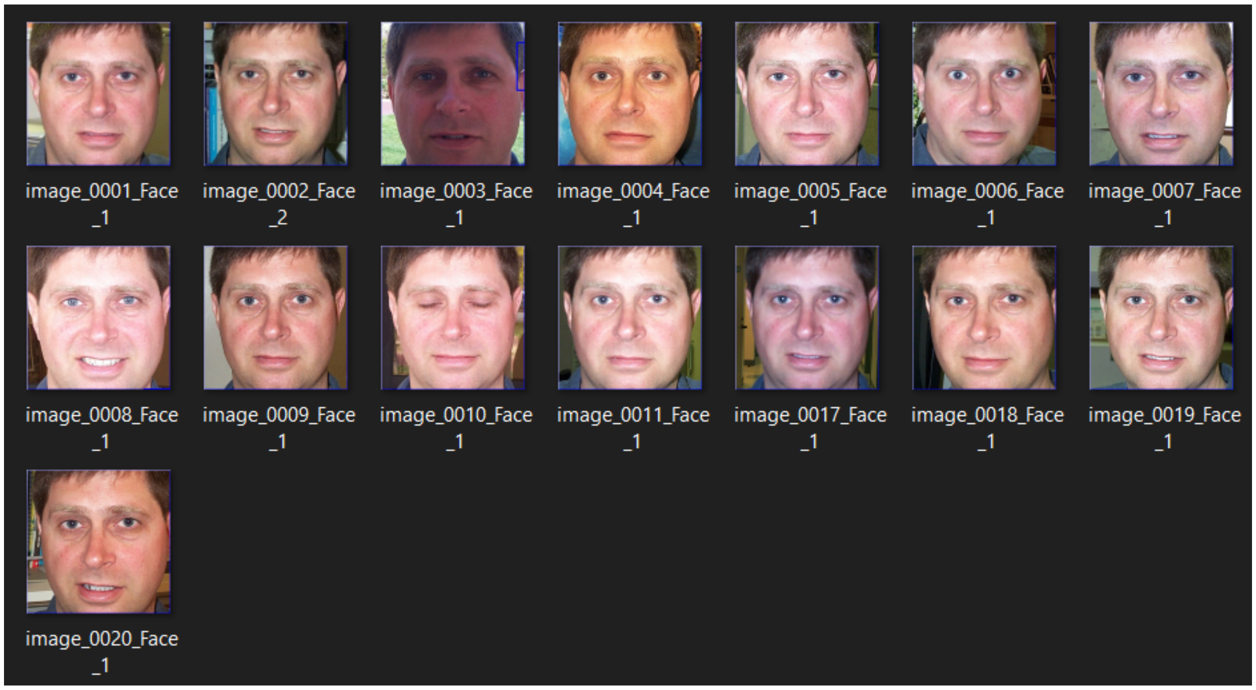
October 11, 2021

Wrote a python file to make a CNN model for test data of faces. This was for a proof of concept to see if the imageDataGenerator class would be applicable to our project. It used 16 groups each represented by a folder, where the folders contained images of only the same person, but each folder contained images of different people. The CNN model was able to accurately categorize an unseen image 97% of the time.



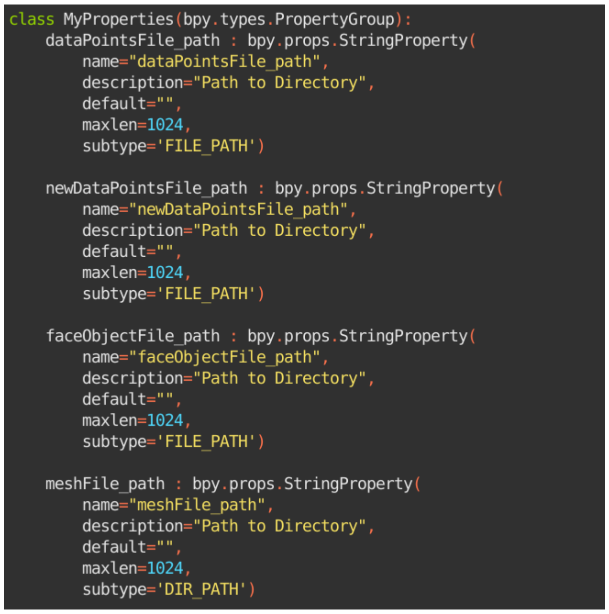
The plan is to apply this concept with groups of similarly structured faces to suggest faces that look similar to one’s own.

October 18, 2021

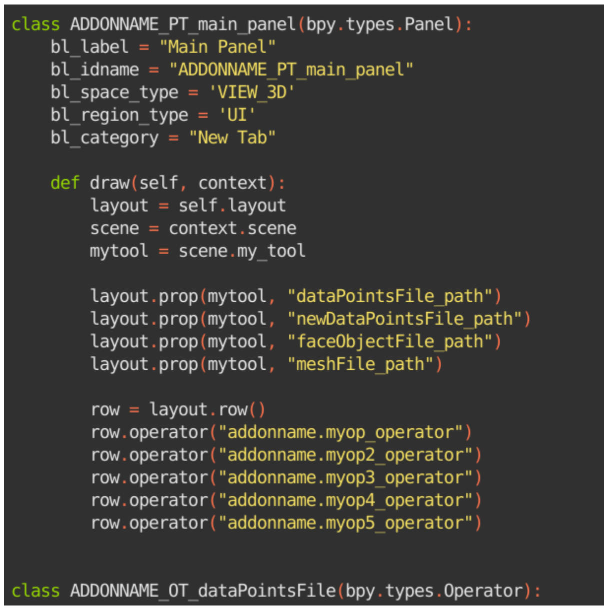
Researched solutions to the old Blender script. The old software had issues with importing _tkinter because Blender doesn’t support it anymore. Accordingly, I decided to begin rewriting the Blender script using its own built-in GUI rather than python’s.

October 21, 2021

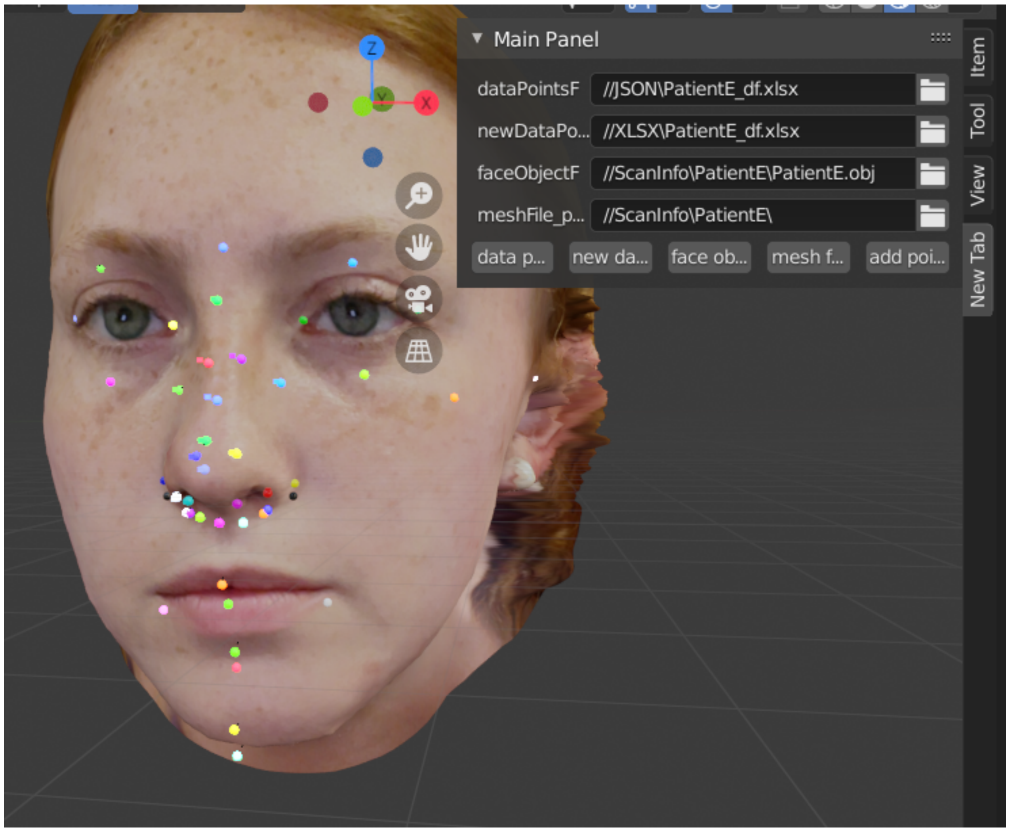
Completed Blender script to model a patient’s face and plot the significant landmarks. Added and registered panel properties and classes to generate usable UI.



```python
class MyProperties(bpy.types.PropertyGroup):
    dataPointsFile_path : bpy.props.StringProperty(
        name="dataPointsFile_path",
        description="Path to Directory",
        default="",
        maxlen=1024,
        subtype='FILE_PATH')

    newDataPointsFile_path : bpy.props.StringProperty(
        name="newDataPointsFile_path",
        description="Path to Directory",
        default="",
        maxlen=1024,
        subtype='FILE_PATH')

    faceObjectFile_path : bpy.props.StringProperty(
        name="faceObjectFile_path",
        description="Path to Directory",
        default="",
        maxlen=1024,
        subtype='FILE_PATH')

    meshFile_path : bpy.props.StringProperty(
        name="meshFile_path",
        description="Path to Directory",
        default="",
        maxlen=1024,
        subtype='DIR_PATH')
```



```python
class ADDONNAME_PT_main_panel(bpy.types.Panel):
    bl_label = "Main Panel"
    bl_idname = "ADDONNAME_PT_main_panel"
    bl_space_type = 'VIEW_3D'
    bl_region_type = 'UI'
    bl_category = "New Tab"

    def draw(self, context):
        layout = self.layout
        scene = context.scene
        mytool = scene.my_tool

        layout.prop(mytool, "dataPointsFile_path")
        layout.prop(mytool, "newDataPointsFile_path")
        layout.prop(mytool, "faceObjectFile_path")
        layout.prop(mytool, "meshFile_path")

        row = layout.row()
        row.operator("addonname.myop_operator")
        row.operator("addonname.myop2_operator")
        row.operator("addonname.myop3_operator")
        row.operator("addonname.myop4_operator")
        row.operator("addonname.myop5_operator")


class ADDONNAME_OT_dataPointsFile(bpy.types.Operator):
```



Main Panel

dataPointsF //JSON\PatientE_df.xlsx

newDataPo... //XLSX\PatientE_df.xlsx

faceObjectF //ScanInfo\PatientE\PatientE.obj

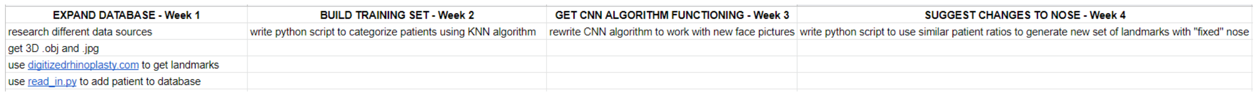
meshFile_p... //ScanInfo\PatientE\

October 26, 2021

Set up a group meeting to work on system requirements specifications document as well as the system design document. Completed the introduction and overall description sections of SRS. Completed the introduction and system architecture sections of SDD.

November 1, 2021

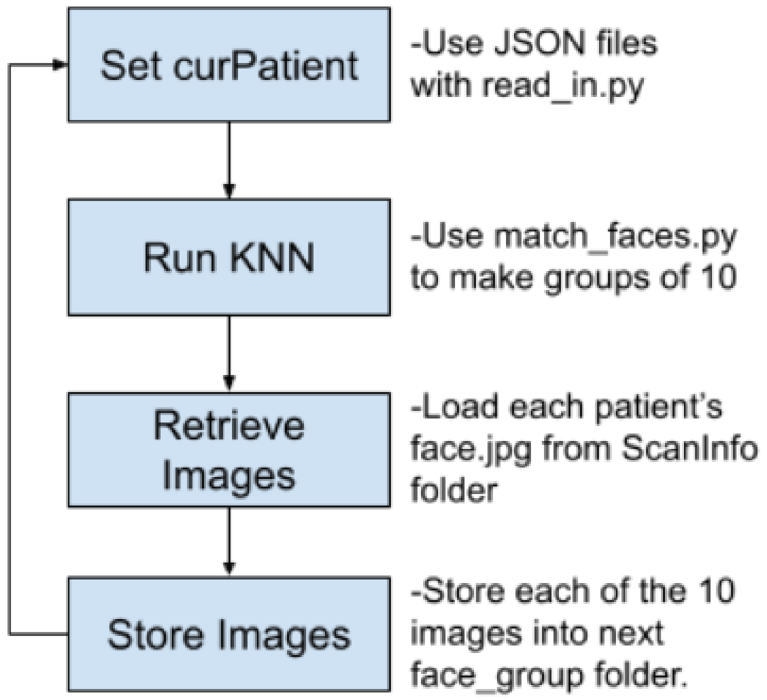
Created a general timeline of the tasks that needed to be completed for sprint 3. Set up a group meeting to assign group members their respective roles in order to finish the necessary tasks on time.



EXPAND DATABASE - Week 1	BUILD TRAINING SET - Week 2	GET CNN ALGORITHM FUNCTIONING - Week 3	SUGGEST CHANGES TO NOSE - Week 4
research different data sources	write python script to categorize patients using KNN algorithm	rewrite CNN algorithm to work with new face pictures	write python script to use similar patient ratios to generate new set of landmarks with "fixed" nose
get 3D .obj and .jpg			
use digitizedrhinoplasty.com to get landmarks			
use read_in.py to add patient to database			

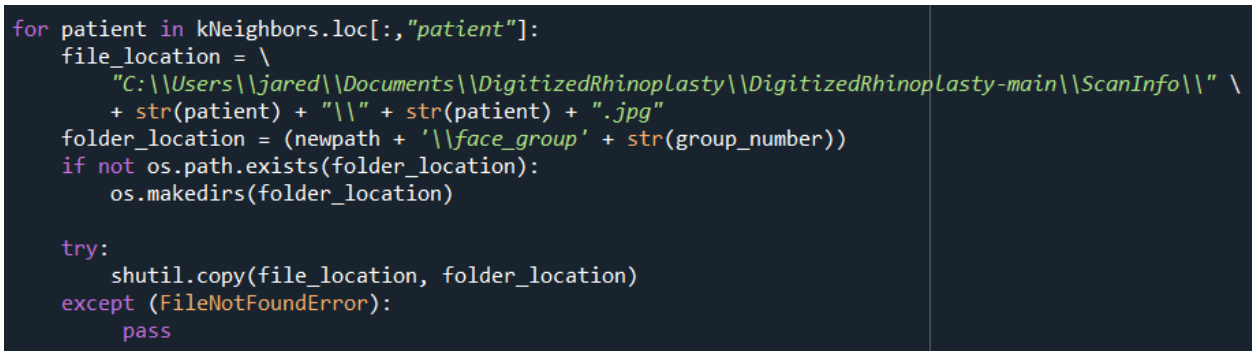
November 8, 2021

Developed structure/tasks to be completed for the creation of face groupings. I created a flow chart for the series of tasks that needed to be completed in order to generate a group of face images.


Set curPatient
-Use JSON files with read_in.py
Run KNN
-Use match_faces.py to make groups of 10
Retrieve Images
-Load each patient's face.jpg from ScanInfo folder
Store Images
-Store each of the 10 images into next face_group folder.

November 10, 2021

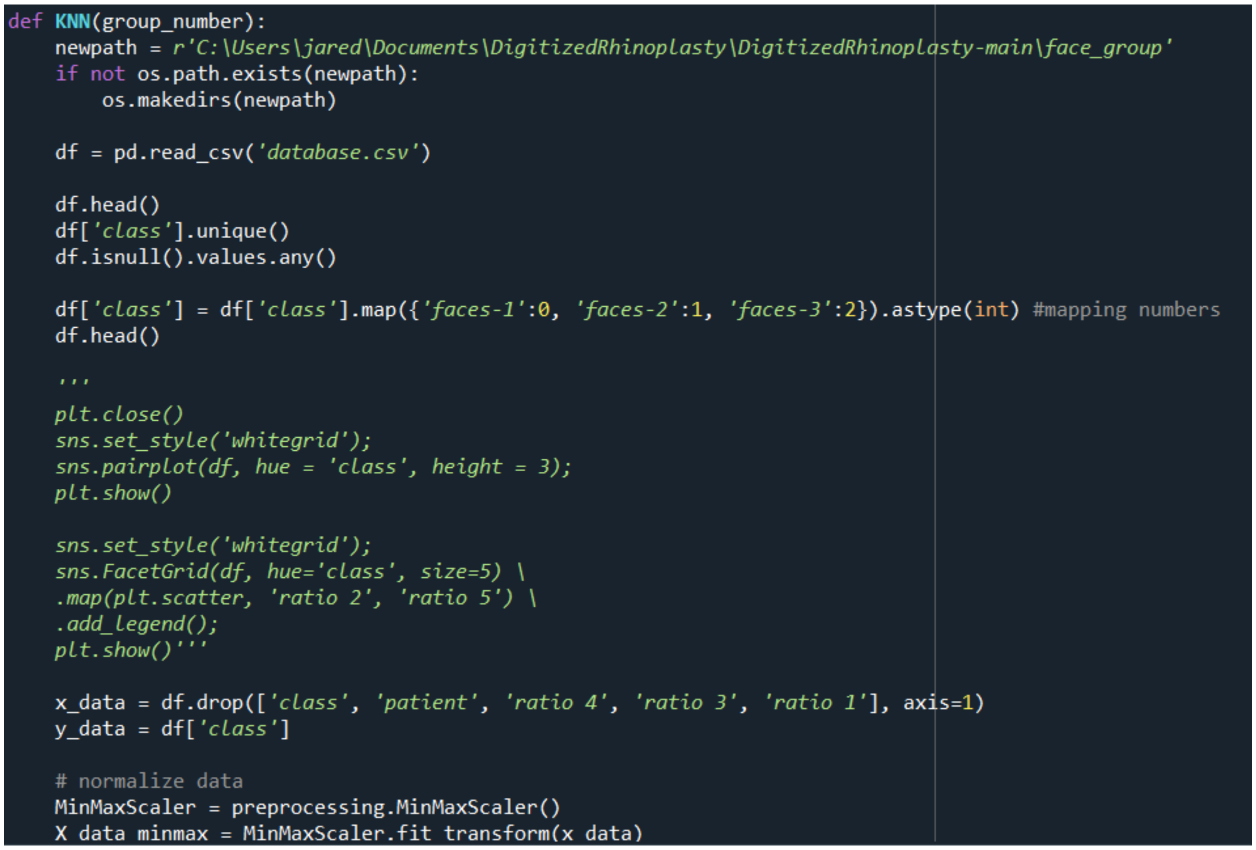
Edited match_faces.py to generate a folder containing a grouping of patient's face jpgs based on the output of the KNN algorithm.



```python
for patient in kNeighbors.loc[:,"patient"]:
    file_location = \
        "C:\\Users\\jared\\Documents\\DigitizedRhinoplasty\\DigitizedRhinoplasty-main\\ScanInfo\\" \
        + str(patient) + "\\" + str(patient) + ".jpg"
    folder_location = (newpath + '\\face_group' + str(group_number))
    if not os.path.exists(folder_location):
        os.makedirs(folder_location)

    try:
        shutil.copy(file_location, folder_location)
    except (FileNotFoundError):
         pass
```

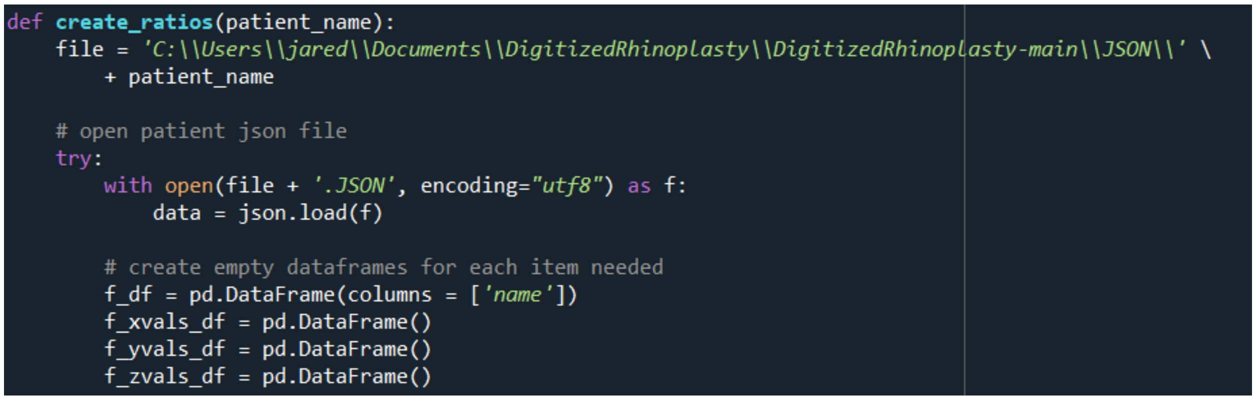
November 18, 2021

Encapsulated match_faces.py into a callable function that takes in the group_number as a parameter. This added functionality allows the KNN algorithm to be called multiple times in a separate program.



```python
def KNN(group_number):
    newpath = r'C:\Users\jared\Documents\DigitizedRhinoplasty\DigitizedRhinoplasty-main\face_group'
    if not os.path.exists(newpath):
        os.makedirs(newpath)

    df = pd.read_csv('database.csv')

    df.head()
    df['class'].unique()
    df.isnull().values.any()

    df['class'] = df['class'].map({'faces-1':0, 'faces-2':1, 'faces-3':2}).astype(int) #mapping numbers
    df.head()

    '''
    plt.close()
    sns.set_style('whitegrid');
    sns.pairplot(df, hue = 'class', height = 3);
    plt.show()

    sns.set_style('whitegrid');
    sns.FacetGrid(df, hue='class', size=5) \
    .map(plt.scatter, 'ratio 2', 'ratio 5') \
    .add_legend();
    plt.show()'''

    x_data = df.drop(['class', 'patient', 'ratio 4', 'ratio 3', 'ratio 1'], axis=1)
    y_data = df['class']

    # normalize data
    MinMaxScaler = preprocessing.MinMaxScaler()
    X_data_minmax = MinMaxScaler.fit_transform(x_data)
```

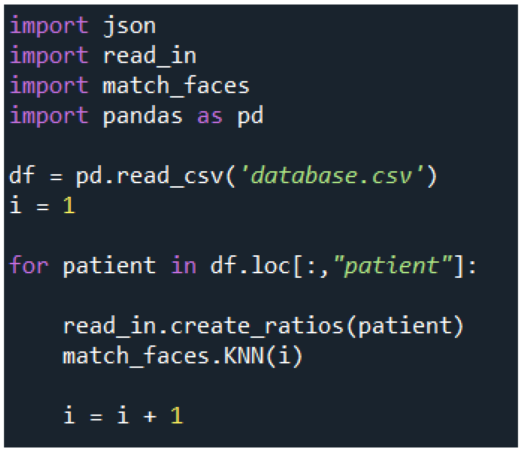
Encapsulated read_in.py into a callable function (create_ratios()) that takes a patient's name as input in order for a separate file to set the current patient multiple times.



```
def create_ratios(patient_name):
    file = 'C:\\Users\\jared\\Documents\\DigitizedRhinoplasty\\DigitizedRhinoplasty-main\\JSON\\' \
        + patient_name

    # open patient json file
    try:
        with open(file + '.JSON', encoding="utf8") as f:
            data = json.load(f)

        # create empty dataframes for each item needed
        f_df = pd.DataFrame(columns = ['name'])
        f_xvals_df = pd.DataFrame()
        f_yvals_df = pd.DataFrame()
        f_zvals_df = pd.DataFrame()
```

November 22, 2021

Wrote a python program to use read_in.py and match_faces.py to generate groupings of 10 or less faces for each patient in the database.

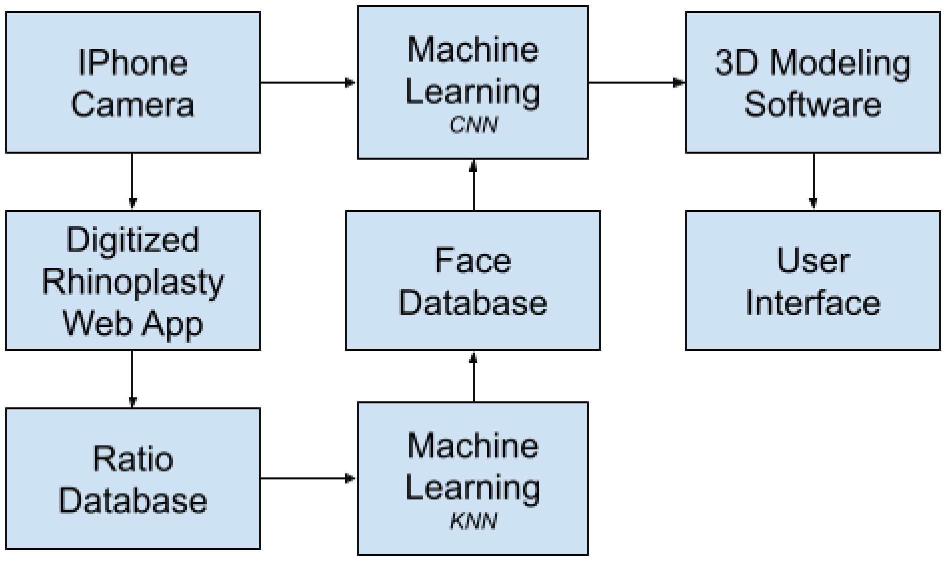


```
import json
import read_in
import match_faces
import pandas as pd

df = pd.read_csv('database.csv')
i = 1

for patient in df.loc[:,"patient"]:

    read_in.create_ratios(patient)
    match_faces.KNN(i)

    i = i + 1
```

November 30, 2021

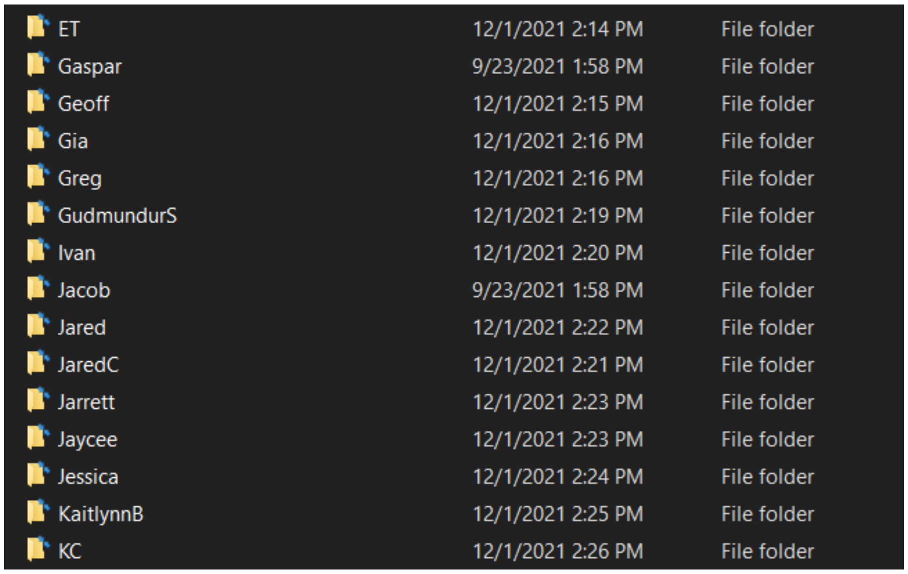
Set up a group meeting to finalize system requirements specifications document as well as the system design document. Completed the external interface requirements, system features, and other functional requirements sections of the SRS. Completed the human machine interface, detailed design, and external interfaces sections of the SDD.

Updated system architecture diagram to more accurately apply to the new development/code structure.



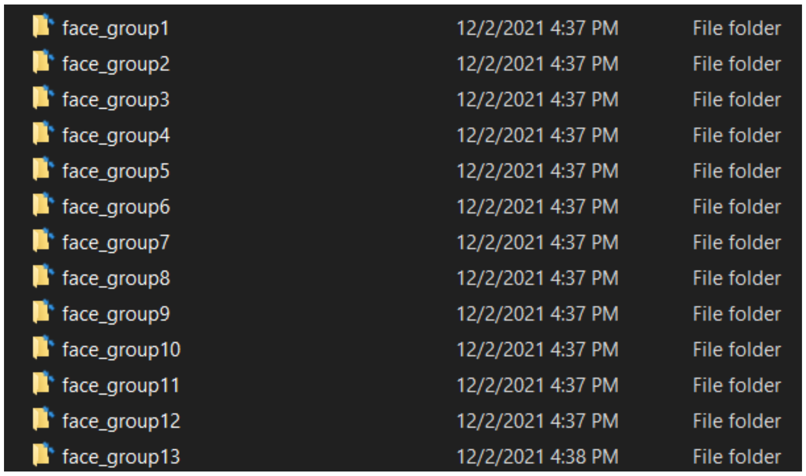
December 1, 2021

Downloaded 77 new face scans provided by Alessandra Oo and Maree Kelley and added them to their respective folders in the Scaninfo folder.



ET	12/1/2021 2:14 PM	File folder
Gaspar	9/23/2021 1:58 PM	File folder
Geoff	12/1/2021 2:15 PM	File folder
Gia	12/1/2021 2:16 PM	File folder
Greg	12/1/2021 2:16 PM	File folder
GudmundurS	12/1/2021 2:19 PM	File folder
Ivan	12/1/2021 2:20 PM	File folder
Jacob	9/23/2021 1:58 PM	File folder
Jared	12/1/2021 2:22 PM	File folder
JaredC	12/1/2021 2:21 PM	File folder
Jarrett	12/1/2021 2:23 PM	File folder
Jaycee	12/1/2021 2:23 PM	File folder
Jessica	12/1/2021 2:24 PM	File folder
KaitlynnB	12/1/2021 2:25 PM	File folder
KC	12/1/2021 2:26 PM	File folder

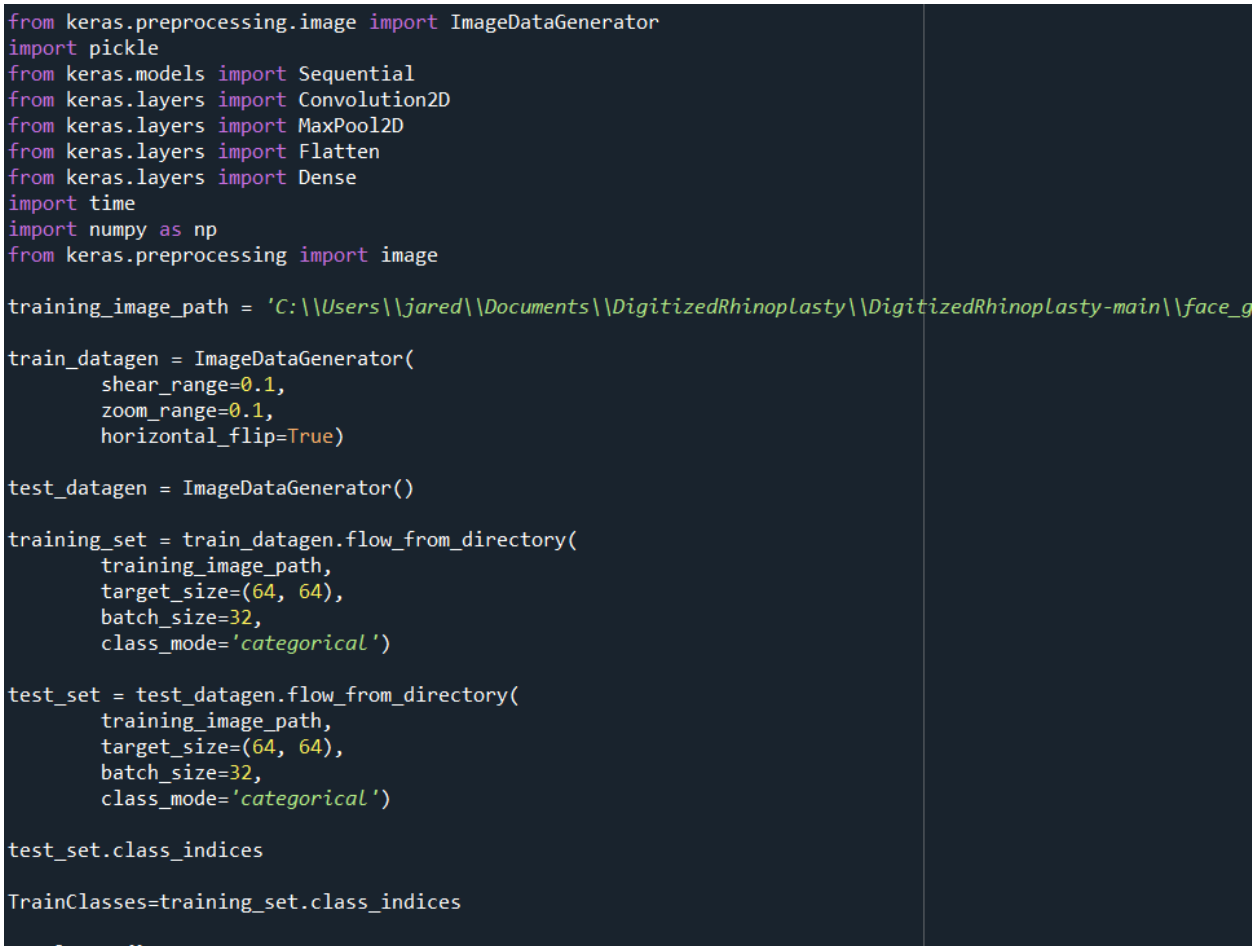
Used the newly added faces and sorter.py to create face groups of similarly structured faces, and saved them in respective folders.



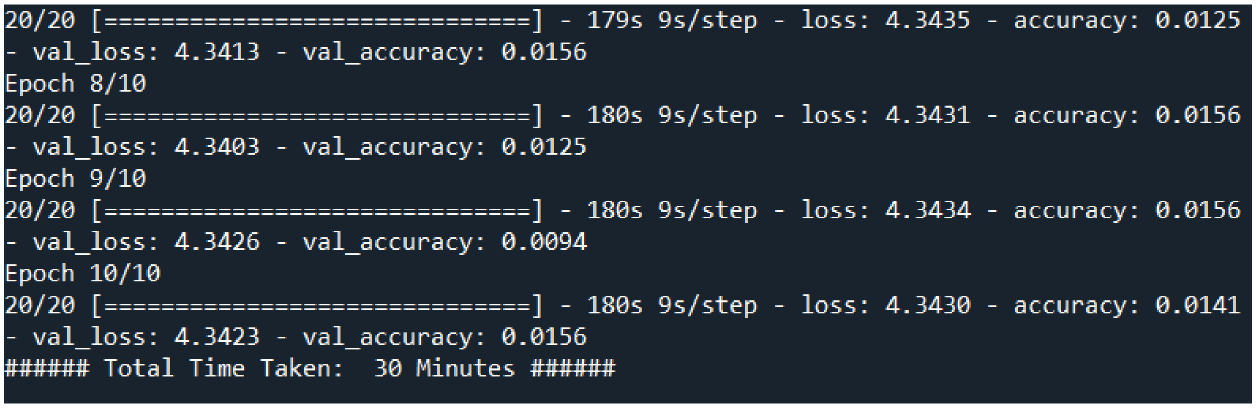
face_group1	12/2/2021 4:37 PM	File folder
face_group2	12/2/2021 4:37 PM	File folder
face_group3	12/2/2021 4:37 PM	File folder
face_group4	12/2/2021 4:37 PM	File folder
face_group5	12/2/2021 4:37 PM	File folder
face_group6	12/2/2021 4:37 PM	File folder
face_group7	12/2/2021 4:37 PM	File folder
face_group8	12/2/2021 4:37 PM	File folder
face_group9	12/2/2021 4:37 PM	File folder
face_group10	12/2/2021 4:37 PM	File folder
face_group11	12/2/2021 4:37 PM	File folder
face_group12	12/2/2021 4:37 PM	File folder
face_group13	12/2/2021 4:38 PM	File folder

December 2, 2021

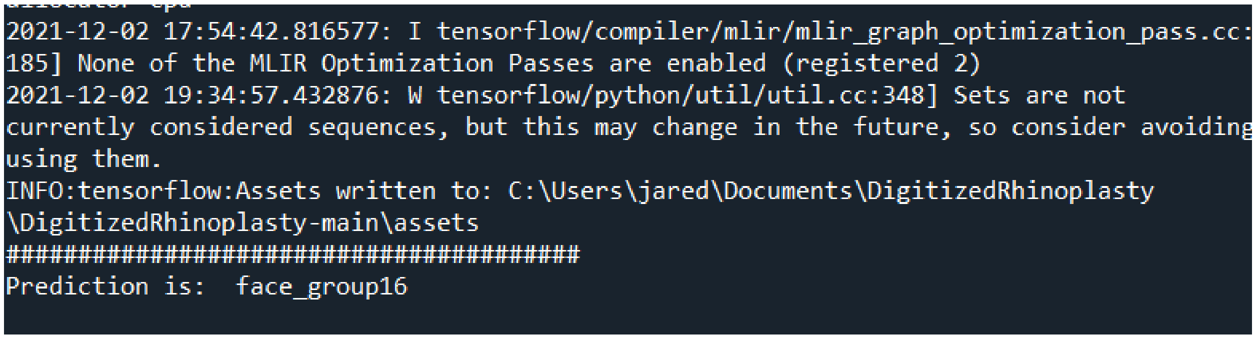
Wrote the python script match_faces_CNN.py, which creates and saves a CNN model using the face groupings created earlier as the training set.



```python
from keras.preprocessing.image import ImageDataGenerator
import pickle
from keras.models import Sequential
from keras.layers import Convolution2D
from keras.layers import MaxPool2D
from keras.layers import Flatten
from keras.layers import Dense
import time
import numpy as np
from keras.preprocessing import image

training_image_path = 'C:\\Users\\jared\\Documents\\DigitizedRhinoplasty\\DigitizedRhinoplasty-main\\face_g

train_datagen = ImageDataGenerator(
        shear_range=0.1,
        zoom_range=0.1,
        horizontal_flip=True)

test_datagen = ImageDataGenerator()

training_set = train_datagen.flow_from_directory(
        training_image_path,
        target_size=(64, 64),
        batch_size=32,
        class_mode='categorical')

test_set = test_datagen.flow_from_directory(
        training_image_path,
        target_size=(64, 64),
        batch_size=32,
        class_mode='categorical')

test_set.class_indices

TrainClasses=training_set.class_indices
```



```
20/20 [==============================] - 179s 9s/step - loss: 4.3435 - accuracy: 0.0125
- val_loss: 4.3413 - val_accuracy: 0.0156
Epoch 8/10
20/20 [==============================] - 180s 9s/step - loss: 4.3431 - accuracy: 0.0156
- val_loss: 4.3403 - val_accuracy: 0.0125
Epoch 9/10
20/20 [==============================] - 180s 9s/step - loss: 4.3434 - accuracy: 0.0156
- val_loss: 4.3426 - val_accuracy: 0.0094
Epoch 10/10
20/20 [==============================] - 180s 9s/step - loss: 4.3430 - accuracy: 0.0141
- val_loss: 4.3423 - val_accuracy: 0.0156
###### Total Time Taken:  30 Minutes ######
```



```
2021-12-02 17:54:42.816577: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:
185] None of the MLIR Optimization Passes are enabled (registered 2)
2021-12-02 19:34:57.432876: W tensorflow/python/util/util.cc:348] Sets are not
currently considered sequences, but this may change in the future, so consider avoiding
using them.
INFO:tensorflow:Assets written to: C:\Users\jared\Documents\DigitizedRhinoplasty
\DigitizedRhinoplasty-main\assets
########################################
Prediction is:  face_group16
```