


Connect with Manas Pandit:

🔗 LinkedIn 💻 GitHub

Pratyush Kaushal (21BCE2198)

A third-year Computer Science Engineering (CSE) student at VIT Vellore, actively engaged in exploring the realms of machine learning and cybersecurity. Passionate about leveraging technology to address complex challenges, this individual combines theoretical knowledge with practical skills to contribute meaningfully to the ever-evolving fields of AI and cybersecurity. You can reach out to me on LinkedIn or check my projects on GitHub.



Pratyush Kaushal (21BCE2198)

A third-year Computer Science Engineering (CSE) student at VIT Vellore, actively engaged in exploring the realms of machine learning and cybersecurity. Passionate about leveraging technology to address complex challenges, this individual combines theoretical knowledge with practical skills to contribute meaningfully to the ever-evolving fields of AI and cybersecurity. You can reach out to me on LinkedIn or check my projects on GitHub.

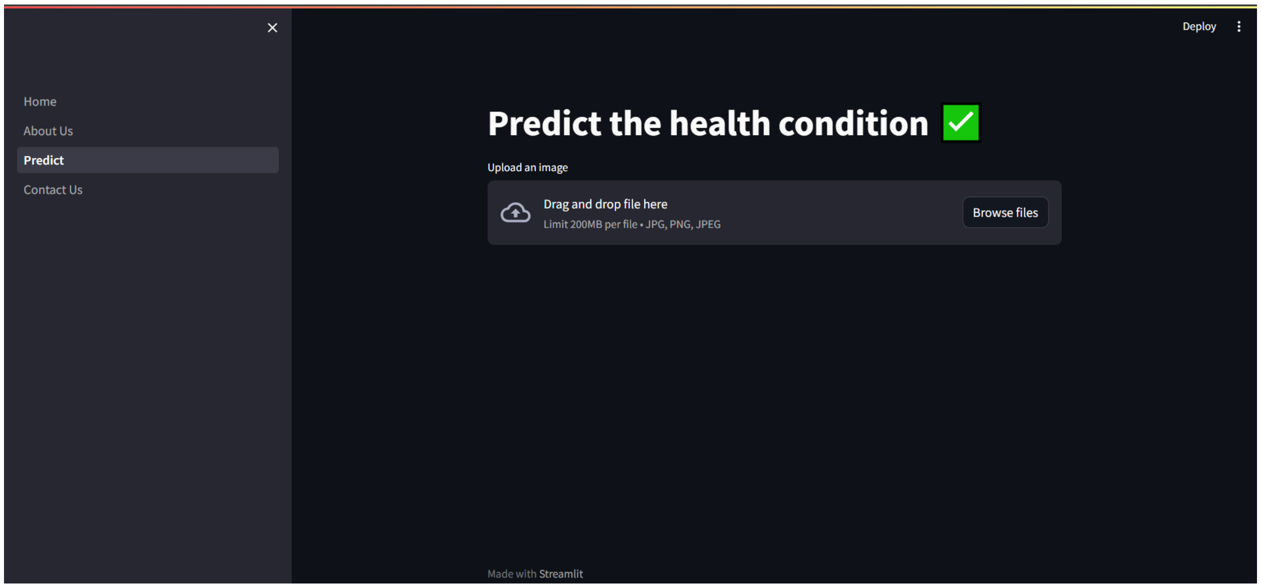
Connect with Pratyush Kaushal:

🔗 LinkedIn 💻 GitHub

We hope you liked our effort for this model and show us some love ❤️.

Made with Streamlit

Nail Disease Prediction Page:-



Predict the health condition ✅

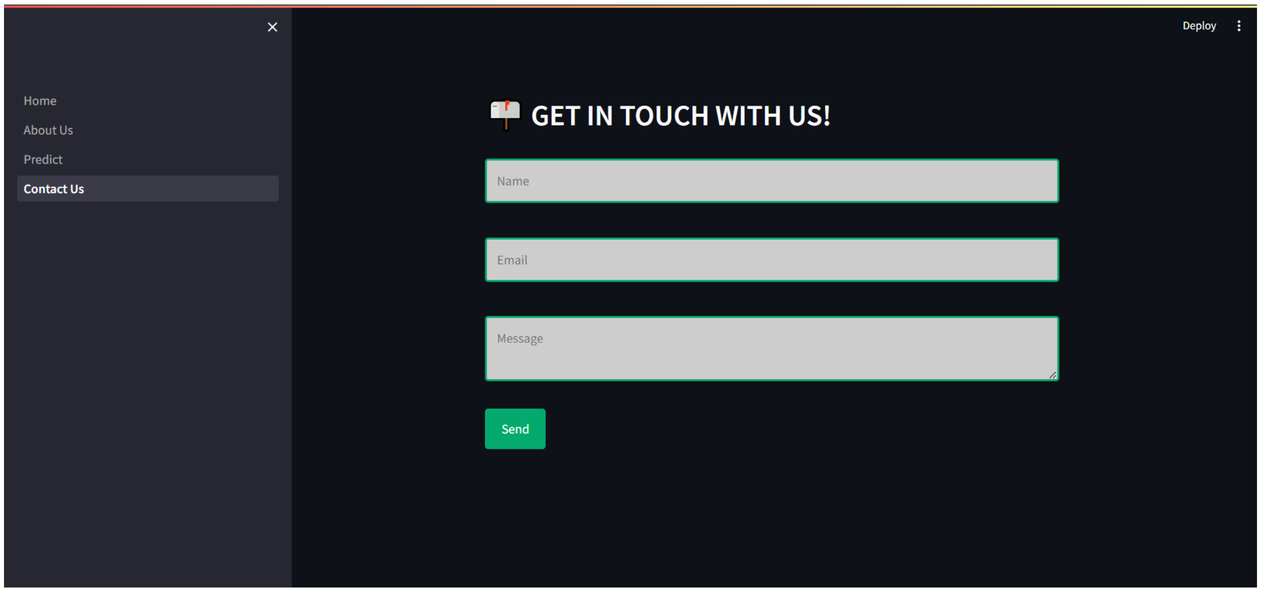
Upload an image

Drag and drop file here
Limit 200MB per file • JPG, PNG, JPEG

Browse files

Made with Streamlit

Contact us page:-



Activity 2: Build python code

Task 1: Importing Libraries

The first step is usually importing the libraries that will be needed in the program.

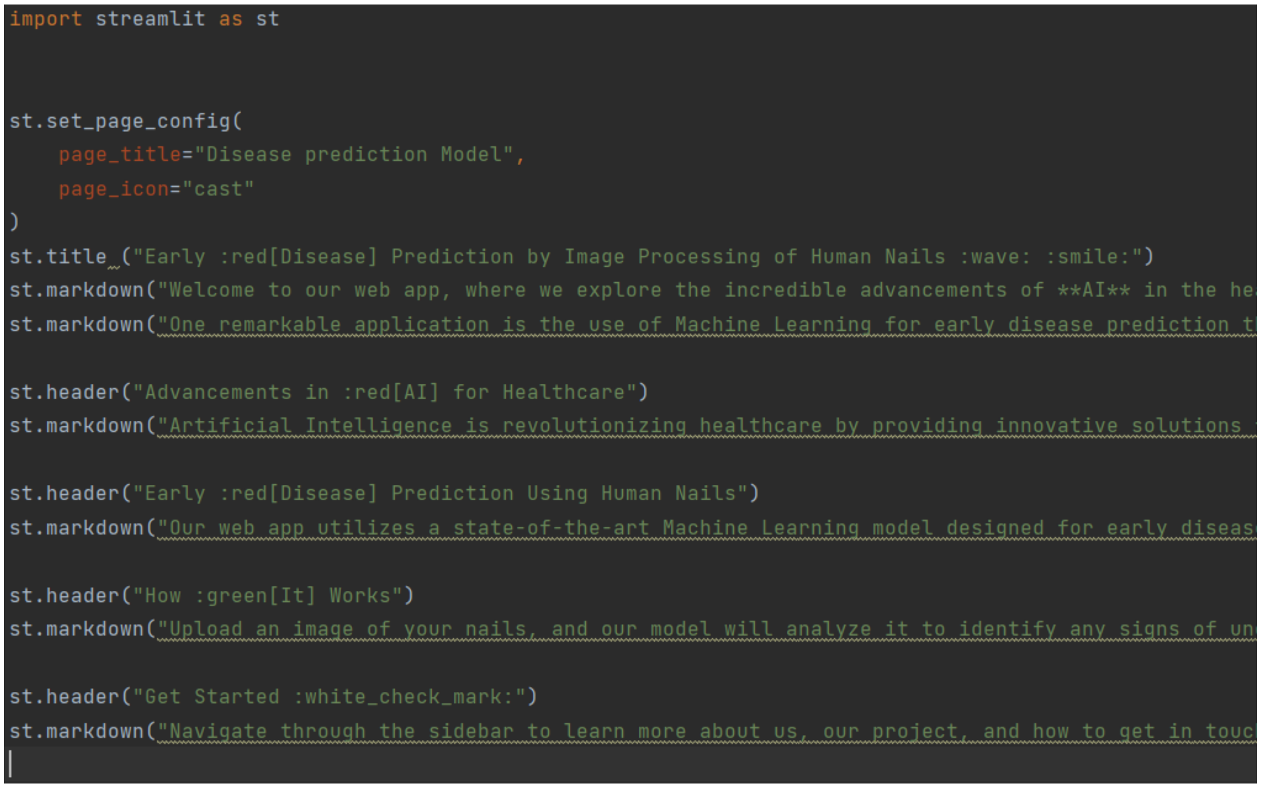
```
import streamlit as st
import tensorflow as tf
from PIL import Image
import numpy as np
from streamlit_option_menu import option_menu
```

Importing the streamlit module in the project is mandatory. Once after loading the libraries, we need to load the model

Task 2: Creating our Streamlit application and loading our model by using load_model method

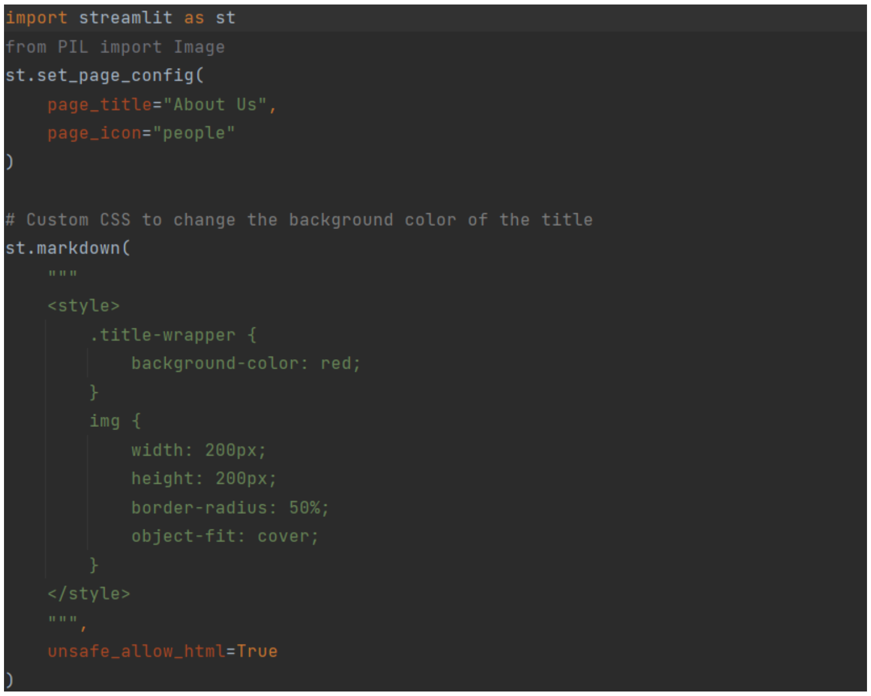
```
model = tf.keras.models.load_model('pages/vgg-16-nail.h5')
```

Default home page

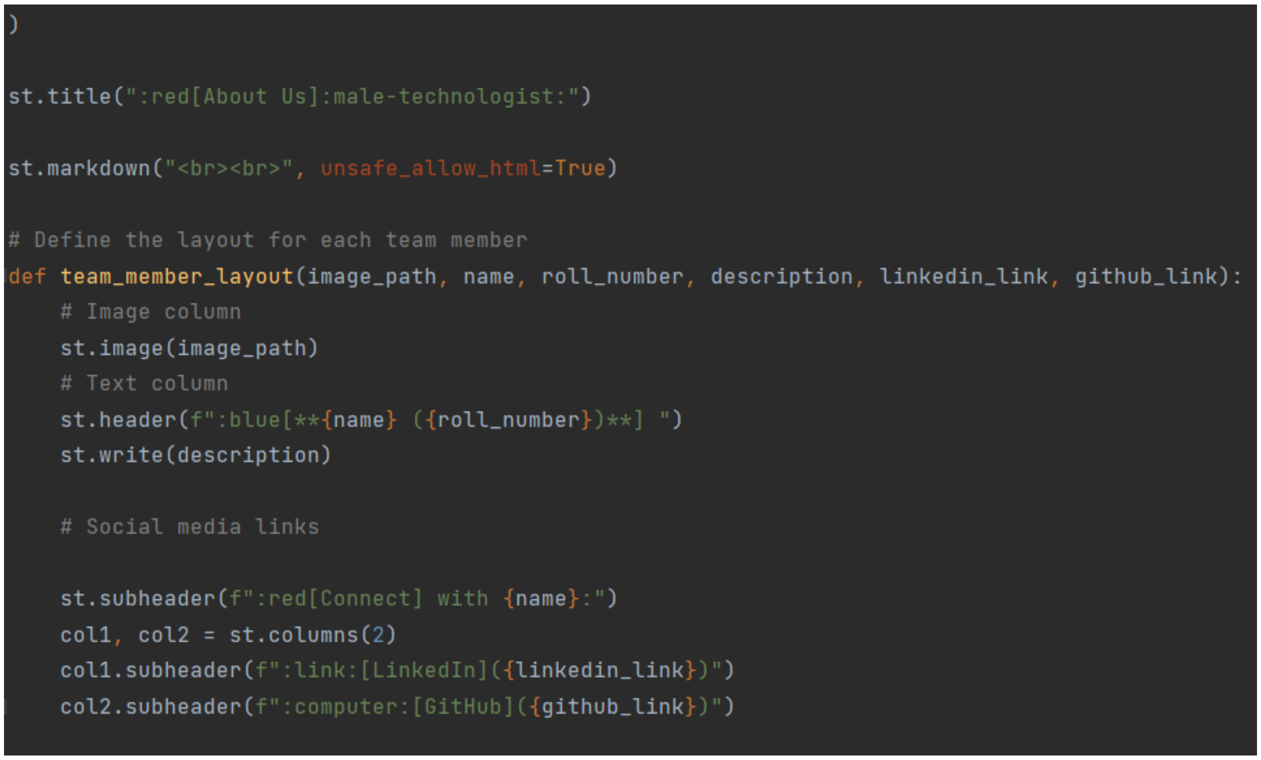


```
import streamlit as st


st.set_page_config(
    page_title="Disease prediction Model",
    page_icon="cast"
)
st.title ("Early :red[Disease] Prediction by Image Processing of Human Nails :wave: :smile:")
st.markdown("Welcome to our web app, where we explore the incredible advancements of **AI** in the he
st.markdown("One remarkable application is the use of Machine Learning for early disease prediction t

st.header("Advancements in :red[AI] for Healthcare")
st.markdown("Artificial Intelligence is revolutionizing healthcare by providing innovative solutions

st.header("Early :red[Disease] Prediction Using Human Nails")
st.markdown("Our web app utilizes a state-of-the-art Machine Learning model designed for early diseas

st.header("How :green[It] Works")
st.markdown("Upload an image of your nails, and our model will analyze it to identify any signs of un

st.header("Get Started :white_check_mark:")
st.markdown("Navigate through the sidebar to learn more about us, our project, and how to get in touc
```

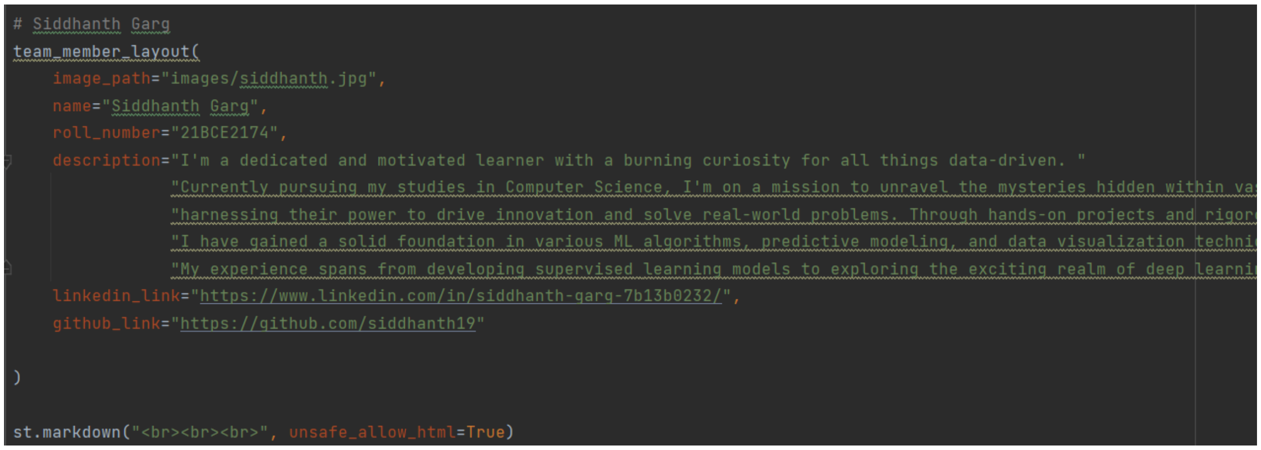
When you click on about us on sidebar, it will redirect you to the about us page



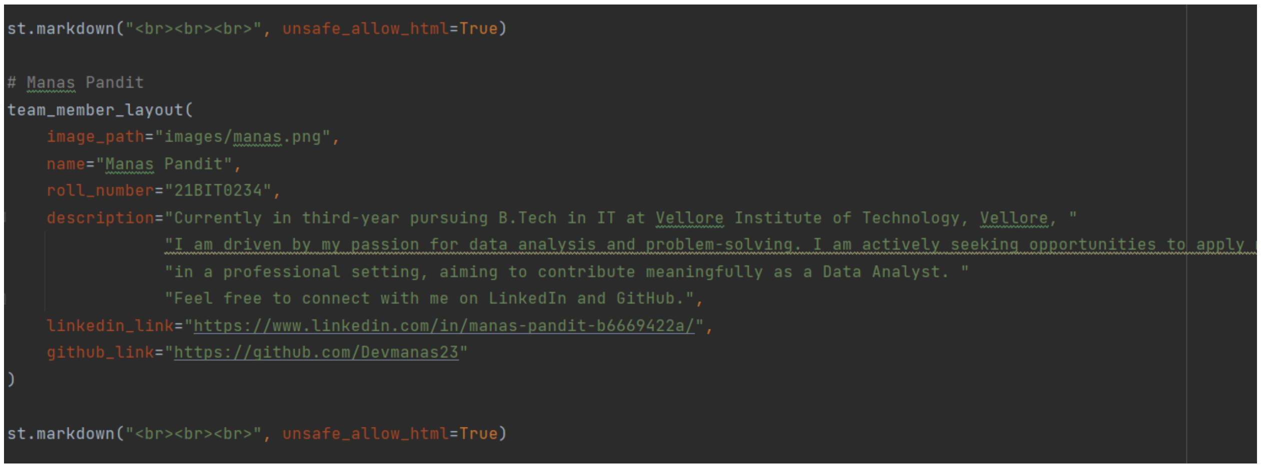
```
import streamlit as st
from PIL import Image
st.set_page_config(
    page_title="About Us",
    page_icon="people"
)

# Custom CSS to change the background color of the title
st.markdown(
    """
    <style>
        .title-wrapper {
            background-color: red;
        }
        img {
            width: 200px;
            height: 200px;
            border-radius: 50%;
            object-fit: cover;
        }
    </style>
    """,
    unsafe_allow_html=True
)
```



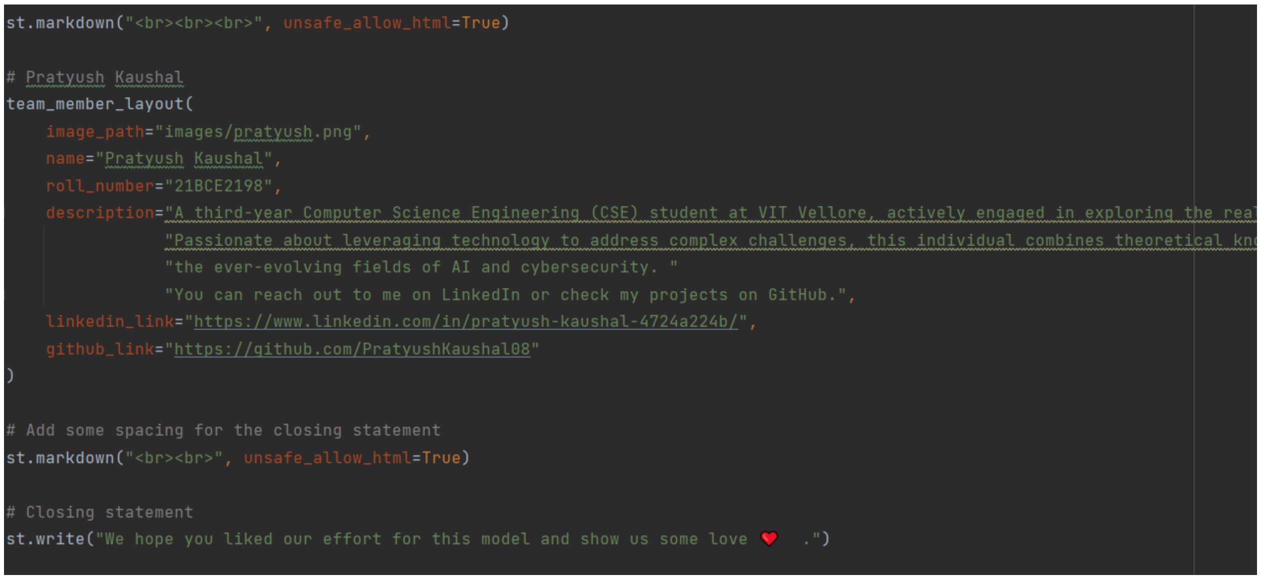
```python
)

st.title(":red[About Us]:male-technologist:")

st.markdown("<br><br>", unsafe_allow_html=True)

# Define the layout for each team member
def team_member_layout(image_path, name, roll_number, description, linkedin_link, github_link):
    # Image column
    st.image(image_path)
    # Text column
    st.header(f":blue[**{name} ({roll_number})**] ")
    st.write(description)

    # Social media links

    st.subheader(f":red[Connect] with {name}:")
    col1, col2 = st.columns(2)
    col1.subheader(f":link:[LinkedIn]({linkedin_link})")
    col2.subheader(f":computer:[GitHub]({github_link})")
```



```python
# Siddhanth Garg
team_member_layout(
    image_path="images/siddhanth.jpg",
    name="Siddhanth Garg",
    roll_number="21BCE2174",
    description="I'm a dedicated and motivated learner with a burning curiosity for all things data-driven. "
                "Currently pursuing my studies in Computer Science, I'm on a mission to unravel the mysteries hidden within va
                "harnessing their power to drive innovation and solve real-world problems. Through hands-on projects and rigor
                "I have gained a solid foundation in various ML algorithms, predictive modeling, and data visualization techni
                "My experience spans from developing supervised learning models to exploring the exciting realm of deep learni
    linkedin_link="https://www.linkedin.com/in/siddhanth-garg-7b13b0232/",
    github_link="https://github.com/siddhanth19"

)

st.markdown("<br><br><br>", unsafe_allow_html=True)
```

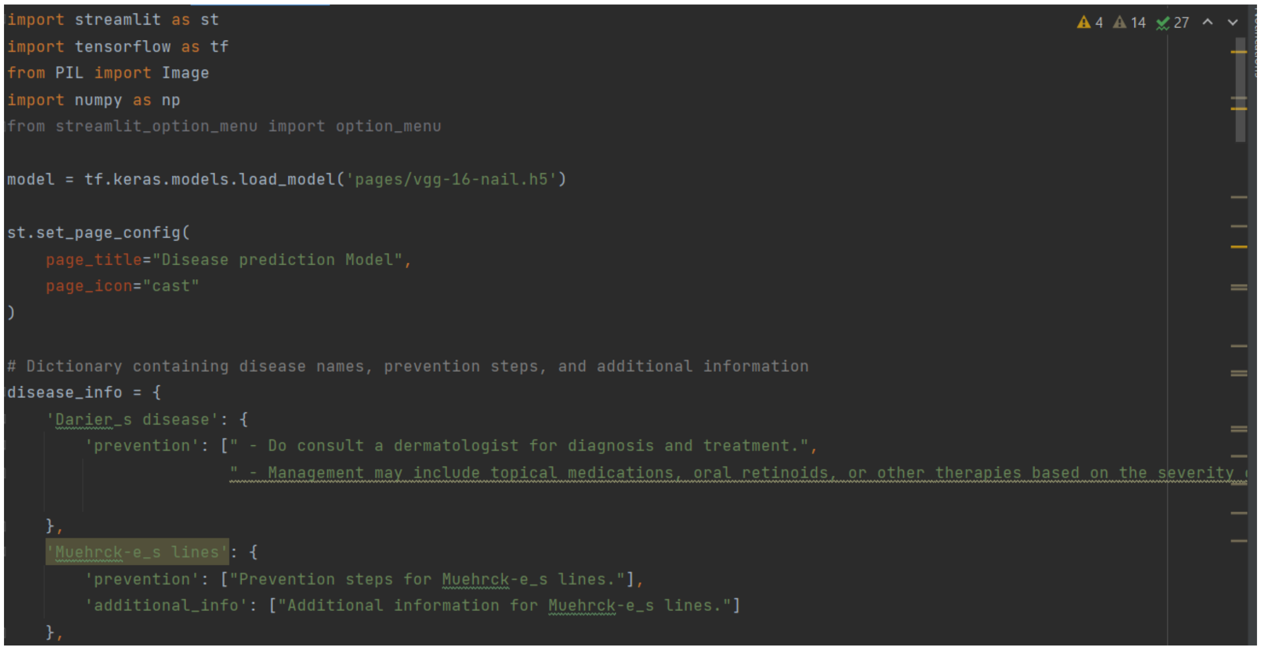


```
st.markdown("<br><br><br>", unsafe_allow_html=True)

# Manas Pandit
team_member_layout(
    image_path="images/manas.png",
    name="Manas Pandit",
    roll_number="21BIT0234",
    description="Currently in third-year pursuing B.Tech in IT at Vellore Institute of Technology, Vellore, "
                "I am driven by my passion for data analysis and problem-solving. I am actively seeking opportunities to apply
                "in a professional setting, aiming to contribute meaningfully as a Data Analyst. "
                "Feel free to connect with me on LinkedIn and GitHub.",
    linkedin_link="https://www.linkedin.com/in/manas-pandit-b6669422a/",
    github_link="https://github.com/Devmanas23"
)

st.markdown("<br><br><br>", unsafe_allow_html=True)
```

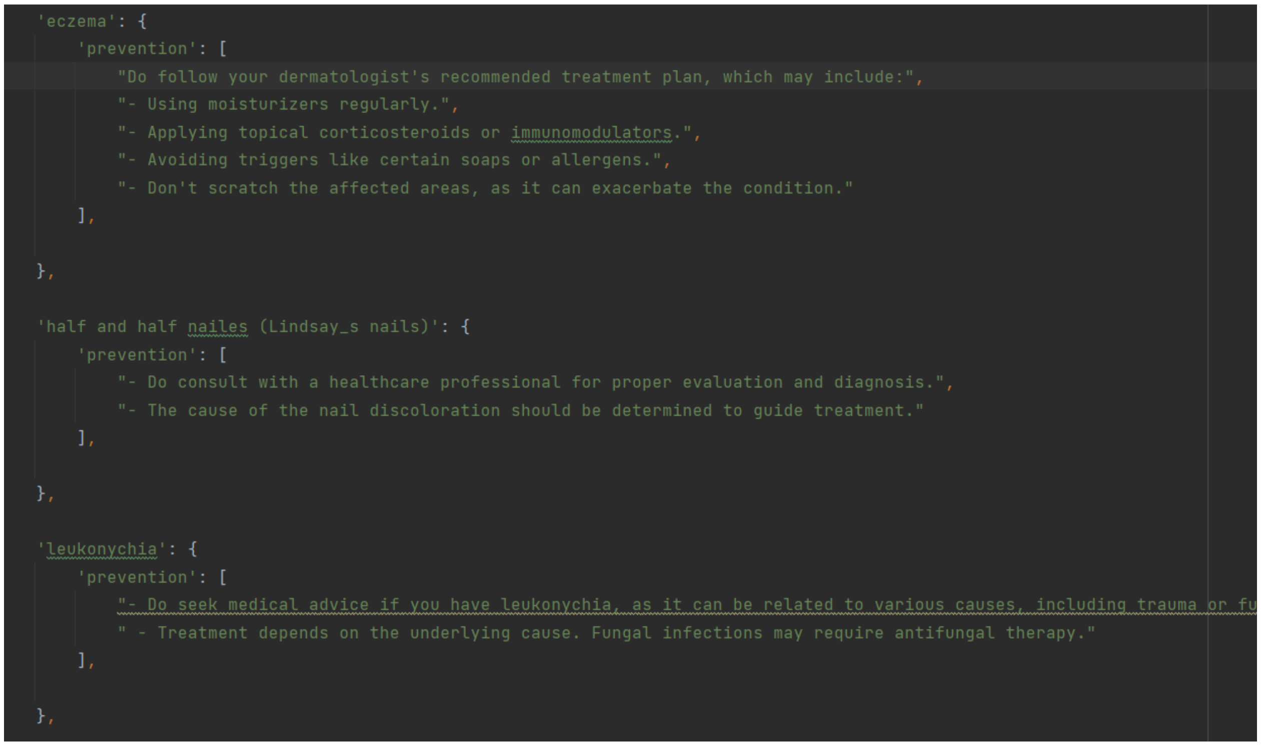


```
st.markdown("<br><br><br>", unsafe_allow_html=True)

# Pratyush Kaushal
team_member_layout(
    image_path="images/pratyush.png",
    name="Pratyush Kaushal",
    roll_number="21BCE2198",
    description="A third-year Computer Science Engineering (CSE) student at VIT Vellore, actively engaged in exploring the rea
                "Passionate about leveraging technology to address complex challenges, this individual combines theoretical kn
                "the ever-evolving fields of AI and cybersecurity. "
                "You can reach out to me on LinkedIn or check my projects on GitHub.",
    linkedin_link="https://www.linkedin.com/in/pratyush-kaushal-4724a224b/",
    github_link="https://github.com/PratyushKaushal08"
)

# Add some spacing for the closing statement
st.markdown("<br><br>", unsafe_allow_html=True)

# Closing statement
st.write("We hope you liked our effort for this model and show us some love ❤️ .")
```

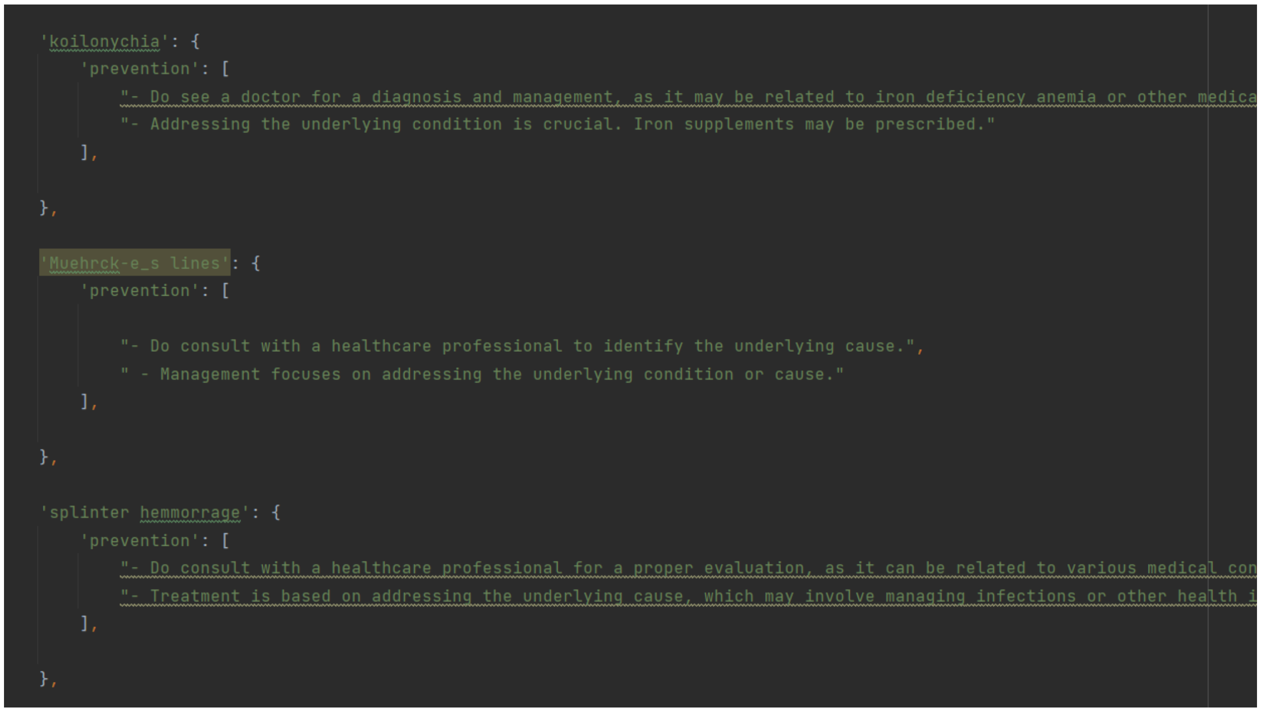
When you click on predict , it will redirect you to the Nail Disease Predict page



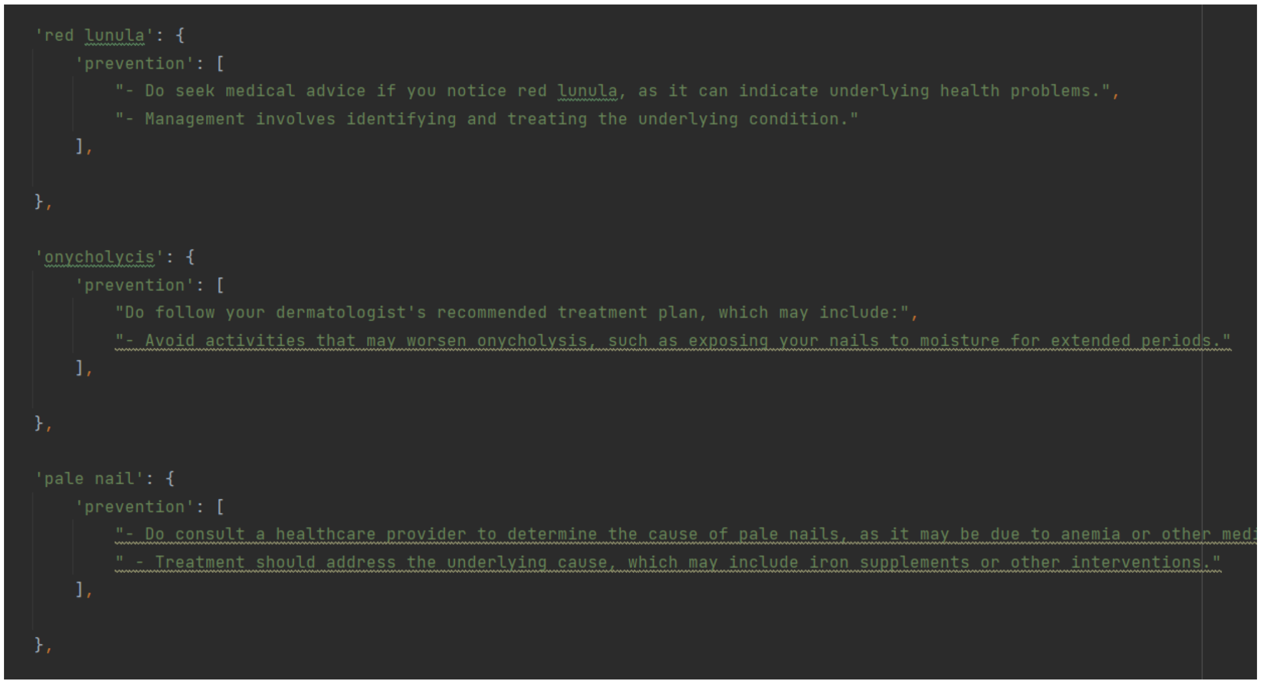
```python
import streamlit as st
import tensorflow as tf
from PIL import Image
import numpy as np
from streamlit_option_menu import option_menu

model = tf.keras.models.load_model('pages/vgg-16-nail.h5')

st.set_page_config(
    page_title="Disease prediction Model",
    page_icon="cast"
)

# Dictionary containing disease names, prevention steps, and additional information
disease_info = {
    'Darier_s disease': {
        'prevention': [" - Do consult a dermatologist for diagnosis and treatment.",
                       " - Management may include topical medications, oral retinoids, or other therapies based on the severity

    },
    'Muehrck-e_s lines': {
        'prevention': ["Prevention steps for Muehrck-e_s lines."],
        'additional_info': ["Additional information for Muehrck-e_s lines."]
    },
```



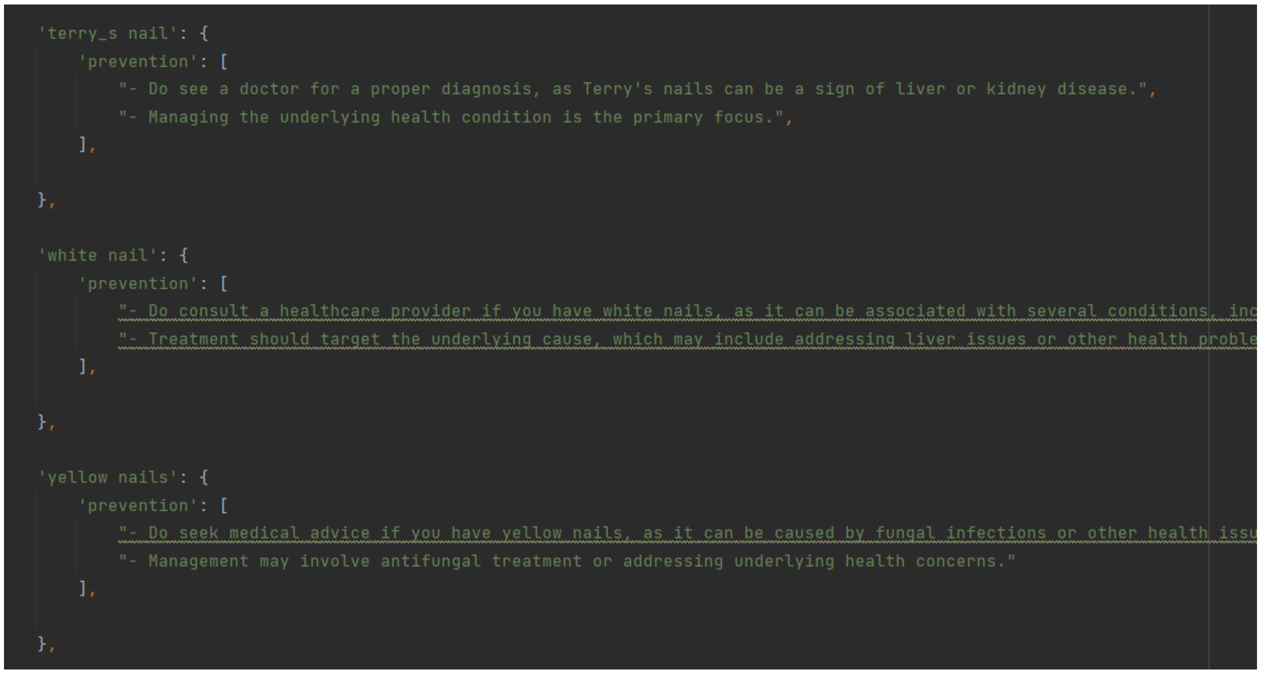
```python
    'eczema': {
        'prevention': [
            "Do follow your dermatologist's recommended treatment plan, which may include:",
            "- Using moisturizers regularly.",
            "- Applying topical corticosteroids or immunomodulators.",
            "- Avoiding triggers like certain soaps or allergens.",
            "- Don't scratch the affected areas, as it can exacerbate the condition."
        ],

    },

    'half and half nailes (Lindsay_s nails)': {
        'prevention': [
            "- Do consult with a healthcare professional for proper evaluation and diagnosis.",
            "- The cause of the nail discoloration should be determined to guide treatment."
        ],

    },

    'leukonychia': {
        'prevention': [
            "- Do seek medical advice if you have leukonychia, as it can be related to various causes, including trauma or fu
            " - Treatment depends on the underlying cause. Fungal infections may require antifungal therapy."
        ],

    },
```



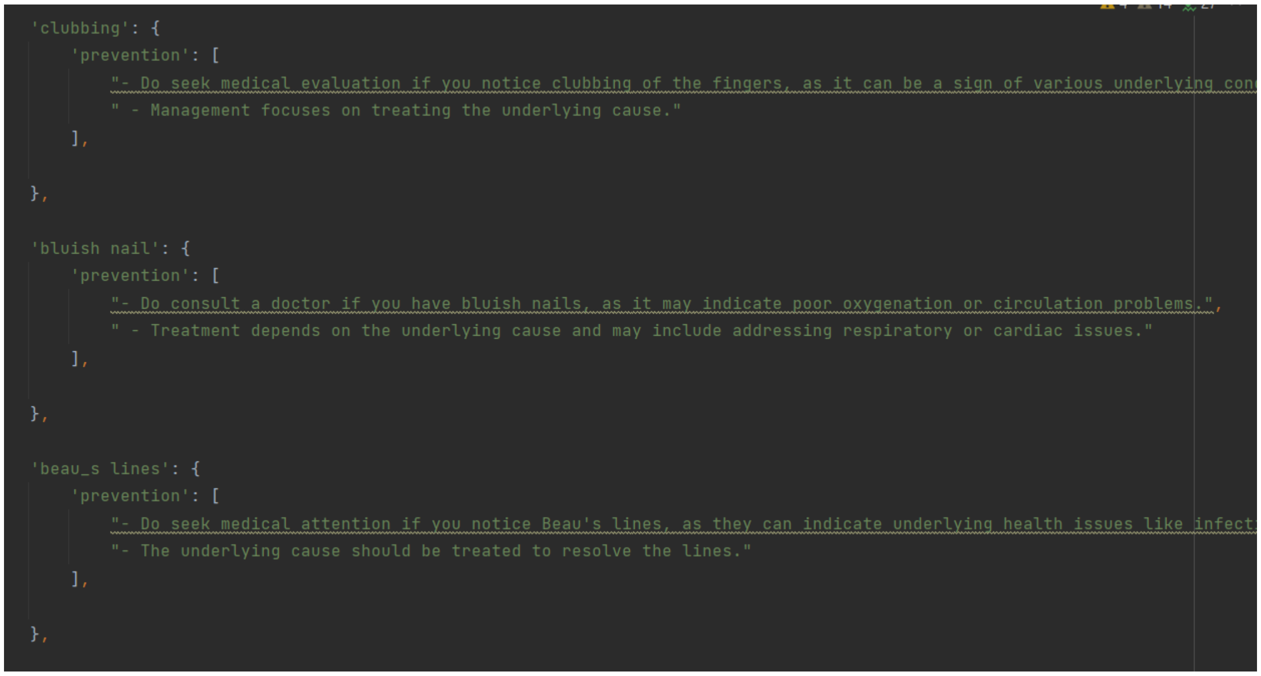
```
'koilonychia': {
    'prevention': [
        "- Do see a doctor for a diagnosis and management, as it may be related to iron deficiency anemia or other medica
        "- Addressing the underlying condition is crucial. Iron supplements may be prescribed."
    ],

},

'Muehrck-e_s lines': {
    'prevention': [

        "- Do consult with a healthcare professional to identify the underlying cause.",
        " - Management focuses on addressing the underlying condition or cause."
    ],

},

'splinter hemmorrage': {
    'prevention': [
        "- Do consult with a healthcare professional for a proper evaluation, as it can be related to various medical con
        "- Treatment is based on addressing the underlying cause, which may involve managing infections or other health i
    ],

},
```



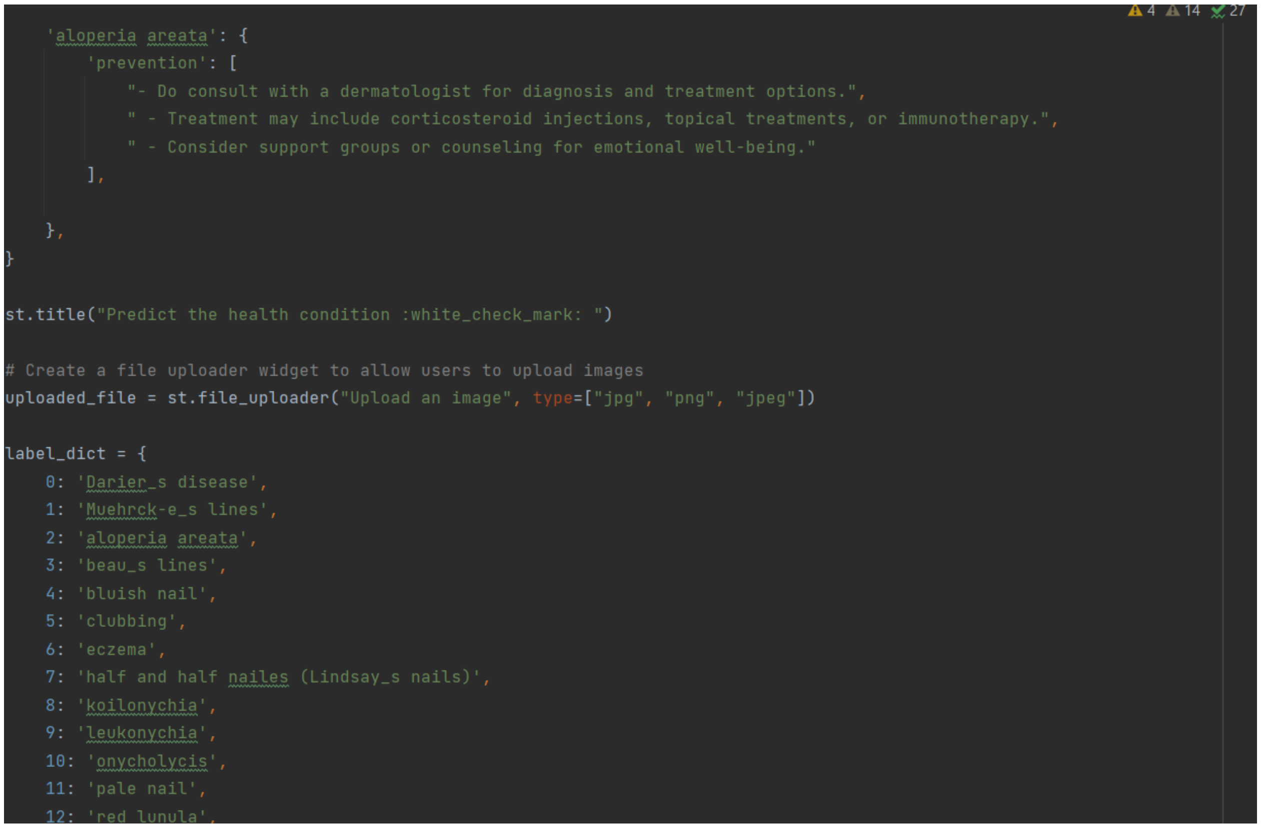
```
'red lunula': {
    'prevention': [
        "- Do seek medical advice if you notice red lunula, as it can indicate underlying health problems.",
        "- Management involves identifying and treating the underlying condition."
    ],

},

'onycholycis': {
    'prevention': [
        "Do follow your dermatologist's recommended treatment plan, which may include:",
        "- Avoid activities that may worsen onycholysis, such as exposing your nails to moisture for extended periods."
    ],

},

'pale nail': {
    'prevention': [
        "- Do consult a healthcare provider to determine the cause of pale nails, as it may be due to anemia or other medi
        " - Treatment should address the underlying cause, which may include iron supplements or other interventions."
    ],

},
```



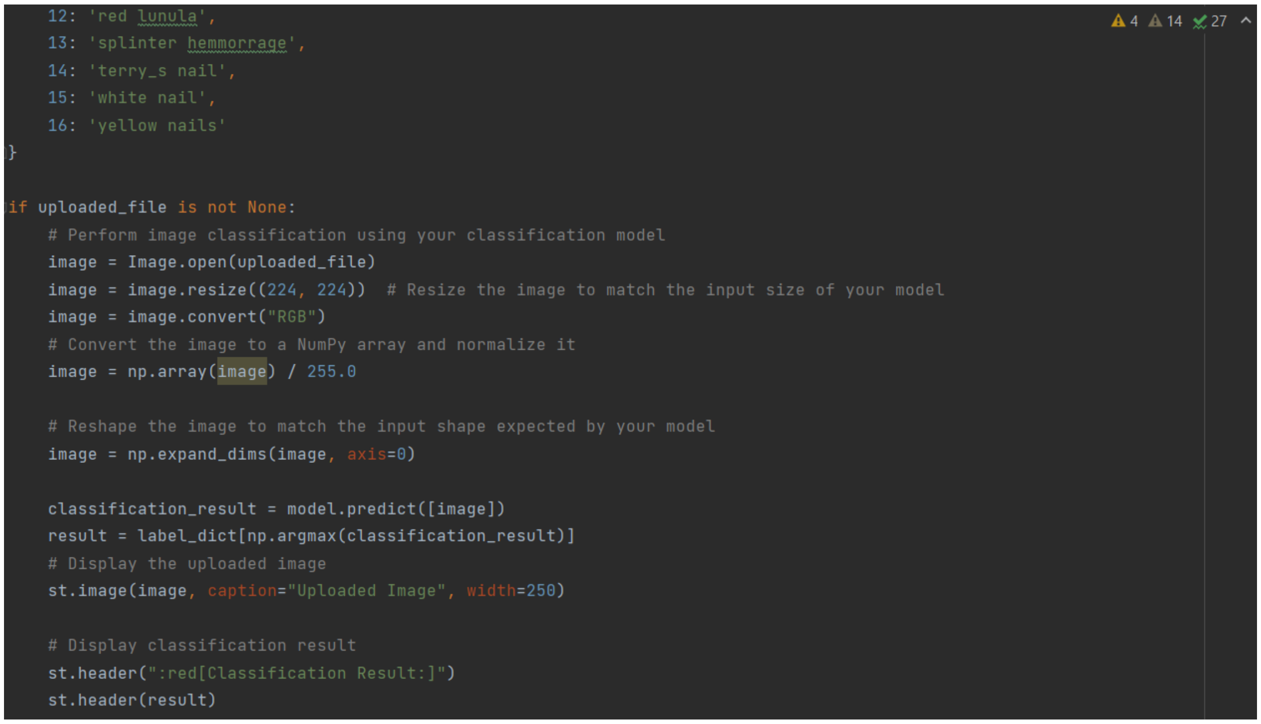
```
'terry_s nail': {
    'prevention': [
        "- Do see a doctor for a proper diagnosis, as Terry's nails can be a sign of liver or kidney disease.",
        "- Managing the underlying health condition is the primary focus.",
    ],

},

'white nail': {
    'prevention': [
        "- Do consult a healthcare provider if you have white nails, as it can be associated with several conditions, inc
        "- Treatment should target the underlying cause, which may include addressing liver issues or other health proble
    ],

},

'yellow nails': {
    'prevention': [
        "- Do seek medical advice if you have yellow nails, as it can be caused by fungal infections or other health issu
        "- Management may involve antifungal treatment or addressing underlying health concerns."
    ],

},
```



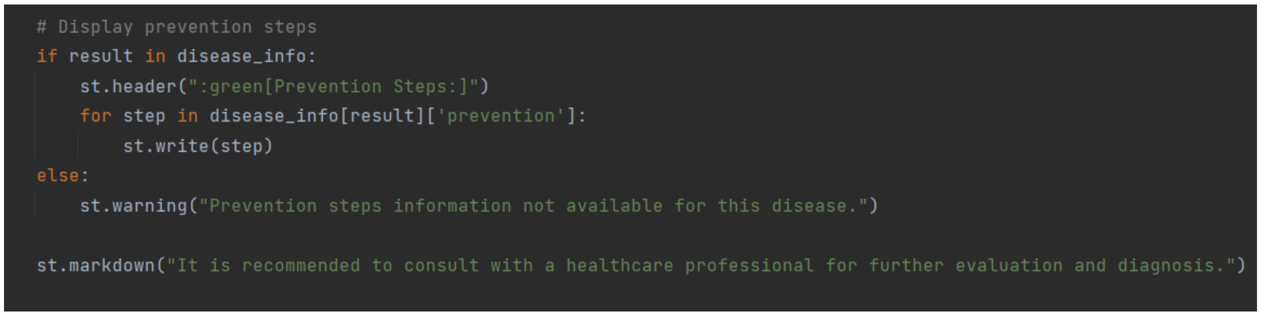
```
'clubbing': {
    'prevention': [
        "- Do seek medical evaluation if you notice clubbing of the fingers, as it can be a sign of various underlying con
        " - Management focuses on treating the underlying cause."
    ],

},

'bluish nail': {
    'prevention': [
        "- Do consult a doctor if you have bluish nails, as it may indicate poor oxygenation or circulation problems.",
        " - Treatment depends on the underlying cause and may include addressing respiratory or cardiac issues."
    ],

},

'beau_s lines': {
    'prevention': [
        "- Do seek medical attention if you notice Beau's lines, as they can indicate underlying health issues like infect
        "- The underlying cause should be treated to resolve the lines."
    ],

},
```



```
    'aloperia areata': {
        'prevention': [
            "- Do consult with a dermatologist for diagnosis and treatment options.",
            " - Treatment may include corticosteroid injections, topical treatments, or immunotherapy.",
            " - Consider support groups or counseling for emotional well-being."
        ],

    },
}

st.title("Predict the health condition :white_check_mark: ")

# Create a file uploader widget to allow users to upload images
uploaded_file = st.file_uploader("Upload an image", type=["jpg", "png", "jpeg"])

label_dict = {
    0: 'Darier_s disease',
    1: 'Muehrck-e_s lines',
    2: 'aloperia areata',
    3: 'beau_s lines',
    4: 'bluish nail',
    5: 'clubbing',
    6: 'eczema',
    7: 'half and half nailes (Lindsay_s nails)',
    8: 'koilonychia',
    9: 'leukonychia',
    10: 'onycholycis',
    11: 'pale nail',
    12: 'red lunula',
```

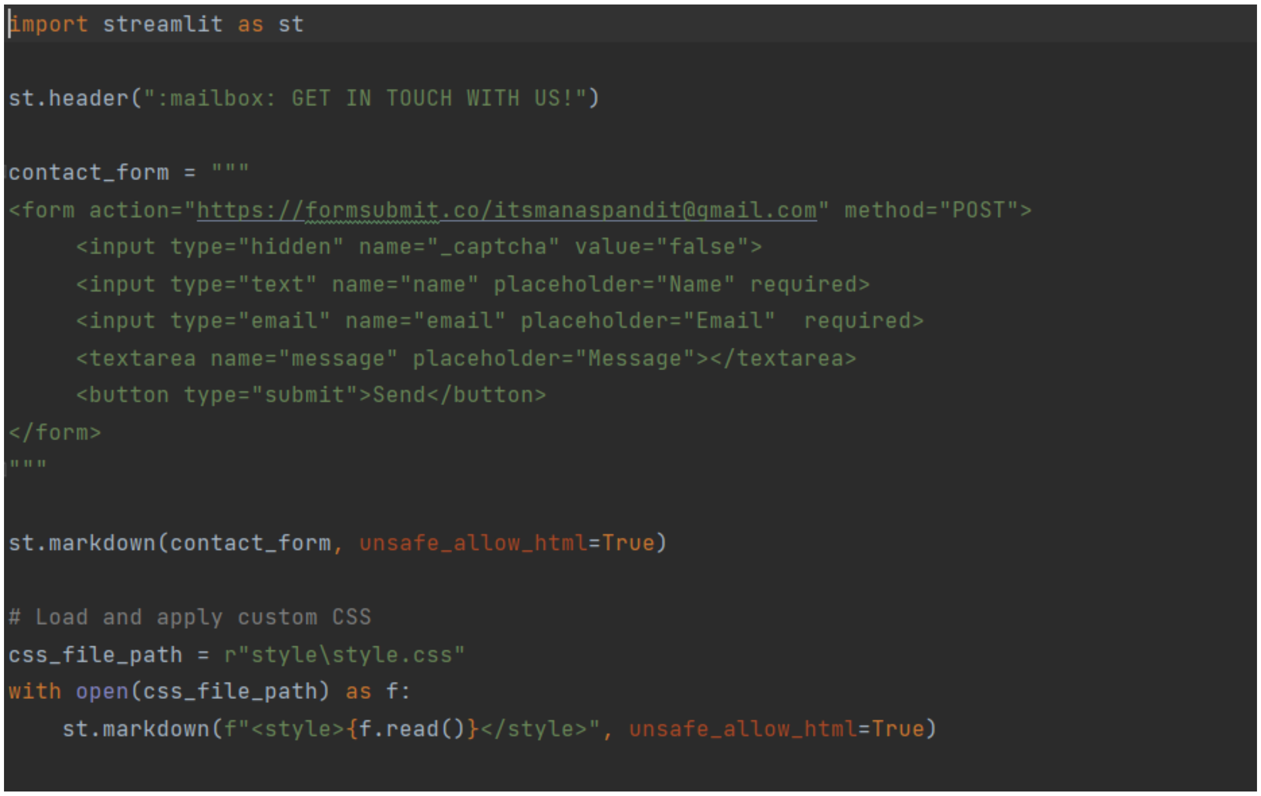


```
    12: 'red lunula',
    13: 'splinter hemmorrage',
    14: 'terry_s nail',
    15: 'white nail',
    16: 'yellow nails'
}

if uploaded_file is not None:
    # Perform image classification using your classification model
    image = Image.open(uploaded_file)
    image = image.resize((224, 224))  # Resize the image to match the input size of your model
    image = image.convert("RGB")
    # Convert the image to a NumPy array and normalize it
    image = np.array(image) / 255.0

    # Reshape the image to match the input shape expected by your model
    image = np.expand_dims(image, axis=0)

    classification_result = model.predict([image])
    result = label_dict[np.argmax(classification_result)]
    # Display the uploaded image
    st.image(image, caption="Uploaded Image", width=250)

    # Display classification result
    st.header(":red[Classification Result:]")
    st.header(result)
```



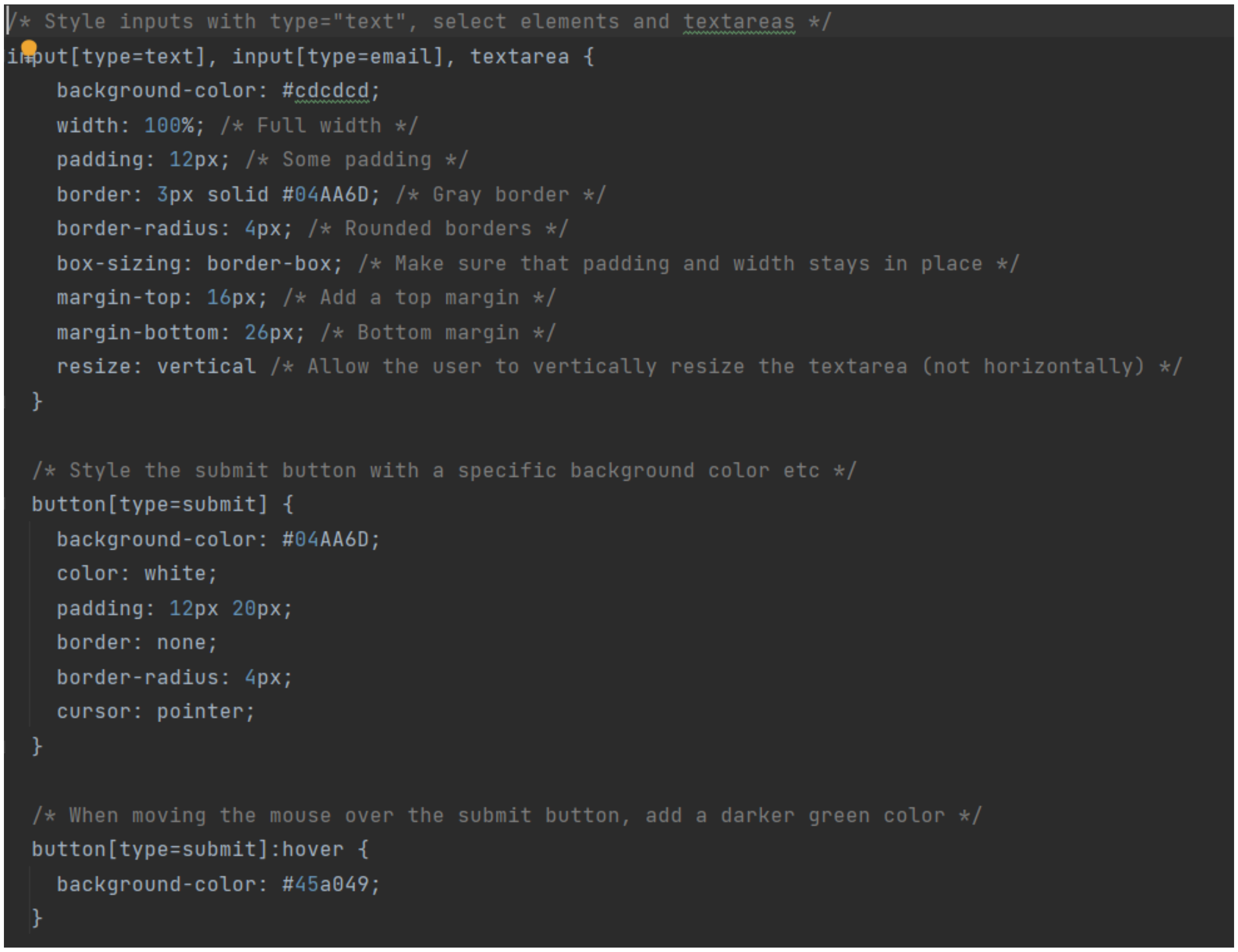
```python
# Display prevention steps
if result in disease_info:
    st.header(":green[Prevention Steps:]")
    for step in disease_info[result]['prevention']:
        st.write(step)
else:
    st.warning("Prevention steps information not available for this disease.")

st.markdown("It is recommended to consult with a healthcare professional for further evaluation and diagnosis.")
```

When you click on contact us, it will redirect you to the contact us page



```python
import streamlit as st

st.header(":mailbox: GET IN TOUCH WITH US!")

contact_form = """
<form action="https://formsubmit.co/itsmanaspandit@gmail.com" method="POST">
     <input type="hidden" name="_captcha" value="false">
     <input type="text" name="name" placeholder="Name" required>
     <input type="email" name="email" placeholder="Email"  required>
     <textarea name="message" placeholder="Message"></textarea>
     <button type="submit">Send</button>
</form>
"""

st.markdown(contact_form, unsafe_allow_html=True)

# Load and apply custom CSS
css_file_path = r"style\style.css"
with open(css_file_path) as f:
    st.markdown(f"<style>{f.read()}</style>", unsafe_allow_html=True)
```

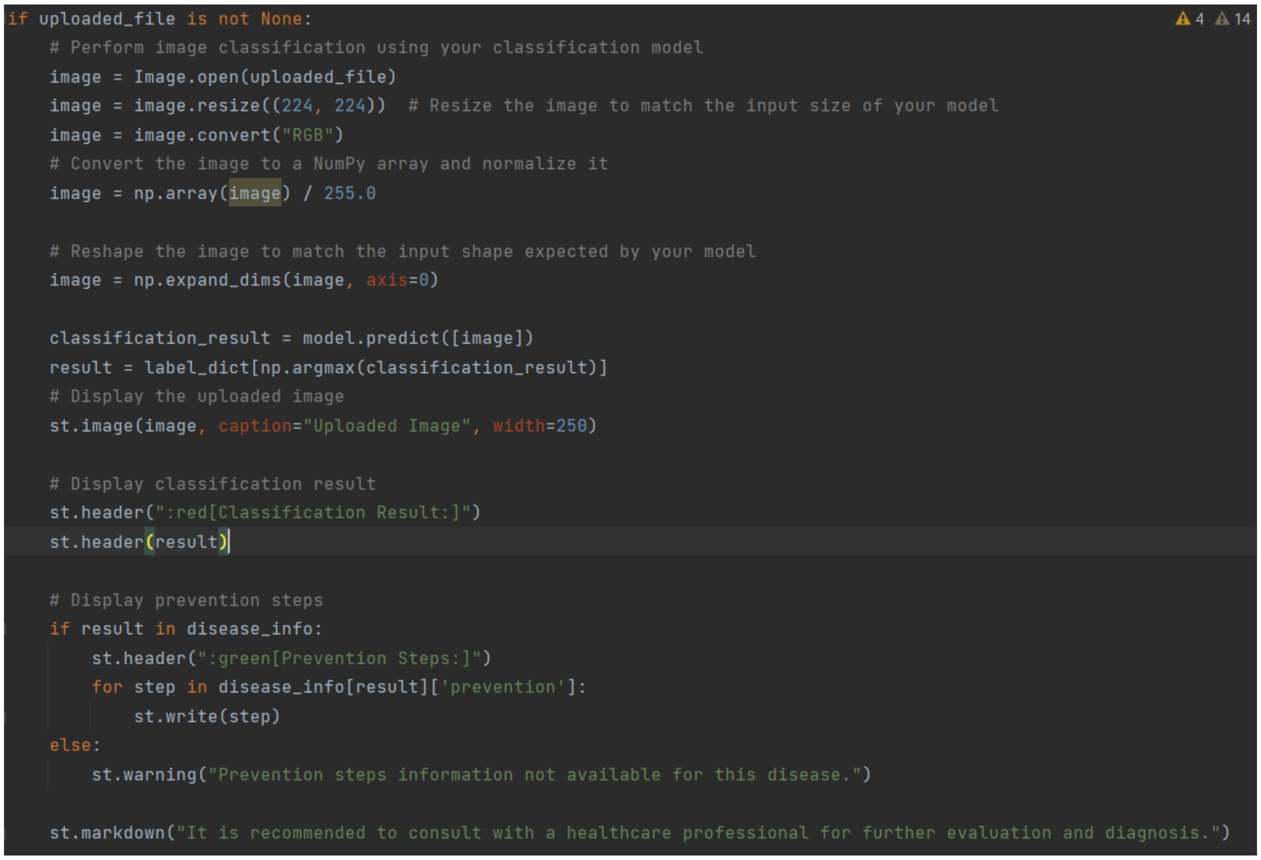
CSS FILE



```
/* Style inputs with type="text", select elements and textareas */
input[type=text], input[type=email], textarea {
    background-color: #cdcdcd;
    width: 100%; /* Full width */
    padding: 12px; /* Some padding */
    border: 3px solid #04AA6D; /* Gray border */
    border-radius: 4px; /* Rounded borders */
    box-sizing: border-box; /* Make sure that padding and width stays in place */
    margin-top: 16px; /* Add a top margin */
    margin-bottom: 26px; /* Bottom margin */
    resize: vertical /* Allow the user to vertically resize the textarea (not horizontally) */
  }

  /* Style the submit button with a specific background color etc */
  button[type=submit] {
    background-color: #04AA6D;
    color: white;
    padding: 12px 20px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
  }

  /* When moving the mouse over the submit button, add a darker green color */
  button[type=submit]:hover {
    background-color: #45a049;
  }
```

Once you click on the Predict in Nail Disease Home page, it will redirect you to Nail Disease Prediction page

Showcasing prediction on UI:



```
if uploaded_file is not None:
    # Perform image classification using your classification model
    image = Image.open(uploaded_file)
    image = image.resize((224, 224))  # Resize the image to match the input size of your model
    image = image.convert("RGB")
    # Convert the image to a NumPy array and normalize it
    image = np.array(image) / 255.0

    # Reshape the image to match the input shape expected by your model
    image = np.expand_dims(image, axis=0)

    classification_result = model.predict([image])
    result = label_dict[np.argmax(classification_result)]
    # Display the uploaded image
    st.image(image, caption="Uploaded Image", width=250)

    # Display classification result
    st.header(":red[Classification Result:]")
    st.header(result)

    # Display prevention steps
    if result in disease_info:
        st.header(":green[Prevention Steps:]")
        for step in disease_info[result]['prevention']:
            st.write(step)
    else:
        st.warning("Prevention steps information not available for this disease.")

    st.markdown("It is recommended to consult with a healthcare professional for further evaluation and diagnosis.")
```

Note: Target size need to change to 224,224 – Because vgg16 takes the input format in that size

Here we are using the uploaded file on our web app and making predictions. We are applying some image processing techniques and then sending that preprocessed image to the model for predicting the class. This returns the numerical value of a class (like 0, 1, 2, etc.). This numerical value is passed to the index variable declared. This returns the name of the class. This name is rendered to the result variable used.

Finally, run the application

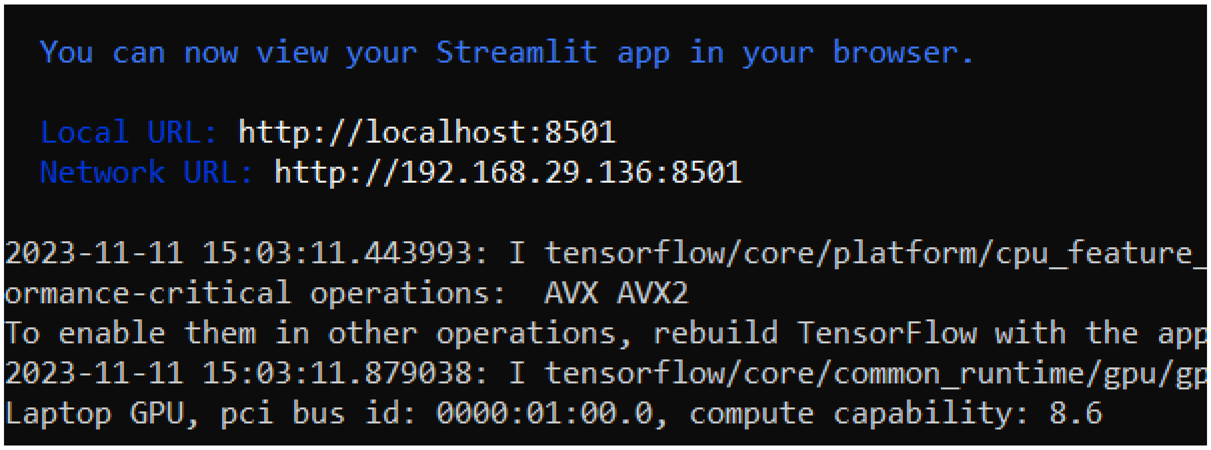
This is used to run the application in a localhost.

Activity 3: Run the application

- Open the command prompt from the start menu.
- Navigate to the folder where your Home.py resides.
- Now type "streamlit run Home.py" command.
- It will show the local host where your app is running on **http://192.168.29.136:8501**
- Copy that local host URL and open that URL in the browser. It does navigate me to where you can view your web page.

- Enter the values, click on the predict button and see the result/prediction on the web page.

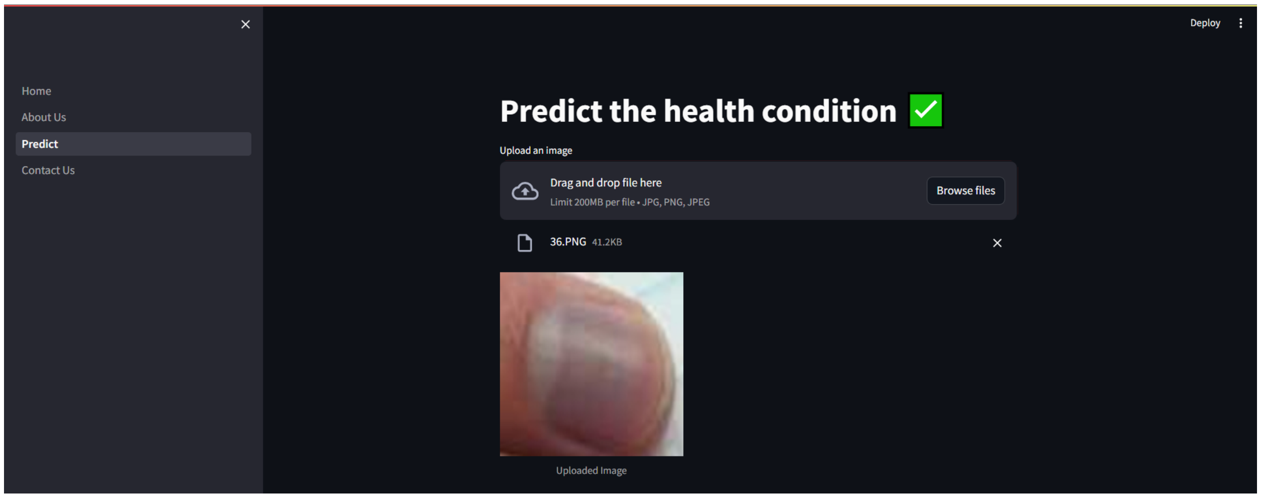
Then it will run on localhost: 8501

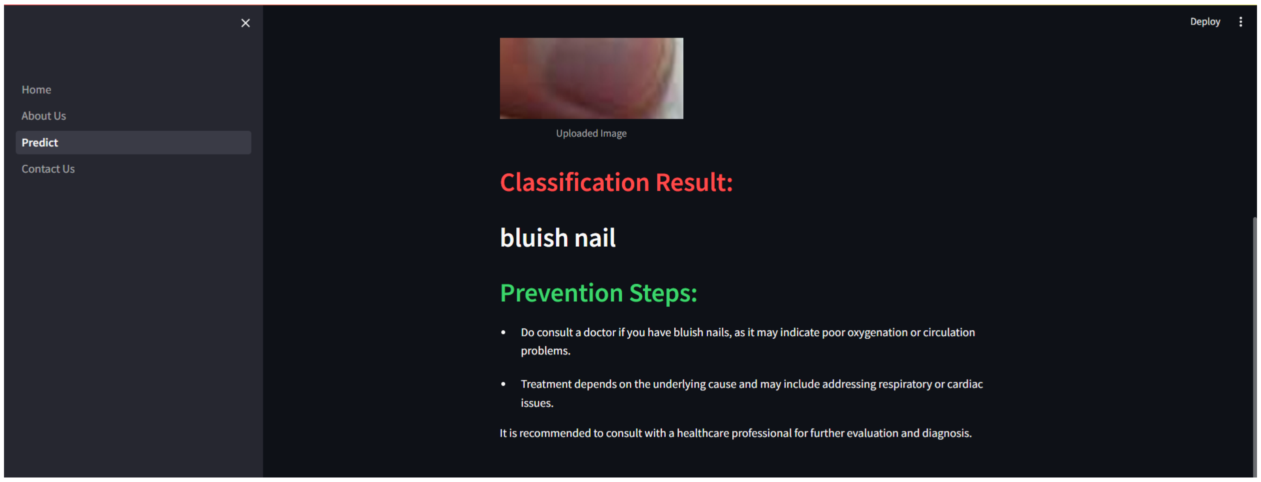


```
  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.29.136:8501

2023-11-11 15:03:11.443993: I tensorflow/core/platform/cpu_feature_
ormance-critical operations:  AVX AVX2
To enable them in other operations, rebuild TensorFlow with the app
2023-11-11 15:03:11.879038: I tensorflow/core/common_runtime/gpu/gp
Laptop GPU, pci bus id: 0000:01:00.0, compute capability: 8.6
```

Navigate to the localhost (http://192.168.29.136:8501) where you can view your web page.

FINAL OUTPUT:
Output 1





Classification Result:

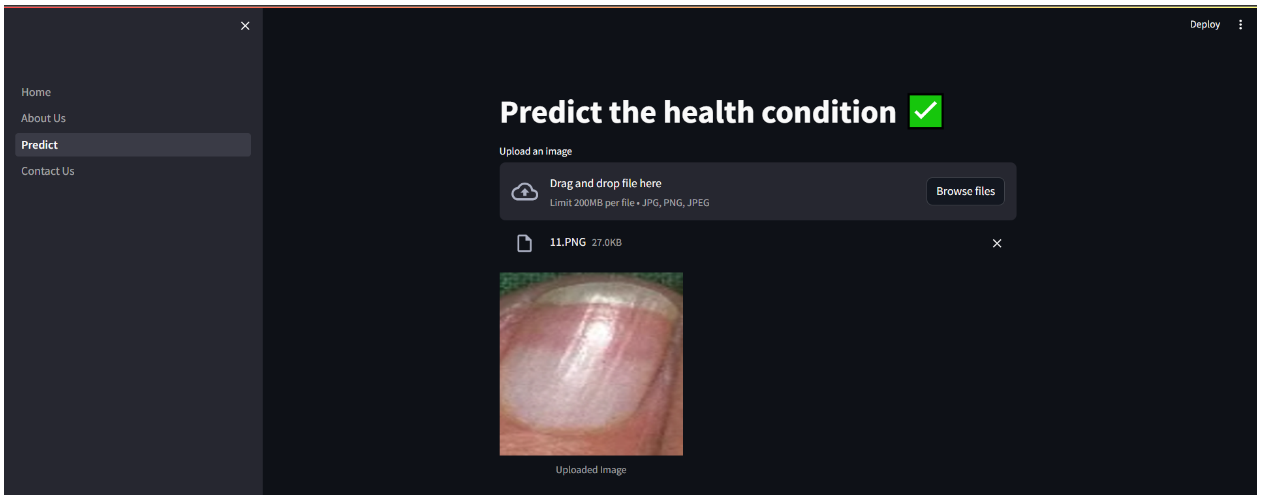
bluish nail

Prevention Steps:

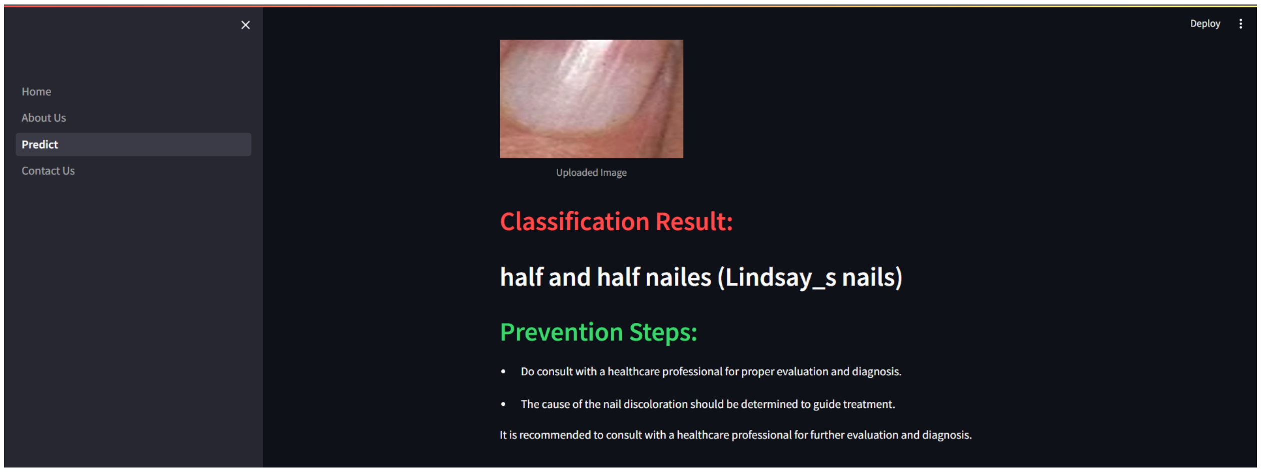
- Do consult a doctor if you have bluish nails, as it may indicate poor oxygenation or circulation problems.
- Treatment depends on the underlying cause and may include addressing respiratory or cardiac issues.

It is recommended to consult with a healthcare professional for further evaluation and diagnosis.

Output 2:



Predict the health condition ✅



Classification Result:

half and half nailes (Lindsay_s nails)

Prevention Steps:

- Do consult with a healthcare professional for proper evaluation and diagnosis.
- The cause of the nail discoloration should be determined to guide treatment.

It is recommended to consult with a healthcare professional for further evaluation and diagnosis.