changeImage.py

#!/usr/bin/env python3

import os, sys

from PIL import Image

user = os.getenv('USER') # To get the username from environment variable

image\_directory = '/home/{}/supplier-data/images/'.format(user)

for image\_name in os.listdir(image\_directory):

    if not image\_name.startswith('.') and 'tiff' in image\_name:

        image\_path = image\_directory + image\_name

        path = os.path.splitext(image\_path)[0]

        im = Image.open(image\_path)

        new\_path = '{}.jpeg'.format(path)

        im.convert('RGB').resize((600, 400)).save(new\_path, "JPEG")

supplier\_image\_upload.py

#!/usr/bin/env python3

import requests, os

# The URL to upload the images

url = "http://localhost/upload/"

# To get the username from environment variable

USER = os.getenv('USER')

# The directory which contains all the images.

image\_directory = '/home/{}/supplier-data/images/'.format(USER)

# Listing all the files in images directory

files = os.listdir(image\_directory)

# Parsing through all the images

for image\_name in files:

    # Accepting files that has jpeg extension and ignoring hidden files

    if not image\_name.startswith('.') and 'jpeg' in image\_name:

        # creating absolute path for each image

        image\_path = image\_directory + image\_name

        # uploading jpeg files

        with open(image\_path, 'rb') as opened:

            r = requests.post(url, files={'file': opened})

run.py

#!/usr/bin/env python3

import os, requests, json

def catalog\_data(url,description\_dir):

    """This function will return a list of dictionaries with all the details like name, weight, description, image\_name.

    It will change the weight to integer format.

    """

    fruit={}

    for item in os.listdir(description\_dir):

      fruit.clear()

      filename=os.path.join(description\_dir,item)

      with open(filename) as f:

        line=f.readlines()

        description=""

        for i in range(2,len(line)):

          description=description+line[i].strip('\n').replace(u'\xa0',u'')

        fruit["description"]=description

        fruit["weight"]=int(line[1].strip('\n').strip('lbs'))

        fruit["name"]=line[0].strip('\n')

        fruit["image\_name"]=(item.strip('.txt'))+'.jpeg'

        print(fruit)

        if url!="":

          response=requests.post(url, json=fruit)

          print(response.request.url)

          print(response.status\_code)

    return 0

if \_\_name\_\_=='\_\_main\_\_':

    url = 'http://localhost/fruits/'

    user = os.getenv('USER')

    description\_directory = '/home/{}/supplier-data/descriptions/'.format(user)

    catalog\_data(url,description\_directory)

reports.py

#!/usr/bin/env python3

from reportlab.platypus import Paragraph, Spacer, Image, SimpleDocTemplate

from reportlab.lib.styles import getSampleStyleSheet

def generate\_report(file, title, add\_info):

    styles = getSampleStyleSheet()

    report = SimpleDocTemplate(file)

    report\_title = Paragraph(title, styles['h1'])

    report\_info = Paragraph(add\_info, styles['BodyText'])

    empty\_line = Spacer(1,20)

    report.build([report\_title, empty\_line, report\_info, empty\_line])

report\_email.py

#!/usr/bin/env python3

import datetime

import os

from run import catalog\_data

from reports import generate\_report

from emails import generate\_email, send\_email

def pdf\_body(input\_for,desc\_dir):

    """Generating a summary with two lists, which gives the output name and weight"""

    res = []

    wt = []

    for item in os.listdir(desc\_dir):

      filename=os.path.join(desc\_dir,item)

      with open(filename) as f:

        line=f.readlines()

        weight=line[1].strip('\n')

        name=line[0].strip('\n')

        print(name,weight)

        res.append('name: ' +name)

        wt.append('weight: ' +weight)

        print(res)

        print(wt)

    new\_obj = ""  # initializing the object

    # Calling values from two lists one by one.

    for i in range(len(res)):

        if res[i] and input\_for == 'pdf':

            new\_obj += res[i] + '<br />' + wt[i] + '<br />' + '<br />'

    return new\_obj

if \_\_name\_\_ == "\_\_main\_\_":

    user = os.getenv('USER')

    description\_directory = '/home/{}/supplier-data/descriptions/'.format(user)  # The directory which contains all the files with data in it.

    current\_date = datetime.date.today().strftime("%B %d, %Y")  # Creating data in format "May 5, 2020"

    title = 'Processed Update on ' + str(current\_date)  # Title for the PDF file with the created date

    generate\_report('/tmp/processed.pdf', title, pdf\_body('pdf',description\_directory))  # calling the report function from custom module

    email\_subject = 'Upload Completed - Online Fruit Store'  # subject line give in assignment for email

    email\_body = 'All fruits are uploaded to our website successfully. A detailed list is attached to this email.'  # body line give in assignment for email

    msg = generate\_email("automation@example.com", "{}@example.com".format(user),

                         email\_subject, email\_body, "/tmp/processed.pdf")  # structuring email and attaching the file. Then sending the email, using the cus$

    send\_email(msg)

emails.py

#!/usr/bin/env python3

import email

import mimetypes

import smtplib

import os

def generate\_email(sender, recipient, subject, body, attachment\_path):

    """Creates an email with an attachement."""

    # Basic Email formatting

    message = email.message.EmailMessage()

    message["From"] = sender

    message["To"] = recipient

    message["Subject"] = subject

    message.set\_content(body)

    # Making attachment\_path optional, if the attachment variable is empty string, no email will be sent.

    if not attachment\_path == "":

        # Process the attachment and add it to the email

        attachment\_filename = os.path.basename(attachment\_path)

        mime\_type, \_ = mimetypes.guess\_type(attachment\_path)

        mime\_type, mime\_subtype = mime\_type.split('/', 1)

        with open(attachment\_path, 'rb') as ap:

            message.add\_attachment(ap.read(), maintype=mime\_type, subtype=mime\_subtype,

                                   filename=attachment\_filename)

    return message

def send\_email(message):

    """Sends the message to the configured SMTP server."""

    mail\_server = smtplib.SMTP('localhost')

    mail\_server.send\_message(message)

    mail\_server.quit()

health\_check.py

#!/usr/bin/env python3

import socket

import shutil

import psutil

import emails

def check\_localhost():

    localhost = socket.gethostbyname('localhost')

    return localhost== "127.0.0.1"

def check\_disk\_usage(disk):

    """Verifies that there's enough free space on disk"""

    du = shutil.disk\_usage(disk)

    free = du.free / du.total \* 100

    return free > 20

def check\_memory\_usage():

    """Verifies that there's enough free space on disk"""

    mu = psutil.virtual\_memory().available

    total = mu / (1024.0 \*\* 2)

    return total > 500

def check\_cpu\_usage():

    """Verifies that there's enough unused CPU"""

    usage = psutil.cpu\_percent(1)

    return usage < 80

def send\_email(subject):

    email = emails.generate\_email("automation@example.com", "student-01-2e175e2f136d@example.com",

                                  subject,

                                  "Please check your system and resolve the issue as soon as possible.", "")

    emails.send\_email(email)

# If there's not enough disk, or not enough CPU, print an error

if not check\_cpu\_usage() :

    subject="Error - CPU usage is over 80%"

    print(subject)

    send\_email(subject)

if not check\_memory\_usage():

    subject = "Error - Available memory is less than 500MB"

    print(subject)

if not check\_disk\_usage('/') :

    subject = "Error - Available disk space is less than 20%"

    print(subject)

    send\_email(subject)

if not check\_localhost():

    subject = "Error - localhost cannot be resolved to 127.0.0.1"

    print(subject)

    send\_email(subject)