National College of Ireland

Configuration Manual

MSc Research Project
Programme Name

Pooja Krishnamoorthy
Student ID: X23131179

School of Computing
National College of Ireland

Supervisor: Dr Muslim Syed Jammel

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Pooja Krishnamoorthy
Student ID:	X23131179
Programme:	MSc Data Analytics **Year:** 2023-2024
Module:	Configuration Manual
Supervisor Submission :	Dr Muslim Jammel Syed
Due Date:	12/08/2024
Project Title:	Personalized Skincare Recommendations Using Multi-Modal Deep Learning Techniques
Word Count:	555 **Page Count** 12

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Pooja Krishnamoorthy

Date: 12/08/2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Pooja Krishnamoorthy
Student ID: X23131179

1 System Configuration:

Below is the device specification of a machine in which the project was executed.

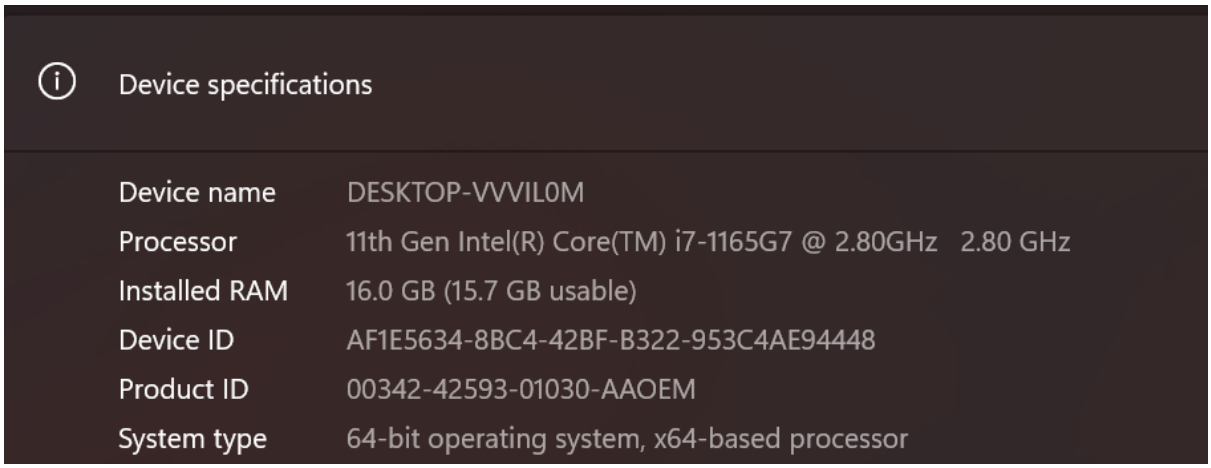

Figure 1

2 Software Requirement:

Below is the software that should be installed.

1. Jupyter Notebook
2. Python 3.9

3 Code Artifacts:

Below are the code artifacts which has to be downloaded.

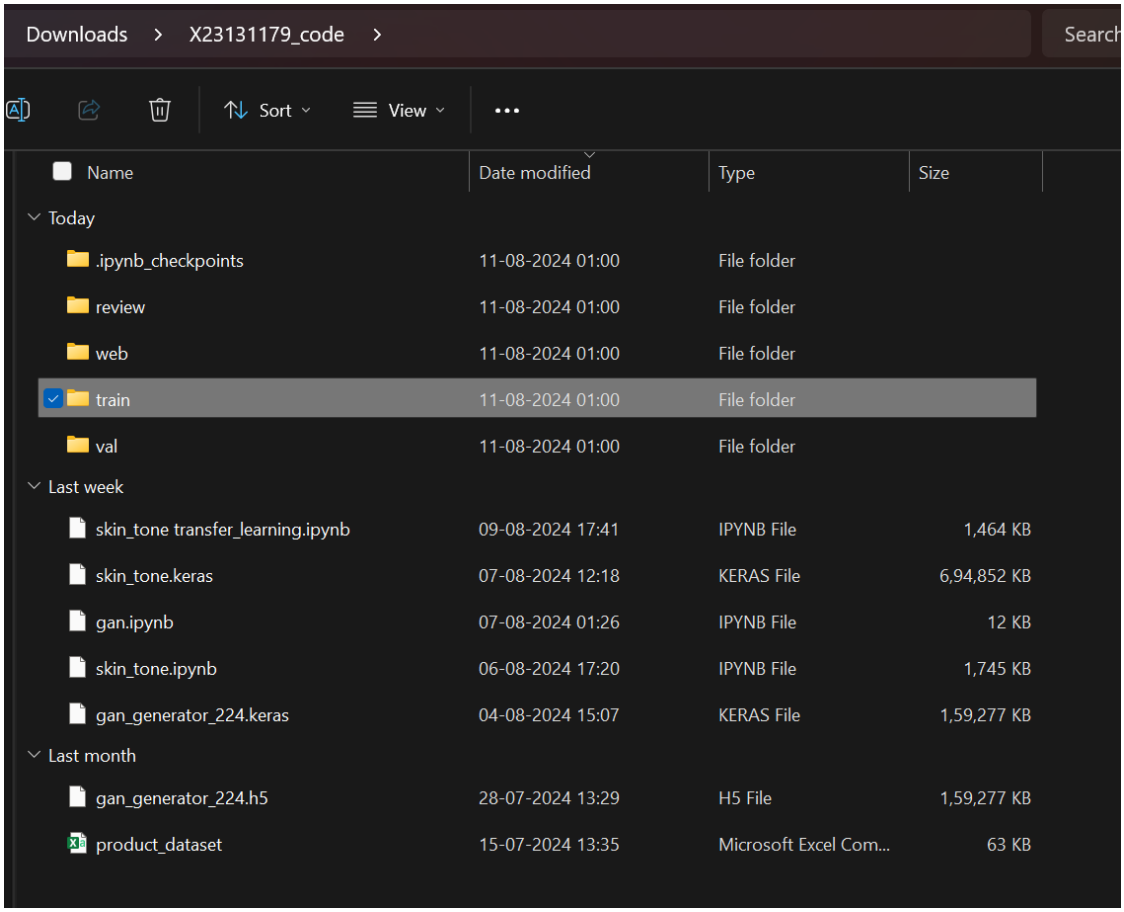

Figure 2

4 Installation

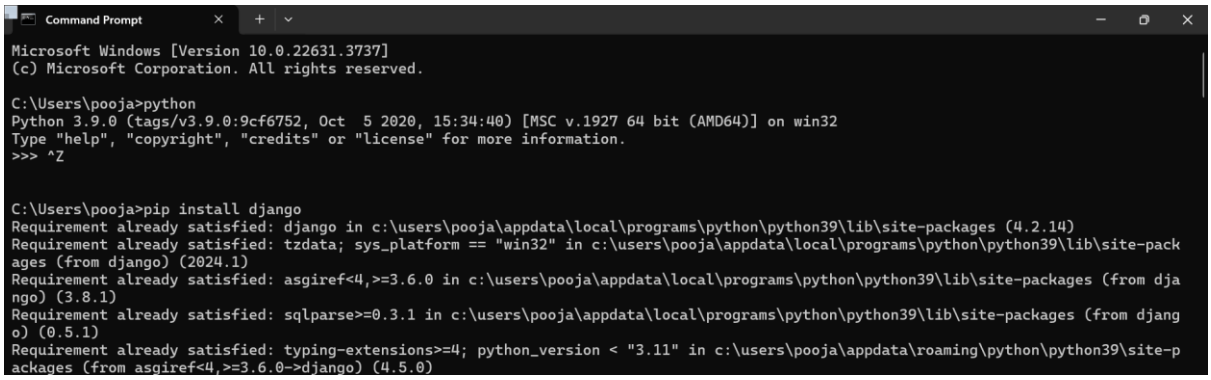

Figure 3

Open Command prompt and run the below commands to install Django which is used for website creation and additional commands required for the models to run.

pip install Django
pip install matplotlib
pip install tensorflow

pip install scikit-learn
pip install notebook

5 Data Preprocessing:

Data preprocessing is performed to make sure that the facial images are in a proper format for the deep learning models. This involves several key steps like resizing the images to a dimension that is consistent and scaling the pixel values. These steps will enhance the ability of the model to improve its performance.

```
import numpy as np
import matplotlib.pyplot as plt
import os
from tensorflow.keras.preprocessing.image import load_img, img_to_array

# Function to plot images in a grid
def plot_images(image_paths, labels, num_images=9):
    plt.figure(figsize=(10, 10))
    for i in range(num_images):
        plt.subplot(3, 3, i + 1)
        img = load_img(image_paths[i], target_size=(150, 150))
        plt.imshow(img)
        plt.title(labels[i])
        plt.axis('off')
    plt.show()

# Get some sample images and labels from the training directory
train_dir = 'train'
class_names = os.listdir(train_dir)

image_paths = []
labels = []

for class_name in class_names:
    class_dir = os.path.join(train_dir, class_name)
    for img_name in os.listdir(class_dir)[:3]:  # Get first 3 images from each class
        image_paths.append(os.path.join(class_dir, img_name))
        labels.append(class_name)

# Plot the images
plot_images(image_paths, labels)
```

Figure 4

6 Data Augmentation:

Data augmentation involves applying various transformations to the original images to create new and slightly altered versions.

```python
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Function to plot augmented images in a grid
def plot_augmented_images(image_path, num_images=9):
    datagen = ImageDataGenerator(
        rotation_range=40,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest'
    )

    img = load_img(image_path)
    img_array = img_to_array(img)
    img_array = img_array.reshape((1,) + img_array.shape)  # Reshape to (1, 150, 150, 3)

    plt.figure(figsize=(10, 10))
    i = 0
    for batch in datagen.flow(img_array, batch_size=1):
        plt.subplot(3, 3, i + 1)
        plt.imshow(batch[0].astype('uint8'))
        plt.axis('off')
        i += 1
        if i == num_images:
            break
    plt.show()

# Plot augmented images for one of the sample images
plot_augmented_images(image_paths[0])
```

Figure 5

7 Model Evaluation CNN

The Convolutional Neural Network (CNN) model has been evaluated to identify its effectiveness in classifying facial images into different skin types such as normal, oily and dry. The evaluation metrics which are used to access the performance includes Precision, accuracy, f1 and recall.

[19]:
```
from sklearn.metrics import precision_score, recall_score, f1_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
y_true = y_true_manual
y_pred = y_pre_manual
# Calculate precision, recall, and F1 score
precision = precision_score(y_true, y_pred, average='weighted')
recall = recall_score(y_true, y_pred, average='weighted')
f1 = f1_score(y_true, y_pred, average='weighted')

print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1 Score: {f1}')

# Generate and print classification report
class_report = classification_report(y_true, y_pred, target_names=class_labels)
print(class_report)

# Generate confusion matrix
conf_matrix = confusion_matrix(y_true, y_pred)

# Plot confusion matrix
plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=class_labels, yticklabels=class_labels)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```

```
Precision: 0.8621618622609056
Recall: 0.7554151624548736
F1 Score: 0.790583679948588
              precision    recall  f1-score   support

         dry       0.38      0.71      0.50       168
      normal       0.98      0.78      0.86       907
        oily       0.19      0.45      0.27        33

    accuracy                           0.76      1108
   macro avg       0.52      0.65      0.54      1108
weighted avg       0.86      0.76      0.79      1108
```

Figure 6

8 Model Training CNN:

Images has been trained using different model and among those CNN is found to be the best model which gives accuracy of 99 percent.

```
[7]: model = Sequential([
        Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)),
        MaxPooling2D(2, 2),
        Conv2D(64, (3, 3), activation='relu'),
        MaxPooling2D(2, 2),
        Conv2D(128, (3, 3), activation='relu'),
        MaxPooling2D(2, 2),
        Flatten(),
        Dense(512, activation='relu'),
        Dropout(0.5),
        Dense(len(train_generator.class_indices), activation='softmax')
    ])

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

[8]: history = model.fit(
        train_generator,
        epochs=25,
    )

    Epoch 1/25
    69/69 [==============================] - 66s 934ms/step - loss: 1.2104 - accuracy: 0.3966
    Epoch 2/25
    69/69 [==============================] - 68s 987ms/step - loss: 1.0412 - accuracy: 0.4683
    Epoch 3/25
    69/69 [==============================] - 69s 994ms/step - loss: 0.8768 - accuracy: 0.5798
    Epoch 4/25
    69/69 [==============================] - 69s 995ms/step - loss: 0.6227 - accuracy: 0.7344
    Epoch 5/25
    69/69 [==============================] - 68s 983ms/step - loss: 0.3948 - accuracy: 0.8386
    Epoch 6/25
    69/69 [==============================] - 67s 974ms/step - loss: 0.2462 - accuracy: 0.9112
    Epoch 7/25
    69/69 [==============================] - 67s 971ms/step - loss: 0.1607 - accuracy: 0.9442
    Epoch 8/25
    69/69 [==============================] - 69s 996ms/step - loss: 0.0848 - accuracy: 0.9737
    Epoch 9/25
    69/69 [==============================] - 68s 990ms/step - loss: 0.1032 - accuracy: 0.9737
    Epoch 10/25
    69/69 [==============================] - 67s 973ms/step - loss: 0.0521 - accuracy: 0.9887
    Epoch 11/25
    69/69 [==============================] - 68s 981ms/step - loss: 0.0532 - accuracy: 0.9859
    Epoch 12/25
    69/69 [==============================] - 69s 1s/step - loss: 0.0361 - accuracy: 0.9937
    Epoch 13/25
    69/69 [==============================] - 69s 1s/step - loss: 0.0332 - accuracy: 0.9909
    Epoch 14/25
    69/69 [==============================] - 69s 1s/step - loss: 0.0412 - accuracy: 0.9918
    Epoch 15/25
    69/69 [==============================] - 69s 1s/step - loss: 0.0267 - accuracy: 0.9927
    Epoch 16/25
    69/69 [==============================] - 69s 999ms/step - loss: 0.0274 - accuracy: 0.9937
    Epoch 17/25
    69/69 [==============================] - 68s 977ms/step - loss: 0.0149 - accuracy: 0.9959
    Epoch 18/25
```

Figure 7

9 Model Predictions

Model has been predicted to offer accurate recommendations which improves user satisfaction and overall performance of the system.

```
class_report = classification_report(y_true, y_pred, target_names=class_names, output_dict=True)
print(class_report)
```

```
{'Oily': {'precision': 0.38387096774193546, 'recall': 0.7083333333333334, 'f1-score': 0.49790794979079495, 'support': 168.0}, 'Normal': {'precision':
0.9750346740638003, 'recall': 0.7750826901874311, 'f1-score': 0.8636363636363638, 'support': 907.0}, 'Dry': {'precision': 0.19480519480519481, 'recal
l': 0.45454545454545453, 'f1-score': 0.2727272727272727, 'support': 33.0}, 'accuracy': 0.7554151624548736, 'macro avg': {'precision': 0.517903612203643
4, 'recall': 0.6459871593554064, 'f1-score': 0.5447571953848105, 'support': 1108.0}, 'weighted avg': {'precision': 0.8621618622609056, 'recall': 0.7554
151624548736, 'f1-score': 0.790583679948588, 'support': 1108.0}}
```

```
import numpy as np

# Reset the generator and make predictions
train_generator.reset()
predictions = model.predict(train_generator)
predicted_classes = np.argmax(predictions, axis=1)
```

```
69/69 [==============================] - 12s 170ms/step
```

```
# Get true labels
true_classes = train_generator.classes
class_labels = list(train_generator.class_indices.keys())
```

Figure 9

10 Running code:

To run the codes after successfully installing Jupyter notebooks, follow these steps:

1. Navigate to the Folder: Open a terminal (or command prompt) and navigate to the folder X23131179_code.

   cd path_to_folder/X23131179_code

2. Launch Jupyter Notebook: Start Jupyter Notebook by running:

   jupyter notebook

3. This will open a web interface of Jupyter Notebook in your default browser.

4. Then Run the below Code Files

- Open the gan.ipynb notebook: In the Jupyter interface, navigate to the gan.ipynb file and open it. click Run all cells to run the code.

- Open the skin_tone_transfer_learning.ipynb notebook: Similarly, open this notebook file, and execute the code.

- Open the skin_tone.ipynb notebook: Finally, run the skin_tone.ipynb file in the same manner.

- Then Access the Website for Product Recommendations which is described in the next section.

11 Website

The project creates a website as a user interface where the skincare products are recommended. In order for the website to run using the code given in the artifacts, the following command should be entered in the command prompt " Python manage.py runserver ".

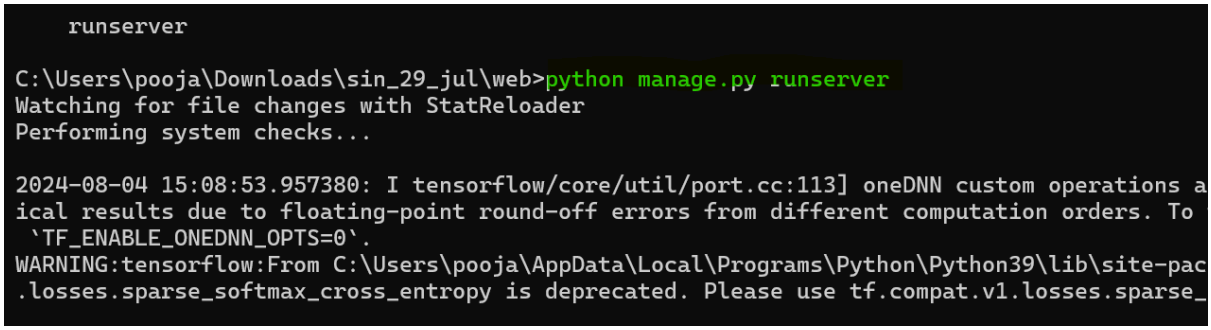

Figure 10

Click on the link highlighted in the below image to open the webpage.

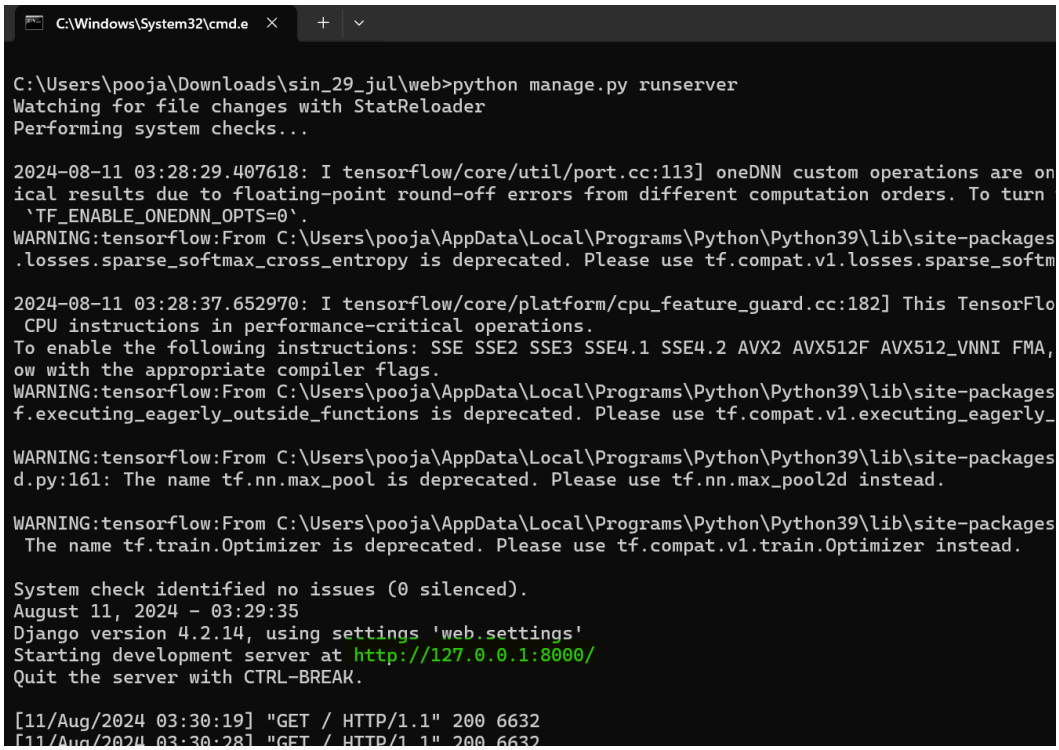

Figure 11

The webpage will be displayed as shown in Fig 12.

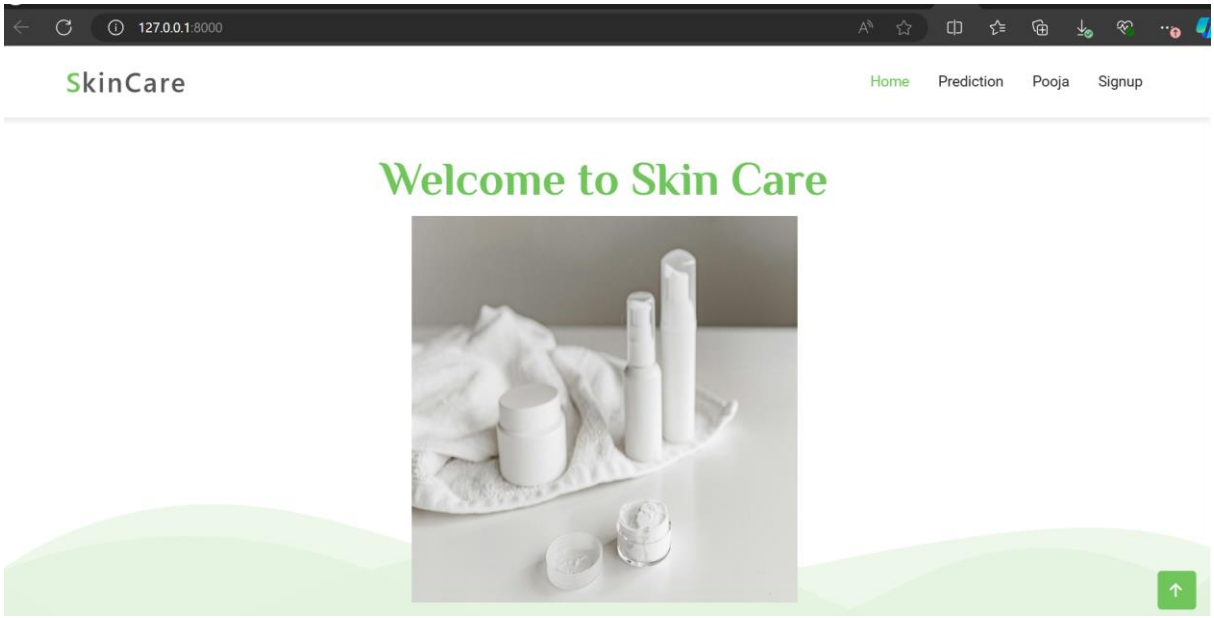

Figure 12

Click on prediction tab and chose the File to be uploaded.

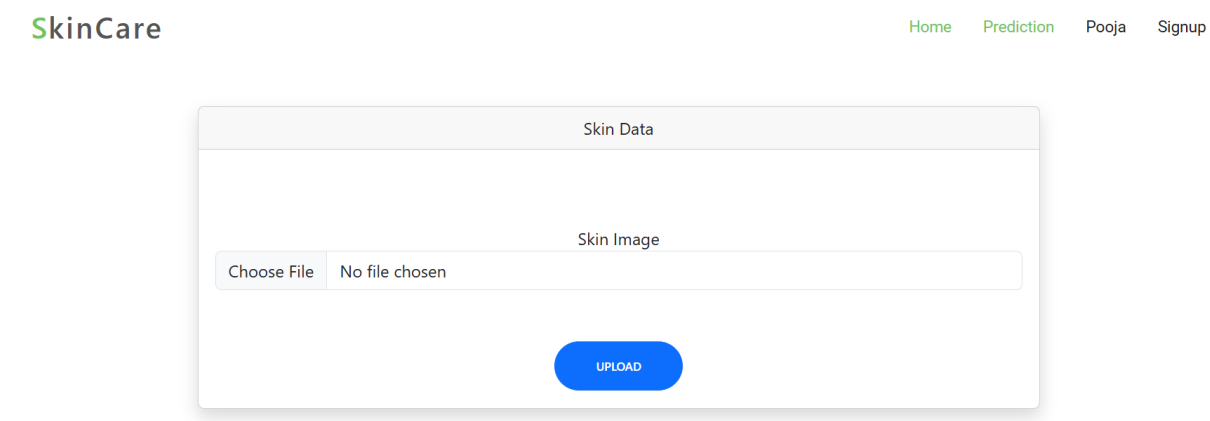

Figure 13

Products will be recommended once the image is uploaded.

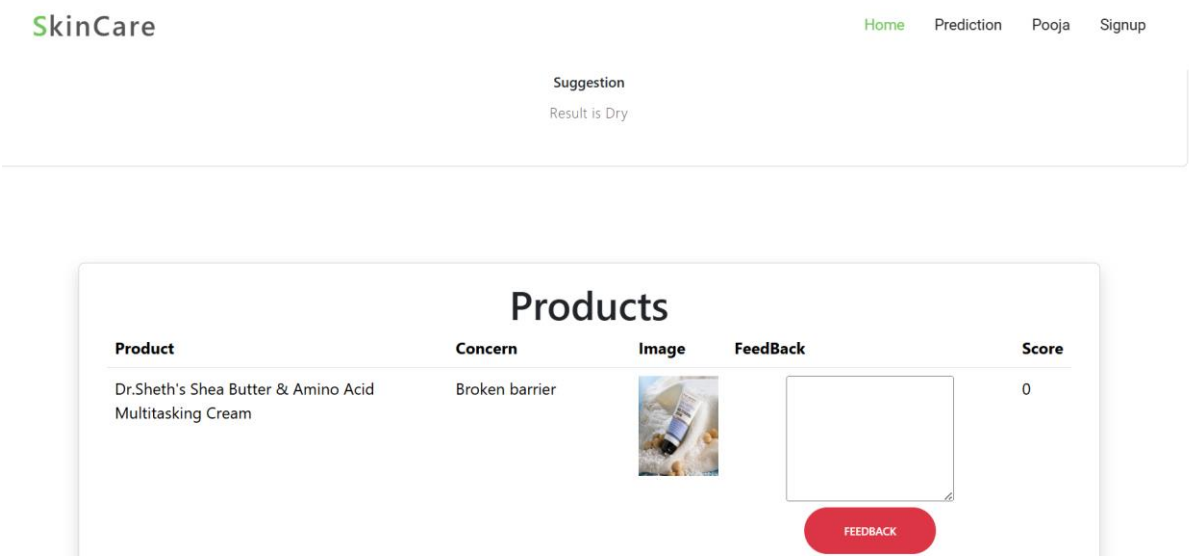

Figure 14