SKIN CANCER DETECTION WEB APPLICATION

A Mini Project Report Submitted
In partial 1ulfilment of the requirement for the award of the degree of

Bachelor of Technology in Computer Science and Engineering

(Artificial Intelligence and Machine Learning)

by

M. Samuel Cyril Raj	20N31A6639
P.V. Sai Praneeth Reddy	20N31A6644

Under the Guidance of

Dr. W Jaishri

Professor, Head of the department
Computational Intelligence MRCET

Department of Computational Intelligence

Malla Reddy College of Engineering & Technology
(Autonomous Institution- UGC, Govt. of India)
JNTUH, Hyderabad, Approved by AICTE, NBA &NAAC with 'A' Grade) Maisammaguda, Kompally,Dhulapally, Secunderabad – 500014
website: www.mrcet.ac.in

2020-2024

DECLARATION

We hereby declare that the project entitled **"SKIN CANCER DETECTION WEB APPLICATION"** submitted to Malla Reddy College of Engineering and Technology**,** affiliated t**o** Jawaharlal Nehru Technological University Hyderabad (JNTUH) for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a result of original research work done by me .It is further declared that the project report or any part thereof has not been previously submitted to any University or Institute for the award of degree or diploma**.**

Student Name1 – M. Samuel Cyril Raj(20N31A6639)

Student Name2 – P.V. Sai Praneeth Reddy(20N31A6644)

Malla Reddy College of Engineering & Technology

(Autonomous Institution- UGC, Govt. of India)

(Affiliated to JNTUH, Hyderabad, Approved by AICTE, NBA &NAAC with 'A' Grade)

Maisammaguda, Kompally, Dhulapally, Secunderabad – 500100 website: www.mrcet.ac.in

CERTIFICATE

This is to certify that this is the bonafide record of the project titled **"SKIN CANCER DETECTION WEB APPLICATION"** submitted by M. Samuel Cyril Raj(20N31A6639), P. V. Sai Praneeth Reddy(20N31A6644) of B.Tech in the partial 3ulfilment of the requirements for the degree of Bachelor of Technology in Computer Science and Engineering, Department of Computational Intelligence during the year 2023-2024 . The results embodied in this project report have not been submitted to any other university or institute for the award of any degree or diploma.

Guide Name **Dr. W Jaishri**
Designation **Professor**

INTERNAL GUIDE **HEAD OF THE DEPARTMENT**

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We feel honored and privileged to place our warm salutation to our college Malla Reddy College of Engineering and technology (UGC-Autonomous), our Director ***Dr. VSK Reddy*** who gave us the opportunity to have experience in engineering and profound technical knowledge.

We are indebted to our Principal ***Dr. S. Srinivasa Rao*** for providing us with facilities to do our project and his constant encouragement and moral support which motivated us to move forward with the project.

We would like to express our gratitude to our Head of the Department ***Dr. D. Sujatha*** for encouraging us in every aspect of our system development and helping us realize our full potential.

We would like to thank our application development guide as well as our internal guide ***Dr. W Jaishri*** for her structured guidance and never-ending encouragement. We are extremely grateful for valuable suggestions and unflinching co-operation throughout application development work.

We would also like to thank all supporting staff of department of CI and all other departments who have been helpful directly or indirectly in making our application development a success.

We would like to thank our parents and friends who have helped us with their valuable suggestions and support has been very helpful in various phases of the completion of the application development.

By

Student Name1 – M. Samuel Cyril Raj(20N31A6639)

Student Name2 – P. V Sai Praneeth Reddy(20N31A6644)

ABSTRACT

Skin cancer is also one of the most common types of cancer in India, with over 1 million cases diagnosed each year. A web application for skin cancer detection can provide a valuable tool for improving early detection of skin cancer.

This web application uses deep learning to classify skin lesions into different types of skin cancer, including melanoma, basal cell carcinoma, and squamous cell carcinoma.

The application is trained on a large dataset of skin lesion images, and it has been shown to achieve high accuracy in detecting skin cancer. To use the application, users simply need to upload a photo of their skin lesion.

The application will then analyze the image and provide a classification result. It is easy to use and accessible to anyone with an internet connection. The application can be used by individuals to check their own skin for suspicious lesions.

TABLE OF CONTENTS

S. No. Topic Page No.

1.INTRODUCTION

This chapter gives an overview about the purpose, aim, objectives, background and operation environment of the system.

1.1 PURPOSE:

This project may be a method for the detection of Melanoma carcinoma using Image processing tools. The proposed system is a skin cancer detection web application that uses deep learning to classify skin lesions into different types of skin cancer, including melanoma. In this input the system is that skin lesion image then applying image processing echniques, it analyses conclude about the presence carcinoma. In Lesion to Image analysis tools checks in the varied Melanoma parameters, Color, Area. Perimeter, diameter etc texture, size and shape analysis for image segmentation and the feature stages. The extracted to feature parameters wont of classify the image as Non Melanoma and Melanoma cancer lesion. Through poll we are getting to collect patient after treatment.

1.2 BACKGROUND OF PROJECT:

The background of WhatsApp sentiment analysis projects can vary depending on the specific context and goals of the project. However, generally speaking, WhatsApp sentiment analysis is driven by the need to gain insights into the emotions, opinions, and attitudes of people communicating on the WhatsApp platform.

1.3 SCOPE OF PROJECT:

The product scope of a WhatsApp sentimental analysis project would depend on the objectives of the project and the stakeholders involved. However, here are some potential aspects that could be included in the product scope: Data collection and storage, Preprocessing and cleaning, Sentiment analysis techniques, Visualization and reporting, Validation and verification.

1.4 PROJECT FEATURES:

- Easy to understand .
- Easy to use and implement.
- Recognises the image and clears out the blurred out images or video.
- It makes it easier for any human beings to understand and make use of it.
- Gives a clear image with enhanced colors too.

1.5 MODULES DESCRIPTION:

This project is composed of the following modules:

- **Image preprocessing:** The code preprocesses the images by rescaling them to a common size of 100x100 pixels and normalizing them by dividing each pixel by 255.
- **Data augmentation:** The code applies data augmentation to the training images. This involves randomly flipping, rotating, and zooming the images.
- **Model training:** The code trains a convolutional neural network (CNN) model on the preprocessed training images. The CNN model is trained to predict the class of each image.
- **Model evaluation:** The code evaluates the performance of the trained model on the test set. The model's accuracy on the test set is used to measure its performance on unseen data.
- **Model prediction:** The code predicts the class of new images using the trained model.

2.SYSTEM REQUIREMENTS

2.1.1 HARDWARE REQUIREMENTS:

- Computer Desktop or Laptop
- System will be using Processor: Core2Duo
- Main Memory: 4 GB RAM (Minimum)
- Hard Disk: 512 GB (Minimum)

2.2 SOFTWARE REQUIREMENT SPECIFICATION:

2.2.1 TensorFlow

Used to train and evaluate the CNN model. TensorFlow is a high-level machine learning library that provides a variety of tools for building and training machine learning models. In the code you provided, TensorFlow is used to train and evaluate a CNN model for skin cancer classification.

2.2.2 Matplotlib.pyplot

Used to plot the training and validation accuracy and loss. Matplotlib.pyplot is a plotting library for Python and NumPy that allows you to create and display a variety of charts and graphs. In the code you provided, Matplotlib.pyplot is used to plot the training and validation accuracy and loss of the CNN model. This helps to visualize the performance of the model during training and to identify any potential problems.

2.2.3 Numpy

Task: Used to preprocess the images and convert them to NumPy arrays. Numpy is a library for scientific computing with Python that provides a variety of tools for working with numerical data. In the code you provided, Numpy is used to preprocess the images by resizing them to a common size and converting them to the correct data type. Numpy is also used to convert the preprocessed images to NumPy arrays, which makes them easier to process with TensorFlow.

2.2.4 Pandas

Used to store and manipulate the image data. Pandas is a library for data analysis and manipulation that provides a variety of tools for working with data in tabular form. In the code you provided, Pandas is used to store and manipulate the image data, such as the image paths and labels. This makes it easier to load and preprocess the image data, and to track the performance of the model on different subsets of the data.

2.2.5 Os

Used to load the image data from the disk. The os module provides a variety of functions for interacting with the operating system. In the code you provided, the os module is used to load the image data from the disk. This involves opening the image files and reading their contents into memory.

2.2.6 PIL

Used to resize and convert the images to the correct format. PIL is a library for image processing that provides a variety of tools for working with images. In the code you provided, PIL is used to resize and convert the images to the correct format for input to the CNN model.

2.2.7 Glob

Used to find all of the image files in the training and test directories. The glob module provides a variety of functions for finding files that match a specified pattern. In the code you provided, the glob module is used to find all of the image files in the training and test directories. This makes it easier to load and preprocess the image data.

2.2.8 Keras

Used to build and train the CNN model. Keras is a high-level deep learning API built on top of TensorFlow that makes it easier to build and train machine learning models. In the code you provided, Keras is used to build and train a CNN model for skin cancer classification.

2.2.9 **Pathlib**

Used to create and manipulate file paths. The pathlib module provides a variety of functions for object-oriented path manipulation. In the code you provided, the pathlib module is used to create and manipulate file paths to the image data.

2.2.10 Json

Used to save and load the model parameters. The json module provides a variety of functions for parsing and manipulating JSON data. In the code you provided, the json module is used to save and load the parameters of the trained CNN model. This makes it possible to reuse the trained model at a later time.

2.2.11 Joblib

Used to save and load the model. The joblib module provides a variety of tools for saving and loading Python objects. In the code you provided, the joblib module is used to save and load the trained CNN model. This makes it easy to share the trained model with others or to use it in other Python applications.

2.2.12 Gradio

Used to create a user interface for the model. Gradio is a Python library that makes it easy to create and deploy machine learning models as web applications. In the code you provided, Gradio is used to create a user interface that allows users to upload images and receive predictions from the model.

2.2.13 OpenCV

Used to convert the grayscale input image to color. OpenCV is a library for computer vision and image processing. In the code you provided, OpenCV is used to convert the grayscale input image to color (RGB) because the model was trained on RGB images.

2.3 SCOPE:

The scope of the project is to develop a web application that allows users to upload images and receive predictions for the presence of skin cancer. The application is powered by a trained CNN model that has been trained on a dataset of skin cancer images.

It can help people to identify potential skin cancer lesions early, which can lead to better treatment outcomes. It can reduce the need for people to visit a dermatologist, which can save time and money.

It can help to raise awareness of skin cancer and encourage people to get regular skin screenings.

2.4 EXISTING SYSTEM:

1. **Dermoscopy**

   Dermoscopy is a non-invasive procedure that uses a specialized magnifying lens to examine the skin. It is used to diagnose a variety of skin conditions, including skin cancer.

   Drawbacks of dermoscopy:

- It can be difficult to interpret dermoscopy images, especially for inexperienced clinicians.
- It is not very accurate in the early stages of skin cancer.
- It is not always possible to distinguish between benign and malignant lesions using dermoscopy alone.

2. **Biopsy**

   A biopsy is a procedure in which a small sample of tissue is removed from the body and examined under a microscope to determine if it is cancerous.

   Drawbacks of biopsy:

- It is an invasive procedure that can cause pain and bleeding.
- It can be difficult to biopsy lesions in difficult-to-reach areas.
- There is a small risk of infection or scarring.

3. **Imaging tests**

   Imaging tests, such as ultrasound, MRI, and CT scans, can be used to visualize skin lesions and assess their depth and extent.

   Drawbacks of imaging tests:

- They can be expensive and time-consuming.
- They are not always accurate in diagnosing skin cancer, especially in the early stages.
- They can expose the patient to radiation.

2.5 PROPOSED SYSTEM:

- This project may be a method for the detection of Melanoma carcinoma using Image processing tools.
- The proposed system is a skin cancer detection web application that uses deep learning to classify skin lesions into different types of skin cancer, including melanoma.
- In this input the system is that skin lesion image then applying image processing echniques, it analyses conclude about the presence carcinoma .
- In Lesion to Image analysis tools checks in the varied Melanoma parameters, Color, Area
- perimeter, diameter etc texture, size and shape analysis for image segmentation and the feature stages.

3. TECHNOLOGIES USED:

3.1 PYTHON: Python is an interpreted, object-oriented, high level programming language with dynamic semantics developed by Guido Van Rossum. It was originally released in 1991. Besides web and software development , python is used for data analytics , machine learning, and even design. It is widely considered among the easiest programming languages for beginners to learn.

3.2 Machine learning : Machine Learning is the field of study that gives computers the capability to learn without being explicitly programmed. ML is one of the most exciting technologies that one would have ever come across. As it is evident from the name, it gives the computer that makes it more similar to humans: The ability to learn. Machine learning is actively being used today, perhaps in many more places than one would expect.

3.3 Convolutional neural networks (CNNs): CNNs are a type of deep learning model that is well-suited for image classification tasks. The CNN model in your project is trained on a dataset of skin cancer images to learn to identify potential skin cancer lesions. CNNs work by extracting features from images and then using these features to classify the images.

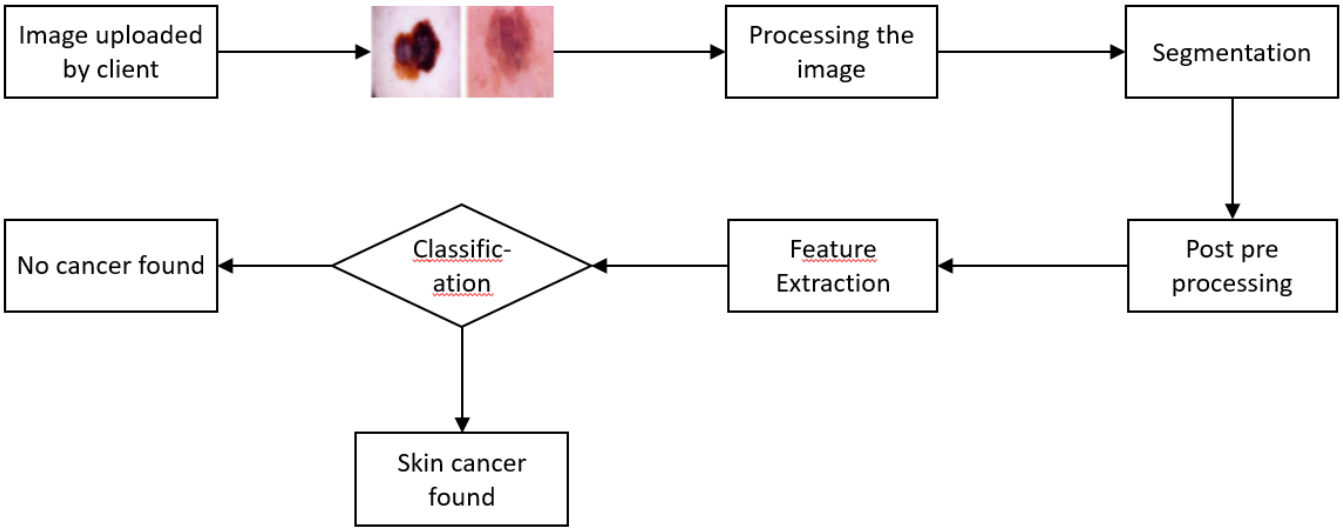
3.4 TensorFlow: TensorFlow is a popular machine learning library that is used to train and deploy CNN models. TensorFlow provides a variety of tools for building and training machine learning models, including CNNs. TensorFlow is also used to deploy machine learning models as web applications.

3.5 Gradio: Gradio is a Python library that makes it easy to create and deploy machine learning models as web applications. Gradio is used to create the user interface for the skin cancer detection system. The Gradio user interface allows users to upload images and receive predictions from the CNN model.

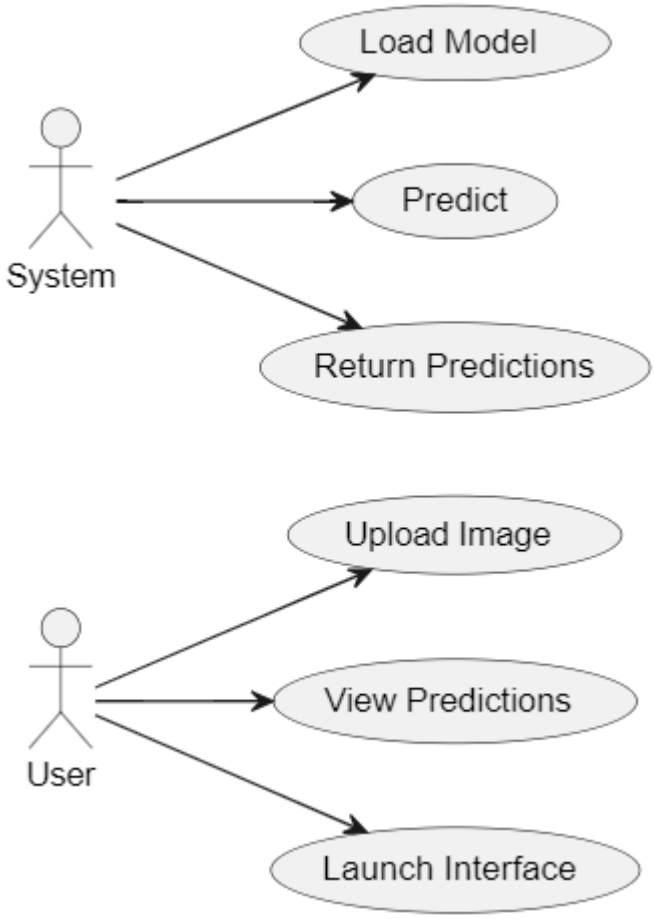
3.6 OpenCV: OpenCV is a Python library for computer vision and image processing. OpenCV is used in the project to convert the grayscale input image to color and to resize the input image to match the model input shape. OpenCV also provides a variety of other tools for image processing, such as filtering and edge detection.

4.SYSTEM DESIGN AND UML DIAGRAMS

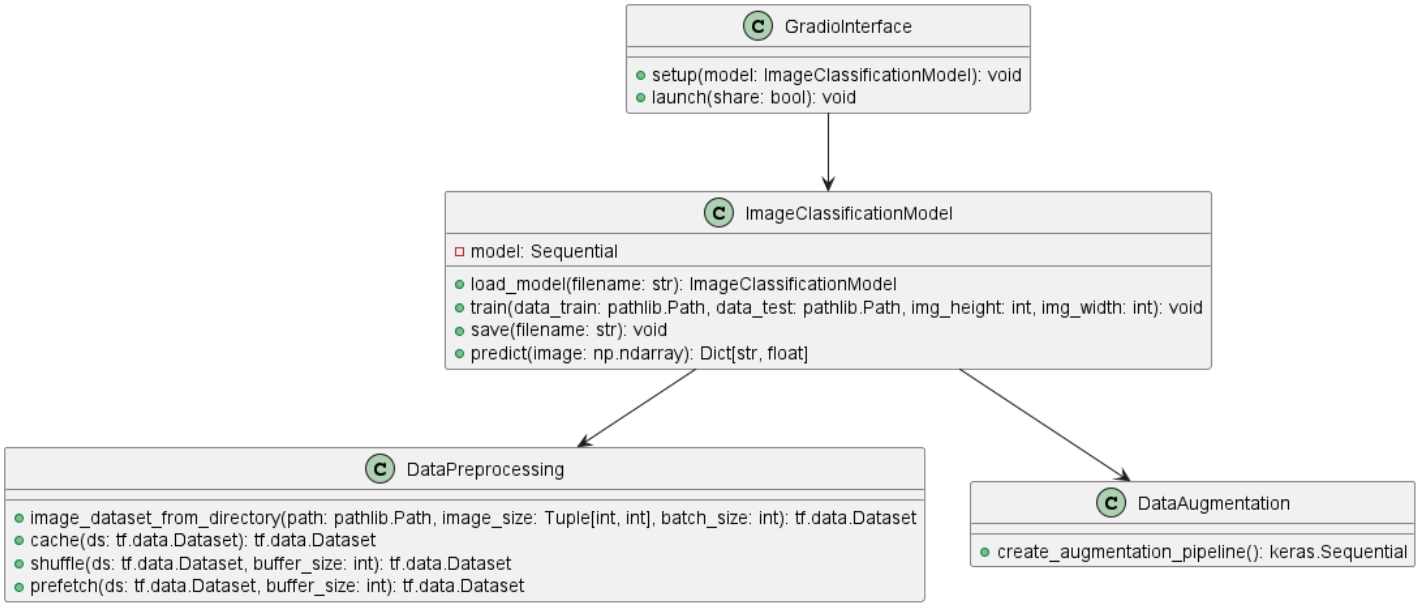
4.1 SYSTEM ARCHITECTURE



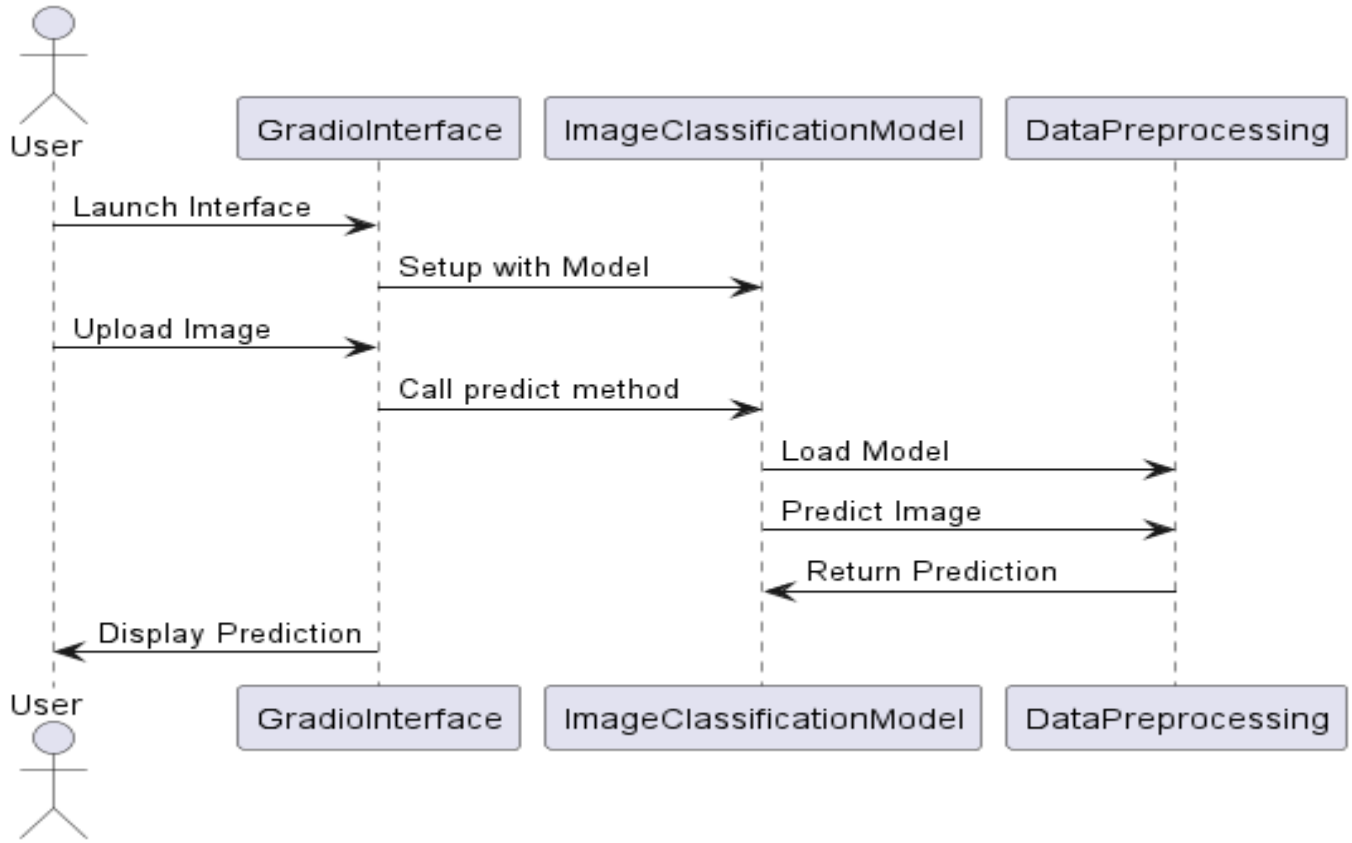
4.2 USECASE DIAGRAM



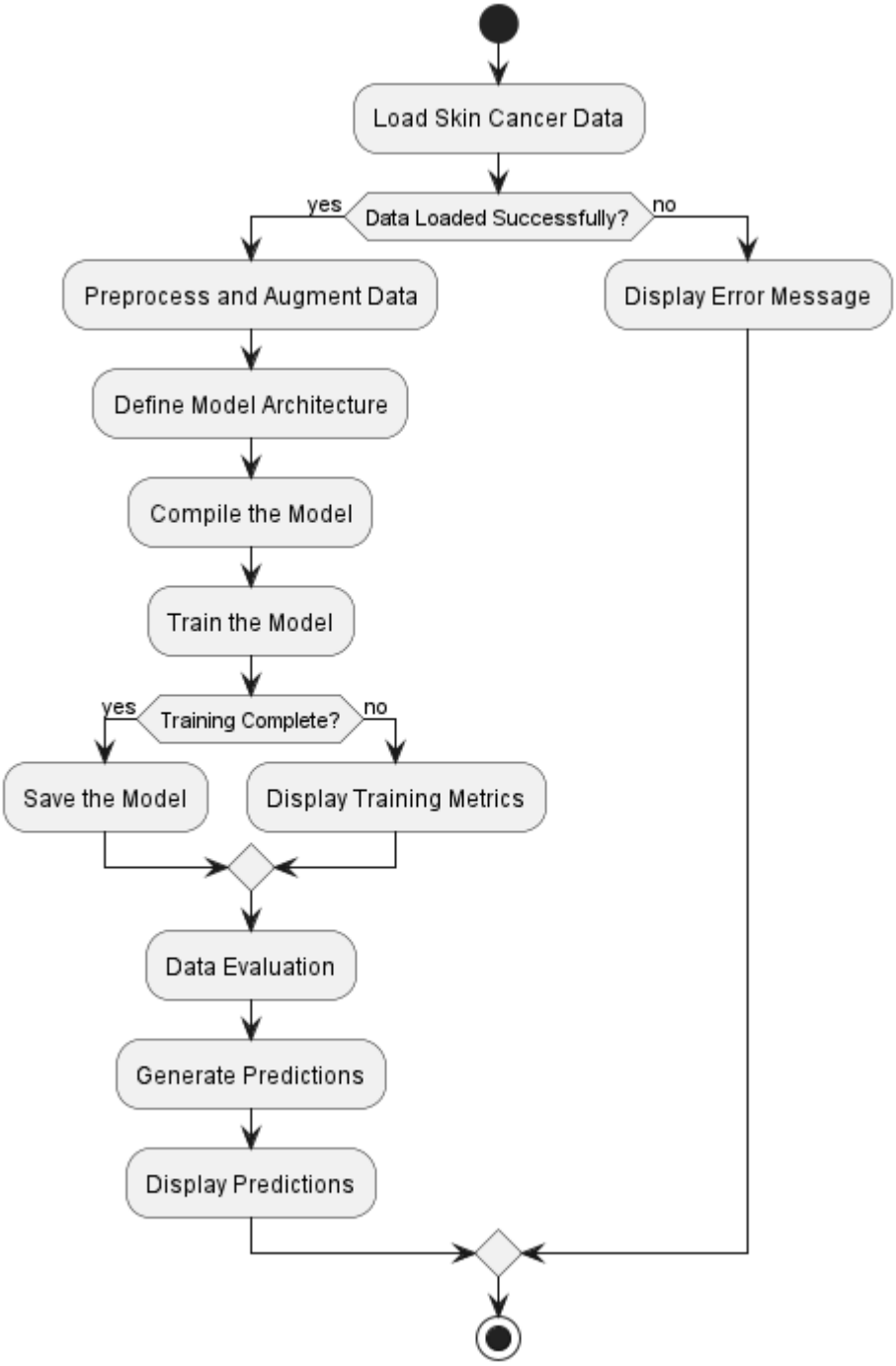
4.2.1 CLASS DIAGRAM



4.2.2 SEQUENCE DIAGRAM

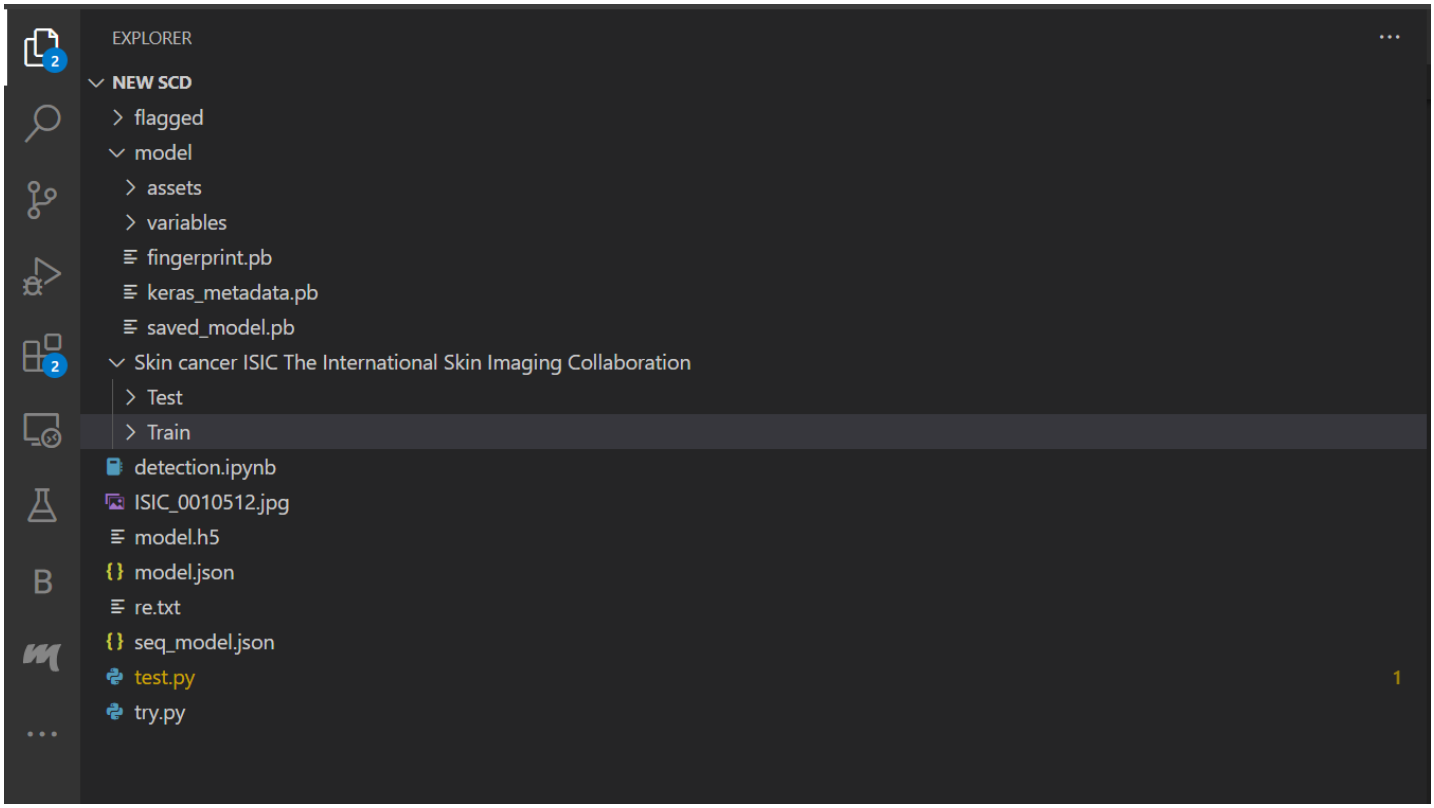


4.2.3 ACTIVITY DIAGRAM



5.IMPLEMENTATION

5.1: CODE



5.1.1 detection.ipynb

```
import pathlib
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import os
import PIL
import glob
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
```

[ ]

```
data_train = pathlib.Path("Skin cancer ISIC The International Skin Imaging Collaboration\Train")
data_test = pathlib.Path("Skin cancer ISIC The International Skin Imaging Collaboration\Test")
```

[ ]

```
image_count_train = len(list(data_train.glob('*/*.jpg')))
print(image_count_train)
image_count_test = len(list(data_test.glob('*/*.jpg')))
print(image_count_test)
```

[ ]

```
batch_size = 32
img_height = 100
img_width = 100
```

[ ]

```
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_train,
    seed=123,
    validation_split = 0.2,
    subset='training',
    image_size=(img_height, img_width),
    batch_size=(batch_size))
```

[ ]

```
import tensorflow as tf
test_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_test,
    seed=123,
    validation_split = 0.2,
    subset='training',
    image_size=(img_height, img_width),
    batch_size=(batch_size))
```

[ ]

```
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
  data_train,
  seed=123
  valid Loading... t = 0.2,
  subset='validation',
  image_size=(img_height, img_width),
  batch_size=(batch_size))
```

[ ]

```
class_names = train_ds.class_names
print(class_names)
```

[ ]

```
import matplotlib.pyplot as plt

### your code goes here, you can use training or validation data to visualize

plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(2):
  for i in range(9):
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(images[i].numpy().astype("uint8"))
    plt.title(class_names[labels[i]])
    plt.axis("off")
```

[ ]

```
(variable) val_ds: Any
imental.AUTOTUNE
e().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
```

[ ]

```
num_classes = 9 # As target class has 9 labels

model = Sequential([
  layers.experimental.preprocessing.Rescaling(1./255, input_shape=(img_height, img_width, 3)),
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])
```

[ ]

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

[ ]

```
model.summary()
```

[ ]

```
epochs = 25
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)
```

[ ]

```
model.save('model.h5')
```

[ ]

```
import json
from joblib import dump, load

# convert the model to a JSON object
model_json = model.to_json()
# save the JSON object to a file

with open("model.json", "w") as json_file:
    json.dump(model_json, json_file)

    #save model
dump(model_json, 'seq_model.json')
```

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```

[ ] Python

```
data_augmentation = keras.Sequential([
                                      layers.experimental.preprocessing.RandomFlip("horizontal", input_shape=(img_height, img_width, 3)),
                                      layers.experimental.preprocessing.RandomRotation(0.1),
                                      layers.experimental.preprocessing.RandomZoom(0.1),
])
```

[ ] Python

```
plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(3):
  for i in range(9):
    augmented_images = data_augmentation(images)
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(augmented_images[0].numpy().astype("uint8"))
    plt.axis("off")
```

[ ]

```
model = Sequential([
  data_augmentation,
  layers.experimental.preprocessing.Rescaling(1./255),
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Dropout(0.2),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])
```

[ ]

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

```
plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(3):
  for i in range(9):
    augmented_images = data_augmentation(images)
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(augmented_images[0].numpy().astype("uint8"))
    plt.axis("off")
```

[ ]

```
model = Sequential([
  data_augmentation,
  layers.experimental.preprocessing.Rescaling(1./255),
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Dropout(0.2),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])
```

[ ]

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
model.summary()
```

```
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=25
)
```

[ ]

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```

[ ]

```
from glob import glob
path_list = [ x for x in glob(os.path.join(data_train, '*', '*.jpg')) ]
lesion_list = [ os.path.basename(os.path.dirname(y)) for y in glob(os.path.join(data_train, '*', '*.jpg')) ]
print(len(lesion_list))
```

[ ]

```
df_dict_original = dict(zip(path_list, lesion_list))
print(list(df_dict_original.items())[:2])
```

[ ]

```
original_df = pd.DataFrame(list(df_dict_original.items()), columns=['Path','Label'])
original_df.head()
```

[ ]

```
original_df[['Label']].value_counts()
```

[ ]

```
model.save('model')
```

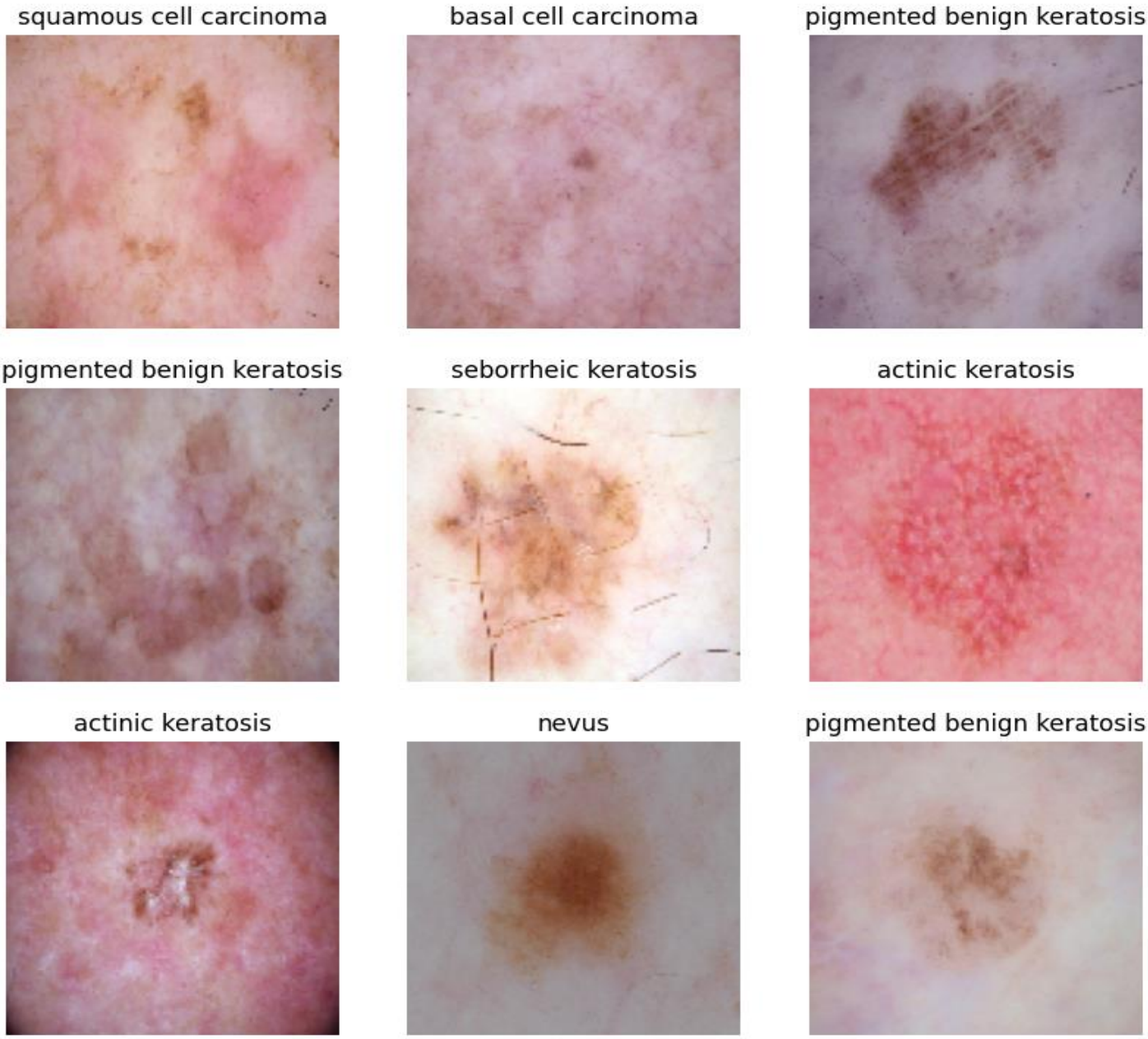
[ ]

5.1.2 test.py

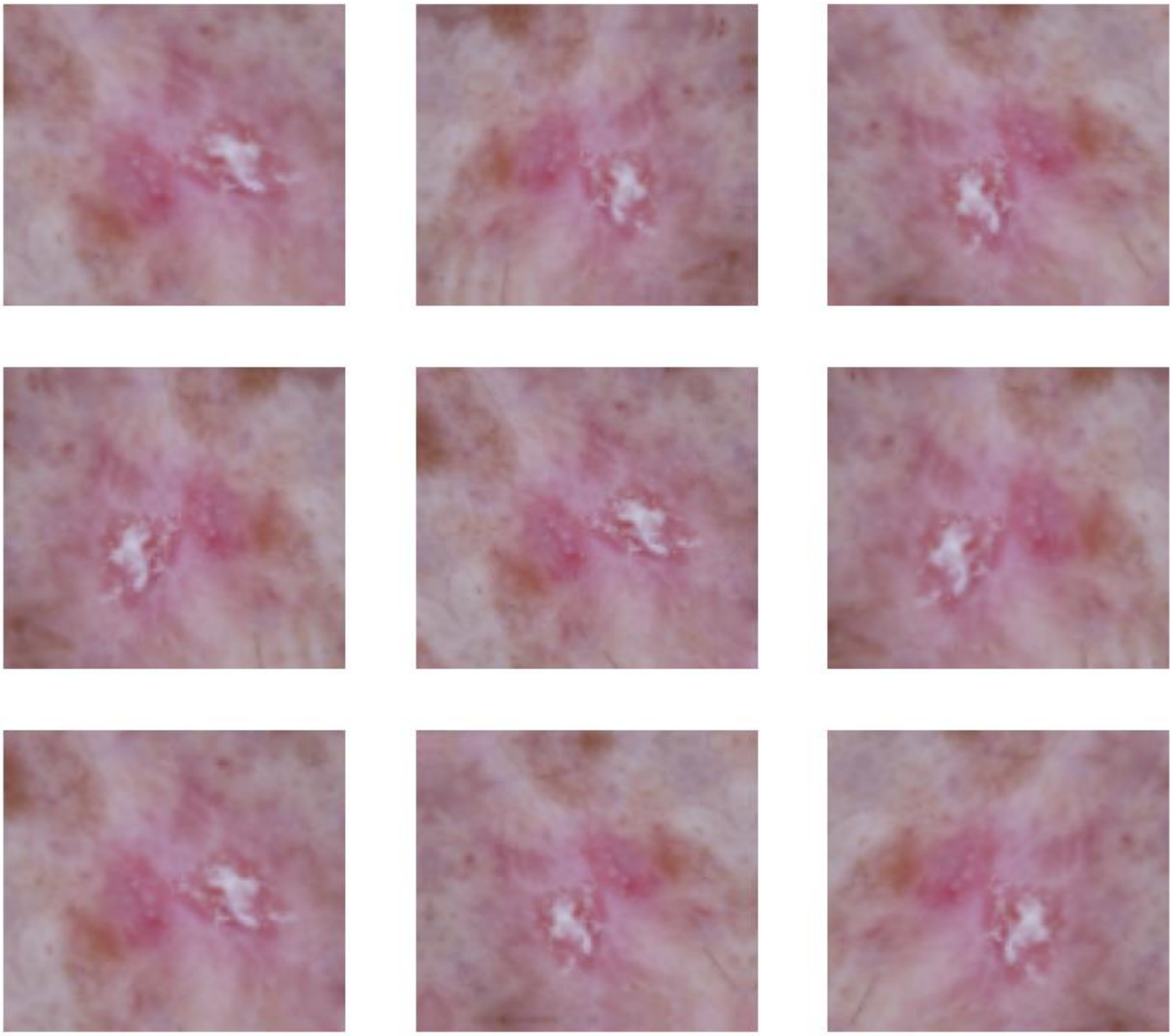
```python
import tensorflow as tf
import numpy as np
import gradio
import gradio as gr
#create a function to make predictions
#return a dictionary of labels and probabilities
model = tf.keras.models.load_model("model.h5")
#def cancer_predict(img):
    #img = img.reshape(1, 100, 100, 1)
    #prediction = model.predict(img).tolist()[0]
    #class_names = ["actinic keratosis","dermatofibroma","basal cell carcinoma",
                   #"melanoma","nevus","pigmented benign keratosis","seborrheic keratosis","vascular lesion","squamous cell carcinoma"]
    #return {class_names[i]: prediction[i] for i in range(2)}
import cv2
classname_value = 0
prediction_value = 0
def cancer_predict(img):
    # Convert grayscale image to color (RGB)
    img_rgb = cv2.cvtColor(img, cv2.COLOR_GRAY2RGB)
    img_rgb = img_rgb.reshape(1, 100, 100, 3)  # Reshape to match model input shape
    prediction = model.predict(img_rgb).tolist()[0]
    class_names = ["actinic keratosis","dermatofibroma","basal cell carcinoma",
                   "melanoma","nevus","pigmented benign keratosis","seborrheic keratosis","vascular lesion","squamous cell carcinoma"]
    return {class_names[i]: prediction[i] for i in range(2)}
    #for i in range(2):
        #classname_value += class_names[i]
        #prediction_value += prediction[i]
    #return{classname_value:prediction_value}

    #return {class_name: round(prob * 100, 2) for class_name, prob in zip(class_names, prediction[0])}

#set the user uploaded image as the input array
#match same shape as the input shape in the model
im = gradio.inputs.Image(shape=(100, 100), image_mode='L', invert_colors=False, source="upload")

#setup the interface
iface = gr.Interface(
    fn = cancer_predict,
    inputs = im,
    outputs = gradio.outputs.Label(),
)
iface.launch(share=True)

```

5.2: OUTPUT

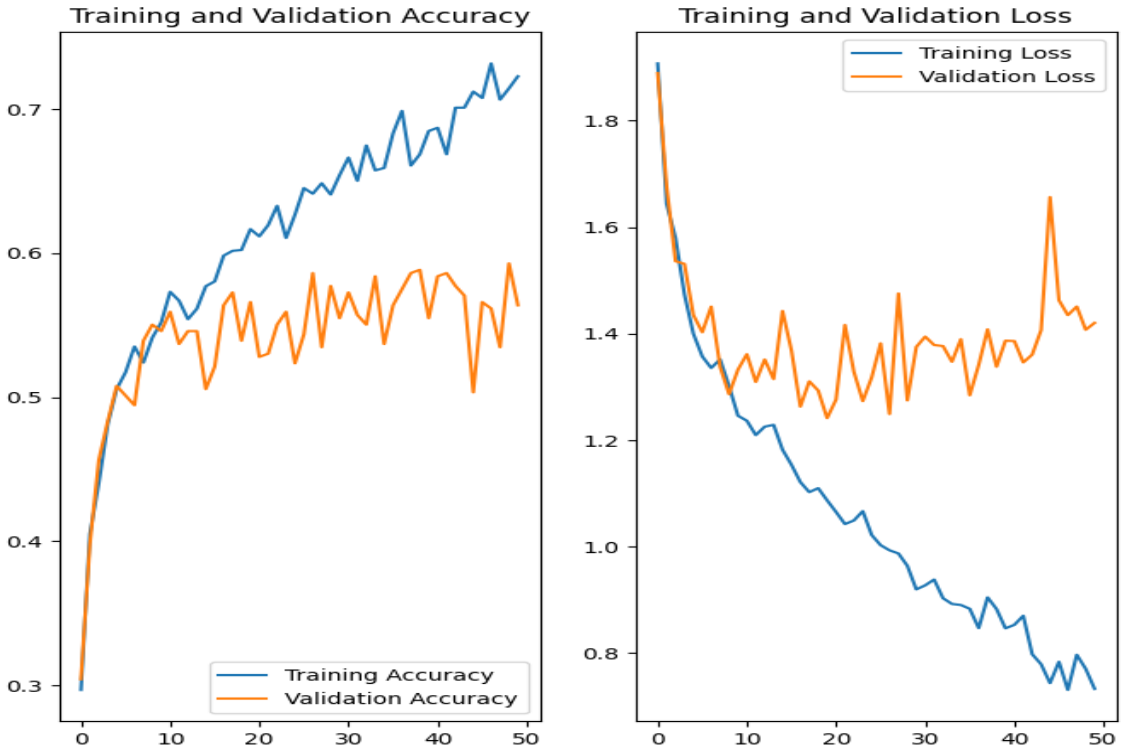
5.2.1 Output 1



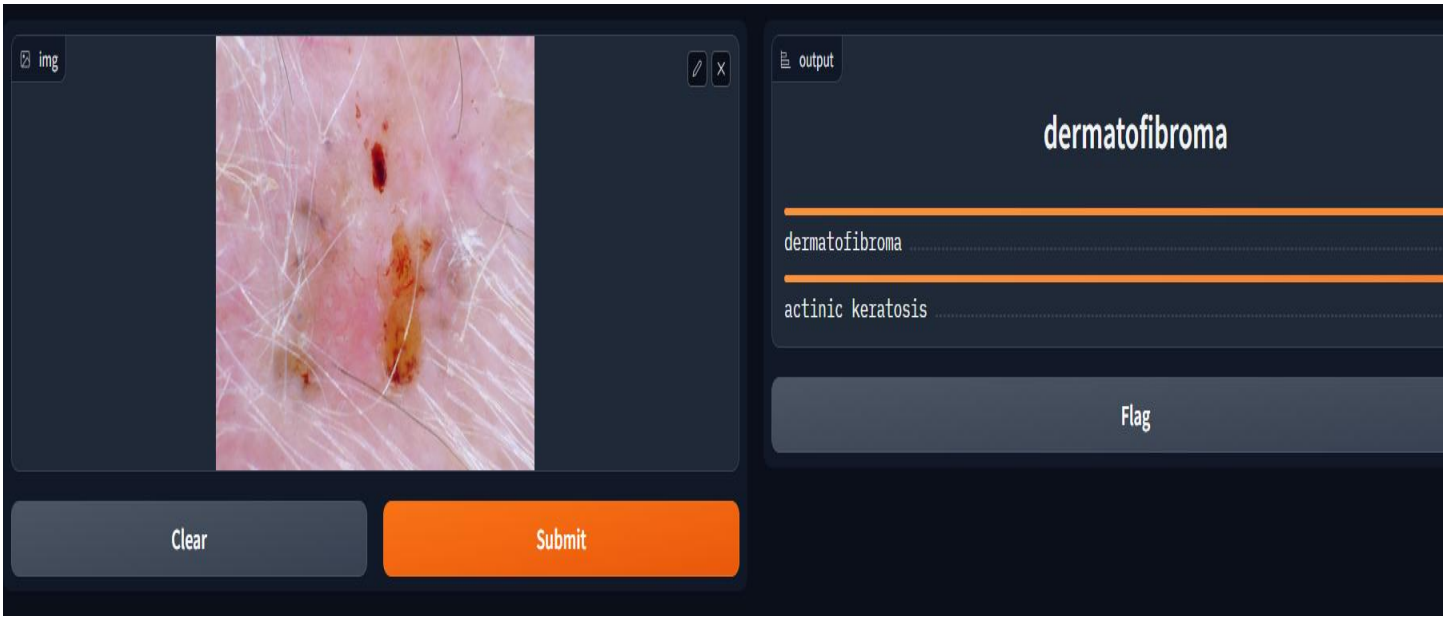
5.2.2 Output 2



5.2.3 Output 3



5.2.4 Output 4



6. CONCLUSION & FUTURE SCOPE

6.1 CONCLUSION:

The skin cancer detection using CNN web application is a promising new technology with the potential to revolutionize the way skin cancer is detected and treated. By making early detection more accessible and affordable, the application could help to save lives.

In addition to the benefits mentioned above, the application also has the potential to:

- Reduce the need for unnecessary biopsies, which can be expensive and painful.
- Improve the quality of life for skin cancer patients by providing them with early diagnosis and treatment.
- Help to reduce the overall cost of healthcare by reducing the need for expensive treatments for advanced skin cancer.

The application is still under development, but it has the potential to make a significant impact on the fight against skin cancer. Future research could focus on improving the accuracy of the CNN model, developing new features for the application, and integrating the application with other healthcare systems.

I believe that the skin cancer detection using CNN web application has the potential to become a standard tool for early detection of skin cancer. I am excited to see how the application develops in the future and how it is used to improve the lives of people with skin cancer.

6.2 FUTURE SCOPE:

The skin cancer detection using CNN web application could be improved in the following ways:

- The CNN model could be trained on a larger and more diverse dataset of images of skin lesions. This would improve the accuracy of the application.
- The application could be integrated with other healthcare systems, such as electronic health records (EHRs). This would allow users to easily share their results with their doctor.
- The application could be developed into a mobile app, making it even more accessible to users.

7.REFERENCES

https://ieeexplore.ieee.org/document/10063762/

https://github.com/charanhu/Skin_Cancer_Detection_MNIST

https://www.kaggle.com/competitions/siim-isic-melanoma-classification