changeImage.py

1. #!/usr/bin/env python3
3. import os
4. from PIL import Image
6. path = os.path.expanduser('~') + '/supplier-data/images/'
8. for image in os.listdir(path):
9. if '.tiff' in image and '.' not in image[0]:
10. img = Image.open(path + image)
11. img.resize((600, 400)).convert("RGB").save(path + image.split('.')[0] + '.jpeg' , 'jpeg')
12. img.close()

supplier\_image\_upload.py

1. *#!/usr/bin/env python3*
3. **import** requests
4. **import** os
6. *# This example shows how a file can be uploaded using*
7. *# The Python Requests module*
8. url = "http://localhost/upload/"
9. IMAGE\_DIR = os.path.expanduser('~') + '/supplier-data/images/'
10. list\_image = os.listdir(IMAGE\_DIR)
11. jpeg\_images = [image\_name **for** image\_name **in** list\_image **if** '.jpeg' **in** image\_name]
13. **for** image **in** jpeg\_images:
14. **with** open(IMAGE\_DIR + image, 'rb') **as** opened:
15. r = requests.post(url, files={'file': opened})

run.py

1. *#! /usr/bin/env python3*
3. **import** os
4. **import** requests

7. BASEPATH\_SUPPLIER\_TEXT\_DES = os.path.expanduser('~') + '/supplier-data/descriptions/'
8. list\_text\_files = os.listdir(BASEPATH\_SUPPLIER\_TEXT\_DES)
10. BASEPATH\_SUPPLIER\_IMAGE = os.path.expanduser('~') + '/supplier-data/images/'
11. list\_image\_files = os.listdir(BASEPATH\_SUPPLIER\_IMAGE)
12. list\_images = [image\_name **for** image\_name **in** list\_image\_files **if** '.jpeg' **in** image\_name]

15. list = []
16. **for** text\_file **in** list\_text\_files:
17. **with** open(BASEPATH\_SUPPLIER\_TEXT\_DES + text\_file, 'r') **as** f:
18. data = {"name":f.readline().rstrip("**\n**"),
19. "weight":int(f.readline().rstrip("**\n**").split(' ')[0]),
20. "description":f.readline().rstrip("**\n**")}
22. **for** image\_file **in** list\_images:
23. **if** image\_file.split('.')[0] **in** text\_file.split('.')[0]:
24. data['image\_name'] = image\_file
26. list.append(data)
28. **for** item **in** list:
29. resp = requests.post('http://127.0.0.1:80/fruits/', json=item)
30. **if** resp.status\_code != 201:
31. **raise** Exception('POST error status={}'.format(resp.status\_code))
32. **print**('Created feedback ID: {}'.format(resp.json()["id"]))

reports.py

1. *#!/usr/bin/env python3*
3. **from** reportlab.platypus **import** SimpleDocTemplate
4. **from** reportlab.platypus **import** Paragraph, Spacer, Image
5. **from** reportlab.lib.styles **import** getSampleStyleSheet
6. **from** reportlab.lib **import** colors
8. **def** generate\_report(filename, title, additional\_info):
9. styles = getSampleStyleSheet()
10. report = SimpleDocTemplate(filename)
11. report\_title = Paragraph(title, styles["h1"])
12. report\_info = Paragraph(additional\_info, styles["Normal"])
13. empty\_line = Spacer(1,20)
14. report.build([report\_title, empty\_line, report\_info])

email.py

1. *#!/usr/bin/env python3*
3. **import** email.message
4. **import** mimetypes
5. **import** os.path
6. **import** smtplib
8. **def** generate\_email(sender, recipient, subject, body, attachment\_path):
9. """Creates an email with an attachement."""
10. *# Basic Email formatting*
11. message = email.message.EmailMessage()
12. message["From"] = sender
13. message["To"] = recipient
14. message["Subject"] = subject
15. message.set\_content(body)
17. *# Process the attachment and add it to the email*
18. attachment\_filename = os.path.basename(attachment\_path)
19. mime\_type, \_ = mimetypes.guess\_type(attachment\_path)
20. mime\_type, mime\_subtype = mime\_type.split('/', 1)
22. **with** open(attachment\_path, 'rb') **as** ap:
23. message.add\_attachment(ap.read(),
24. maintype=mime\_type,
25. subtype=mime\_subtype,
26. filename=attachment\_filename)
28. **return** message
30. **def** send\_email(message):
31. """Sends the message to the configured SMTP server."""
32. mail\_server = smtplib.SMTP('localhost')
33. mail\_server.send\_message(message)
34. mail\_server.quit()
36. **def** generate\_error\_report(sender, recipient, subject, body):
38. message = email.message.EmailMessage()
39. message["From"] = sender
40. message["To"] = recipient
41. message["Subject"] = subject
42. message.set\_content(body)
44. **return** message

health\_check.py

1. *#! /usr/bin/env python3*
3. **import** os
4. **import** shutil
5. **import** psutil
6. **import** socket
7. **from** emails **import** generate\_error\_report, send\_email
9. **def** check\_cpu\_usage():
10. """Verifies that there's enough unused CPU"""
11. usage = psutil.cpu\_percent(1)
12. **return** usage > 80
14. **def** check\_disk\_usage(disk):
15. """Verifies that there's enough free space on disk"""
16. du = shutil.disk\_usage(disk)
17. free = du.free / du.total \* 100
18. **return** free > 20
20. **def** check\_available\_memory():
21. """available memory in linux-instance, in byte"""
22. available\_memory = psutil.virtual\_memory().available/(1024\*1024)
23. **return** available\_memory > 500
25. **def** check\_localhost():
26. """check localhost is correctly configured on 127.0.0.1"""
27. localhost = socket.gethostbyname('localhost')
28. **return** localhost == '127.0.0.1'
30. **if** check\_cpu\_usage():
31. error\_message = "CPU usage is over 80%"
32. **elif** **not** check\_disk\_usage('/'):
33. error\_message = "Available disk space is less than 20%"
34. **elif** **not** check\_available\_memory():
35. error\_message = "Available memory is less than 500MB"
36. **elif** **not** check\_localhost():
37. error\_message = "localhost cannot be resolved to 127.0.0.1"
38. **else**:
39. **pass**
41. *# send email if any error reported*
42. **if** \_\_name\_\_ == "\_\_main\_\_":
43. **try**:
44. sender = "automation@example.com"
45. receiver = "{}@example.com".format(os.environ.get('USER'))
46. subject = "Error - {}".format(error\_message)
47. body = "Please check your system and resolve the issue as soon as possible"
48. message = generate\_error\_report(sender, receiver, subject, body)
49. send\_email(message)
50. **except** NameError:
51. **pass**