Model Development Phase Template

Date	20-06-2025
Team ID	SWDTID1749906902
Project Title	Early Stage Disease Diagnosis System Using Human Nail Image Processing
Maximum Marks	4 Marks

Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots.

Initial Model Training Code:

```
from tensorflow.keras.layers import Dense, Flatten, Input
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img
from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input
from glob import glob
import numpy as np
import matplotlib.pyplot as plt
```

```
[ ] imageSize = 224
    #adding preprocessing layer to the front of vgg
    vgg = VGG16(input_shape=(imageSize, imageSize, 3), weights='imagenet', include_top=False)
```

```
[ ]  #don't train existing weights
     for layer in vgg.layers:
       layer.trainable = False
```

```
[ ]  #our layers - you can add more if you want
     x = Flatten()(vgg.output)
```

```
[ ]  prediction = Dense(17, activation='softmax')(x)
```

```
[ ]  model = Model(inputs=vgg.input, outputs=prediction)
```

```
[ ]  #viewing the structure of the model
     model.summary()
```

```
[ ]  model.compile(
         loss='categorical_crossentropy',
         optimizer='adam',
         metrics=['accuracy']
     )
```

```
[ ]  from tensorflow.keras.preprocessing.image import ImageDataGenerator

     train_datagen = ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True)
     test_datagen = ImageDataGenerator(rescale=1./255)
```

```
     import os

     # Assuming your training data is in a 'train' directory and testing data is in a 'test' directory
     trainPath = '/content/drive/MyDrive/Early Stage Disease Diagnosis System/train'
     testPath = '/content/drive/MyDrive/Early Stage Disease Diagnosis System/test'

     # Define image size
     imageSize = 224

     training_set = train_datagen.flow_from_directory(trainPath,
                                                      target_size = (imageSize, imageSize),
                                                      batch_size = 32,
                                                      class_mode = 'categorical')
     test_set = test_datagen.flow_from_directory(testPath,
                                                 target_size = (imageSize, imageSize),
                                                 batch_size = 32,
                                                 class_mode = 'categorical')
```

```
[ ]  training_set.class_indices
```

```
import sys
# fit the model
r = model.fit(
  training_set,
  validation_data=test_set,
  epochs=100,
  steps_per_epoch=len(training_set)//3,
  validation_steps=len(test_set)//3
)
```

```
#saving the model
model.save('vgg-16-nail-disease.h5')
```

```
#import load_model class for loading h5 file
from tensorflow.keras.models import load_model

#import image class to process the images
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.inception_v3 import preprocess_input
import numpy as np
```

```
#loading saved model file
model = load_model('vgg-16-nail-disease.h5')
```

```
#loading one random image
img = image.load_img(r'/content/drive/MyDrive/Early Stage Disease Diagnosis System/test/Darier_s disease/45.PNG', target_size=(224, 224))
```

```
#converting the image to array format
x=image.img_to_array(img)
x = np.expand_dims(x, axis=0)
img_data = preprocess_input(x)
```

```
predictions = model.predict(img_data)
```

```
index = ['Darier_s disease', 'Muehrck-e_s lines', 'alopecia areata', 'beau_s lines', 'bluish nail',
         'clubbing', 'eczema', 'half and half nailes (Lindsay_s nails)', 'koilonychia', 'leukonychia',
         'onycholysis', 'pale nail', 'red lunula', 'splinter hemmorrage', 'terry_s nail', 'white nail', 'yellow nail']

# Get predicted class index and label
output_index = np.argmax(predictions, axis=1)[0]
result = index[output_index]

print("Predicted Class:", result)
```

Model Validation and Evaluation Report:

Model	Classification Report	F1 Score	Confusion Matrix
TensorFlow / Keras	-	81%	-

VGG16 Model	-	79%	-
Flask	-	64%	-
flask_cors.CORS	-	78%	-