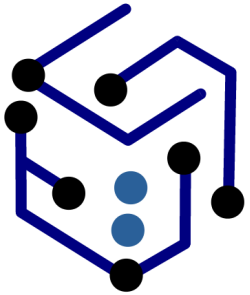
مجموعة الاهتمام بعلم البيانات

Data Science Special Interest Group



Summer Training

Weekly Report

Database for Arabic Handwritten Text

Report date in 2020/07/03

Fatima Abdullah Alogayyel, Imam Muhammad Ibn Saud Islamic University, fatmaalogayyel.300@gmail.com
Dareen Fayez Alluhaybi, Umm Al-Qura University, dareen.f.a@hotmail.com
Mona Aldebas, Imam Muhammad Ibn Saud Islamic University, Monamdbas@gmail.com
Mohammad Yahya Alghafli, King Abdulaziz University, m7md2012g@gmail.com

About this template

Purpose:	To report the group weekly activities in the project and related assignment.
How:	Edit and/or update all related sections needed to reflect the work progress. Use as many slides as needed to report progress in each section. Make copy of the previous week report and update it
Due data:	24 hours before weekly meeting
Team:	The group

Notes:

1. The editable document was created using LibreOffice 6 which can be download from https://www.libreoffice.org/download/ which must be used as a template for the report.
2. Upload the report to “Reports” folder in project’s area of Dropbox
3. Before uploading the report rename the file as follows: weeklyReport-Projectx-yyyymmdd.odp

Teaming and project selections

Team:

- Fatima Abdullah Alogayyel
- Dareen Fayez Alluhaybi
- Mona Aldebas
- Mohammad Yahya Alghafli

Project:

Create an Arabic handwritten Dataset contains images to different words written by volunteers from the community and creating a model for testing the dataset.

Tools identification and selection

Tools:

- Editor to create HTML code like notepad++, atom...etc
- Scanner
- Python editor likes spyder, atom...etc
- Office Timeline to create the timeline

Requirement gathering and analysis

Collect Requirements:

1- We used a google form to create the survey to attract the volunteer (https://docs.google.com/forms/d/e/1FAIpQLSc7V7jop5WEFSbkbzrDuKgHFyyVwsbxG7iHFa9-mz8u22wKcg/viewform?usp=sf_link).

2- Also, we use a google form to collect the image from the volunteer.

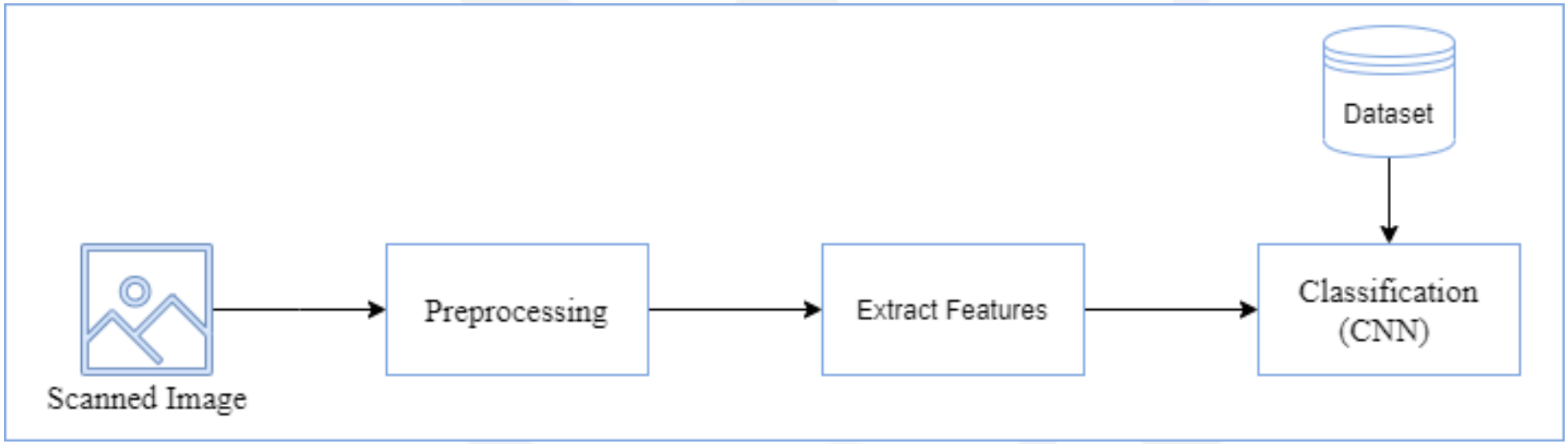
Analysis The Data:

1- We used the file contains the most frequently Arabic words used.

2- Then create a python program to filter the data based on the letters and their position in the words(middle, end, beginning).

Design

Architecture design:



Design ...

Implementation and coding

Code	Expected Result	Pass/Fail
Text Filter	Filter the file of most frequency Arabic words based on the letter and their positions.	Pass
CNN model	Print the class label of the image.	
Words Repetition	Printing the number of times the word repeated in the original file (most frequency Arabic words file)	Pass
Ground truth files generator	Create a ground truth for each image in the data-set	Pass
Image segmentation	Splitting an A4 to multiple images for each word	Pass
Image preprocessing	Extracting Gray image, threshold image, close image, and the original image for each word	Pass

Implementation and coding

Code	Expected Result	Pass/Fail
Export database images	This code exports the images on the database into a folder. Also, it counts number of images uploaded by each writer and update this value on the database.	Pass
Website 07/14 updates	We added login page and a page for the writers to upload the image. Also, we fixed some issues.	Pass
Website 07/21 updates	We added a page contains all the handwritten samples and we allowed the user to see his progress and achievements. Also, build a functionality where we could download and upload through the website.	Pass
Image segmentation (updated version)	In this version, we made the program export all multiple files at one time.	Pass

Implementation and coding

Code	Expected Result	Pass/Fail
Website 07/29 updates	Refactoring the code and activate the data-set button functionality	Pass
Updating the Ground Truth generator	We updated the current program to follow the same data-set structure, and we added the writer ID on each ground truth file.	Pass
Data-set folders generator	A program that generates the folders for each word on different data-sets test, metaData_test, , train, and metaData_train	Pass
Image classification	This program use external API to classify the images into their folders.	Pass
Manual Image classification	This program classify the images based on the user recognition for that word.	Pass
Website 08/04 updates	Create a page for special participants	Pass

Implementation and coding

Code	Expected Result	Pass/Fail
Dataset converter to CSV file	This program will make the convert our data-set into CSV file.	Pass
Website 08/11 updates	Fix some issues, adding contribution page, and FAQ page.	Pass

Implementation and coding ...

Text Filter:

```python
import re
with open('arabicWordFrequency.txt', "r") as reader:
    def write_for_char_start(char):
        reader.seek(0)
        writer = open("Start/"+char+".txt", "w")
        for line in reader:
            if line[0] == char or line[0:3] == "ال"+char:
                filtered_text = re.sub(r"[\d|\s]*", "", line)
                writer.write(filtered_text)
                writer.write("\n")
        writer.close()

    def write_for_char_middle(char):
        reader.seek(0)
        writer = open("Middle/"+char+".txt", "w")
        for line in reader:
            filtered_text = re.sub(r"[\d|\s]*", "", line)
            if bool(re.match(r".+"+char+r".+", filtered_text)):
                writer.write(filtered_text)
                writer.write("\n")
        writer.close()

    def write_for_char_end(char):
        reader.seek(0)
        writer = open("End/"+char+".txt", "w")
        for line in reader:
            filtered_text = re.sub(r"[\d|\s]*", "", line)
            if filtered_text[-1] == char:
                writer.write(filtered_text)
                writer.write("\n")
        writer.close()

```

Figure 1

```python
    write_for_char_start("ا")
    write_for_char_start("أ")
    write_for_char_start("آ")
    write_for_char_start("ب")
    write_for_char_start("ت")
    write_for_char_start("ث")
    write_for_char_start("ج")
    write_for_char_start("ح")
    write_for_char_start("خ")
    write_for_char_start("د")
    write_for_char_start("ذ")
    write_for_char_start("ر")
    write_for_char_start("ز")
    write_for_char_start("س")
    write_for_char_start("ش")
    write_for_char_start("ص")
    write_for_char_start("ض")
    write_for_char_start("ط")
    write_for_char_start("ظ")
    write_for_char_start("ع")
    write_for_char_start("غ")
    write_for_char_start("ف")
    write_for_char_start("ق")
    write_for_char_start("ك")
    write_for_char_start("ل")
    write_for_char_start("م")
    write_for_char_start("ن")
    write_for_char_start("ه")
    write_for_char_start("و")
    write_for_char_start("ي")

```

Figure 2

Implementation and coding ...

Text Filter:

```
53
54      write_for_char_middle("ا")
55      write_for_char_middle("أ")
56      write_for_char_middle("آ")
57      write_for_char_middle("ب")
58      write_for_char_middle("ت")
59      write_for_char_middle("ث")
70      write_for_char_middle("ج")
71      write_for_char_middle("ح")
72      write_for_char_middle("خ")
73      write_for_char_middle("د")
74      write_for_char_middle("ذ")
75      write_for_char_middle("ر")
76      write_for_char_middle("ز")
77      write_for_char_middle("س")
78      write_for_char_middle("ش")
79      write_for_char_middle("ص")
80      write_for_char_middle("ض")
81      write_for_char_middle("ط")
82      write_for_char_middle("ظ")
83      write_for_char_middle("ع")
84      write_for_char_middle("غ")
85      write_for_char_middle("ف")
86      write_for_char_middle("ق")
87      write_for_char_middle("ك")
88      write_for_char_middle("ل")
89      write_for_char_middle("م")
90      write_for_char_middle("ن")
91      write_for_char_middle("ه")
92      write_for_char_middle("و")
93      write_for_char_middle("ي")
```

Figure 3

```
94
95      write_for_char_end("ا")
96      write_for_char_end("أ")
97      write_for_char_end("آ")
98      write_for_char_end("ب")
99      write_for_char_end("ت")
00      write_for_char_end("ث")
01      write_for_char_end("ج")
02      write_for_char_end("ح")
03      write_for_char_end("خ")
04      write_for_char_end("د")
05      write_for_char_end("ذ")
06      write_for_char_end("ر")
07      write_for_char_end("ز")
08      write_for_char_end("س")
09      write_for_char_end("ش")
10      write_for_char_end("ص")
11      write_for_char_end("ض")
12      write_for_char_end("ط")
13      write_for_char_end("ظ")
14      write_for_char_end("ع")
15      write_for_char_end("غ")
16      write_for_char_end("ف")
17      write_for_char_end("ق")
18      write_for_char_end("ك")
19      write_for_char_end("ل")
20      write_for_char_end("م")
21      write_for_char_end("ن")
22      write_for_char_end("ه")
23      write_for_char_end("و")
24      write_for_char_end("ي")
```

Figure 4

Testing

Text Filter:

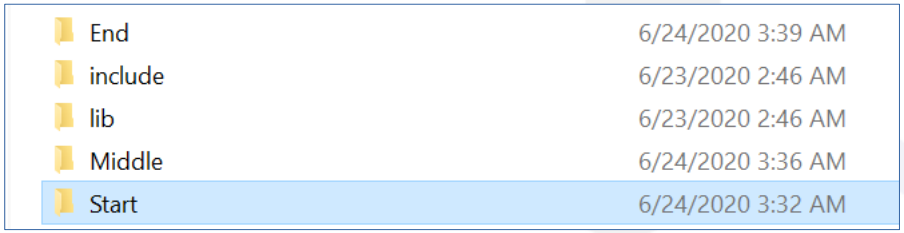


Figure 1

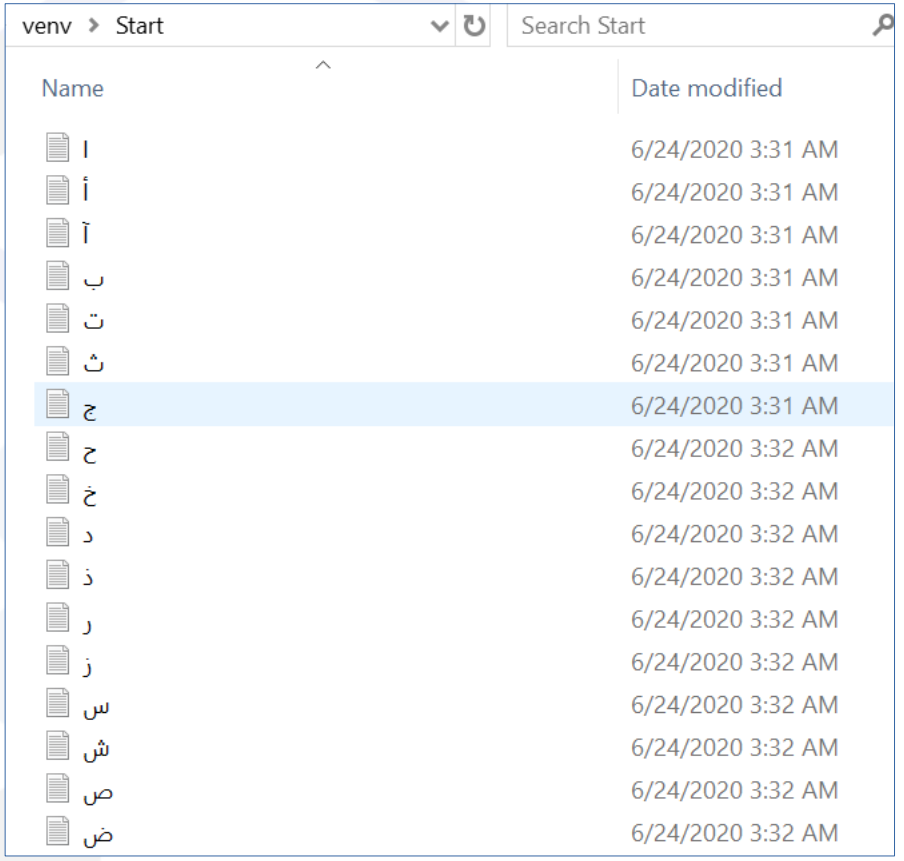


Figure 2

Implementation and coding ...

Words Repetition:

```
# -*- coding: utf-8 -*-


# First_file------------> الكلمات بعد الفرز
# Second_file---------->كل الكلمات بتكرارها
# New_file---------->الكلمات بعد الفرز مع تكرارها
from __future__ import unicode_literals

First_file = open("C://Users//LENOVO//Desktop//Data science special interest group//Database for Arabic Handwritten Text//arabicWordFrequency//tessst.txt", "r", encoding='utf-8')
#Second_file = open("C://Users//LENOVO//Desktop//Data science special interest group//Database for Arabic Handwritten Text//arabicWordFrequency//arabicWordFrequency.txt", "r", encoding='utf-8')
New_file = open("C://Users//LENOVO//Desktop//Data science special interest group//Database for Arabic Handwritten Text//arabicWordFrequency//New_file.txt" , "w+" , encoding='utf-8')

flag=True

for x in First_file:
    Second_file = open( "C://Users//LENOVO//Desktop//Data science special interest group//Database for Arabic Handwritten Text//arabicWordFrequency//arabicWordFrequency.txt",
        "r", encoding='utf-8')
    for y in Second_file:
        list=y.split()
        temp = x[:-1]
#        print("x:",x, "list[0]:",list[0])
 #       print("x:", type(x), "list[0]:", type(list[0]))
  #      print("x:", len(x), "list[0]:", len(list[0]))
        if list[0] == temp :
            print("first if")
            New_file.write(y)
            flag=False
        #check
        if not flag:
            flag=True
            Second_file.close()
            break


# close the file
First_file.close()
New_file.close()

```

Figure 1

Testing

Words Repetition:

لا	7400014
كما	1919037
أبها	1182
إلى	7481467
ألا	331704
آخر	597442
الأعضاء	238443
قرأ	98045
آية	156569
مسألته	2232
ءأنتم	138
القرى	31929
وأنا	207007
العظمى	31397
وحكى	30249
فإن	1488209
الدعوى	30022
الفتى	28534
وإني	40853
الآن	423501
جزء	106182
وآله	77901
شئ	121840
الإجراءات	98447
أولئك	87924
سئل	73616
نتائج	75622
جعفر	286051
الجمعية	338145
الحج	158406
جريج	77468
جل	151320
يحتاج	84964
التجارة	127971
جاء	382670
الحجاج	70545
جاز	95952
الرجال	160199
الوجه	135844
أجل	530465

Figure 1

جاز	95952
الرجال	160199
الوجه	135844
أجل	530465
المسيح	43131
الحديث	846249
الريح	42034
يسمح	30202
حي	56495
المصباح	12886
يحمل	64779
ويباح	1623
حاجة	124847
يطرح	7449
حالات	106281
والمدح	1534
حال	268047
حياته	51690
أحاديث	77141
فرحة	3215
خلال	415848
خير	275415
خرج	177552
إلخ	186537
خيرا	157271
المخ	3809
ترسيخ	3216
وشيخ	5641
الخير	136318
التواريخ	4781
خارج	102167
يخرج	136001
الصواريخ	3846
خالف	28857
خان	22104
مناخ	6515
يصرخ	3835
فرخ	2011
الأشخاص	102696
الخامس	68516
مخالف	25783
الناس	1089457
الحسن	562090

Figure 2

Implementation and coding …

Ground Truth generator:

```python
import os

images=[]
groundtruth=[]
path1='pictures' #need to have folder call pictures
#path11='pictures/الخليج'
path2='GroundTruth' #need to have folder call Groundtruth

def Num_of_dots(word):
    Dictionary_dots = {"ب": 1, "ت": 2, "ث": 3,
                       "ج": 1, "خ": 1, "ذ": 1, "ز": 1,
                       "ش": 3, "ض": 1,
                       "ظ": 1, "غ": 1,
                       "ف": 1, "ق": 2,
                       "ن": 1, "ي": 2
                       }

    count=0
    for x in word:
        if x in Dictionary_dots:
            count+=Dictionary_dots[x]

    print("Number of dots:",count)

    return count

def Num_of_possition(word):
    Dictionary_letters = {"أ": 1, "ا": 1, "آ": 1,"ء": 1, "إ": 1,
                          "ب": 2, "ت": 3, "ث": 4,
                          "ج": 5, "ح": 6, "خ": 7,
                          "د": 8, "ذ": 9, "ر": 10, "ز": 11,
                          "س": 12, "ش": 13, "ص": 14, "ض": 15,
                          "ط": 16, "ظ": 17, "ع": 18, "غ": 19,
                          "ف": 20, "ق": 21, "ك": 22, "ل": 23,
                          "م": 24, "ن": 25, "ه": 26, "و": 27, "ي": 28
                          }
```

Figure 1

Implementation and coding ...

Ground Truth generator:

```
36
37      string_possition=""
38      for x in word:
39          if x in Dictionary_letters:
40              string_possition+= str(Dictionary_letters[x]) + "-"
41
42      string_possition= string_possition[:-1]
43      print(string_possition)
44      return string_possition
45
46  myList = os.listdir(path1)
47  myList2 = os.listdir(path2)
48  noOfClasses = len(myList)
49  print("total number of classes detected",len(myList))
50  for x in myList:
51      i =0
52      y = path1+"/"+str(x)
53      print("y",y)
54      print("x", x)
55      os.mkdir(path2+"/"+str(x))
56      myPicList = os.listdir(path1 + "/" + str(x))
57
58      print(x)
59      myPicList = os.listdir(path1+"/"+str(x))
60      for z in myPicList:
61
62          z2= os.path.splitext(os.path.basename(z))[0]
63          numberOfDot = Num_of_dots(x)
64          numberOfPosition = Num_of_possition(x)
65          file = open(path2 + "/" + str(x) + "/" + z2 + ".txt", "w" ,encoding='utf-8')
66          file.write('Word: '+x+'\n'
67                     +'Image Name: '+z+'\n'
68                     +'Number of letters: '+str(len(x))+"\n"
69                     +"Number of dots: "+str(numberOfDot)+'\n'
70                     +'Position of letters: '+numberOfPosition)
71
72          file.close()
73      print("ww",myPicList[i])
74      print("this ",os.listdir(path1+"/"+str(x)))
75
76  print("-----------------")
```

Figure 2

Testing

Ground Truth generator:

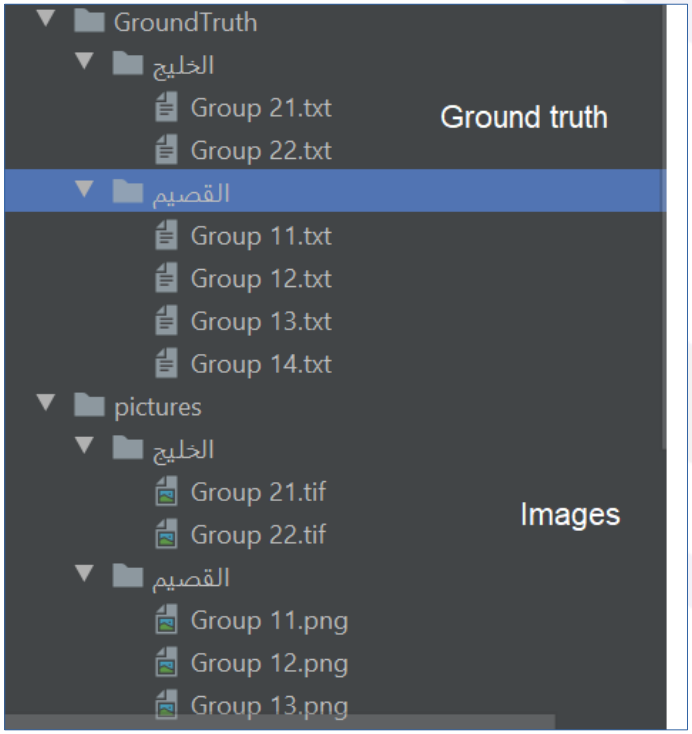


Figure 1

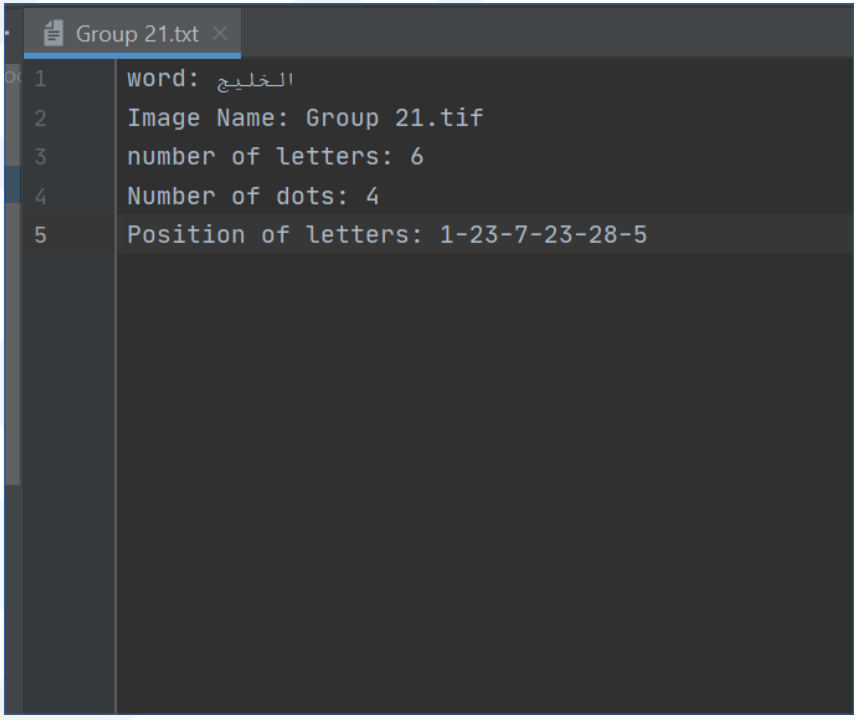


Figure 2

Implementation and coding ...

Image segmentation:

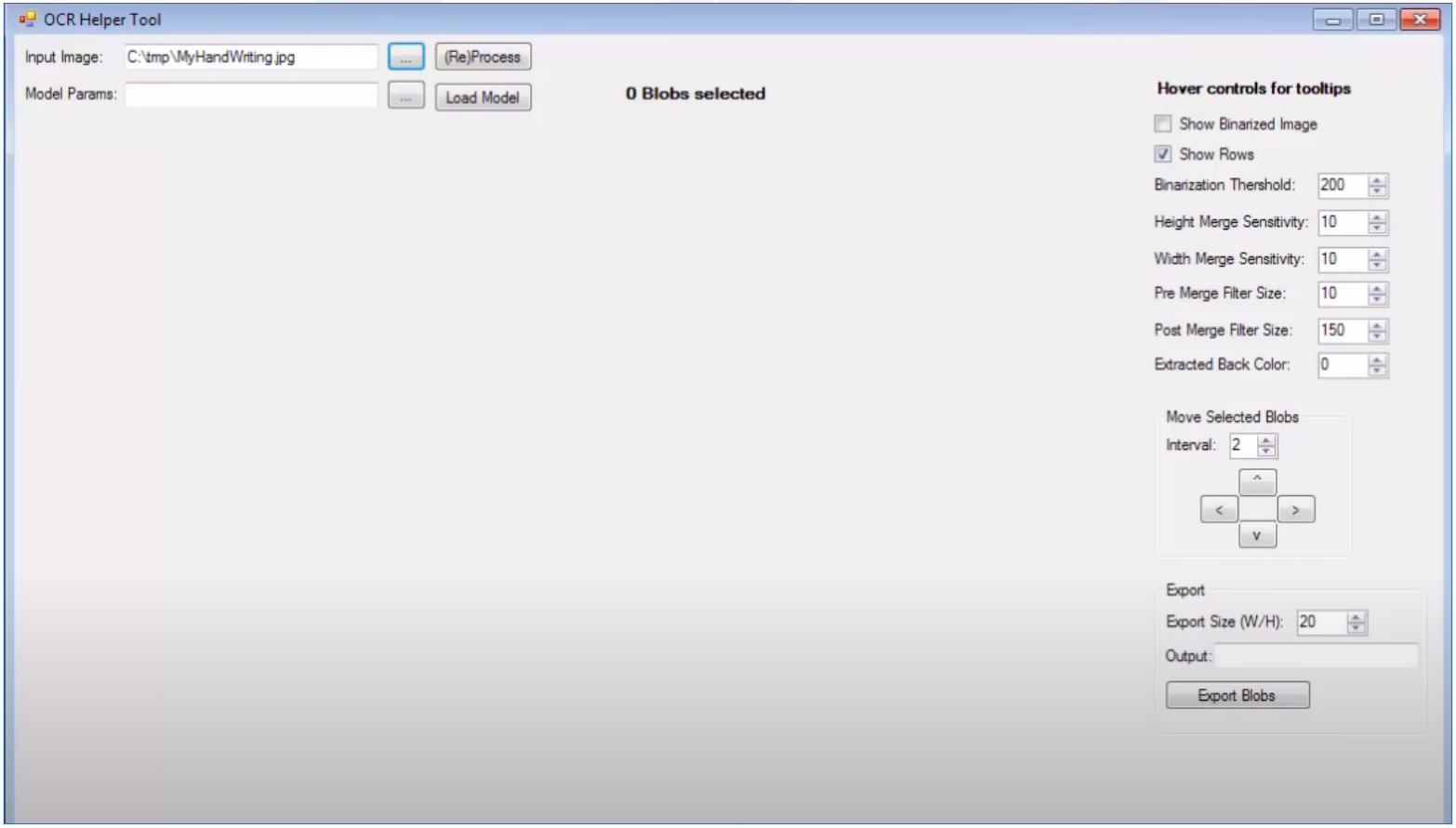


Figure 1

Implementation and coding ...

Image segmentation:

```python
import numpy as np
import cv2

import math

with open("outputfile3.txt", "r") as reader:
    counter = 1
    name_pattern = 1
    for line in reader:
        if counter % 3 == 1:
            image_list = list(eval(line)[:-1])  # this line of code will save the image into a list
            edge_size = int(math.sqrt(len(image_list)))  # we need this variable for reshape the numpy list
            image_np = np.array(image_list, np.uint8)
            image_np = image_np.reshape(edge_size, edge_size)  # now we reshaped the image to let it have the original dimensions
            image_np = np.transpose(image_np)
            file_path = "list_of_images/" + str(name_pattern) + ".jpg"
            cv2.imwrite(file_path, image_np)
            name_pattern += 1
        counter += 1
```

Figure 2

Testing

Image segmentation:

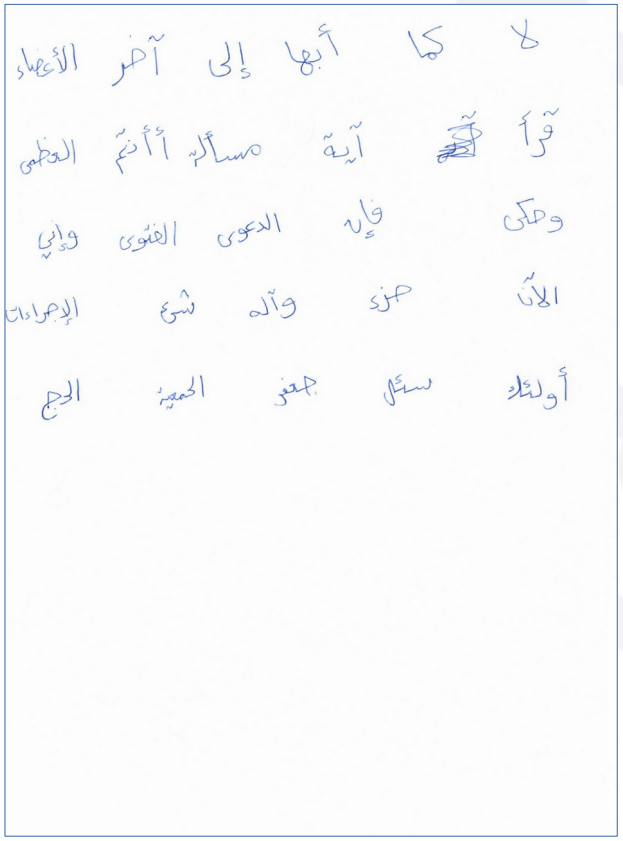


Figure 1

Figure 2

Implementation and coding ...

Image preprocessing:

```python
# -*- coding: utf-8 -*-
"""
Created on Wed Jun 17 20:00:51 2020

@author: LENOVO
"""

import cv2
import numpy as np


################################################################################


#img = cv2.imread("est.jpg")
#
#
#gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
#edges = cv2.Canny(gray, 75, 150)
#
#lines = cv2.HoughLinesP(edges, 1, np.pi/180, 50)
#
#lines = cv2.HoughLinesP(edges, 1, np.pi/180, 30, maxLineGap=250)

################################################################################ working fine

image = cv2.imread("test-600.png")
cv2.imshow('Orginal Image', image)
print(image)
cv2.waitKey()
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) #step 1--------------->convert image to grayscale
cv2.imshow('gray Image', gray)
cv2.waitKey()
sharpen_kernel = np.array([[-1,-1,-1], [-1,9,-1], [-1,-1,-1]])
sharpen = cv2.filter2D(gray, -1, sharpen_kernel)
thresh = cv2.threshold(sharpen, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)[1]
cv2.imshow('thresh', thresh)
cv2.waitKey()

#dose not work
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (4,4))#befor------------>(3,3)
close = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, kernel, iterations=1)
cv2.imshow('close', close)
cv2.waitKey()
result = 255 - close
cv2.imshow('result', result)
cv2.waitKey()


```

Figure 1

```python
##Image Erosion
#kernel = np.ones((4,4),np.uint8)#before--------------->(2,2)
#erosion = cv2.erode(result,kernel,iterations = 1)
#cv2.imshow('erosion', erosion)
#cv2.waitKey()
#
##Image Dilation
##dilation = cv2.dilate(result,kernel,iterations = 1)
##cv2.imshow('dilation', dilation)
##cv2.waitKey()
#
##Image opening
#opening = cv2.morphologyEx(erosion, cv2.MORPH_OPEN, kernel)
#cv2.imshow('opening', opening)
#cv2.waitKey()
#
#
##Image closing
#closing = cv2.morphologyEx(erosion, cv2.MORPH_CLOSE, kernel)
#cv2.imshow('closing', closing)
#cv2.waitKey()
#
#
#
#closing = cv2.equalizeHist(closing)
#cv2.imshow('equalizeHist', closing)
#cv2.waitKey()
#closing = closing/255 #its normalized values will be R/S, G/S and B/S (where, S=R+G+B).
#cv2.imshow('normalized', closing)
#cv2.waitKey()
#
##save image
#closing = cv2.convertScaleAbs(closing, alpha=(255.0))
#cv2.imwrite('savedImage_3.png', closing)


#cv2.imshow('sharpen', sharpen)
#cv2.imshow('thresh', thresh)
#cv2.imshow('close', close)
#cv2.imshow('result', result)
#cv2.waitKey()


################################################################################ dose not work

#
#img = cv2.imread("est.jpg", 0)
#ret, thresh = cv2.threshold(img, 10, 255, cv2.THRESH_OTSU)
#
#print ("Threshold selected : ", ret)
#cv2.imwrite("./debug.jpg", thresh)

```

Figure 2

Implementation and coding ...

Image preprocessing:

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jun 29 06:25:33 2020

@author: LENOVO
"""

from PIL import Image


im = Image.open("savedImage_3.png")
im.save("savedImage_3-300dpi.png", dpi=(300,300))#save image




mm
```

Figure 3

Testing

Image preprocessing:

Figure 1

Implementation and coding ...

Export database images:

```python
import mysql.connector
from mysql.connector import Error

def writeTofile(data, filename):
    # Convert binary data to proper format and write it on Hard Disk
    with open(filename, 'wb') as file:
        file.write(data)
    print("Stored blob data into: ", filename, "\n")

def readBlobData():
    try:
        connection = mysql.connector.connect(host='localhost',
                                             database='eyfad',
                                             user='root',
                                             password='Dareen588588')
        if connection.is_connected():
            db_Info = connection.get_server_info()
            print("Connected to MySQL Server version ", db_Info)
            cursor = connection.cursor()

        sql_fetch_blob_query = "SELECT * FROM image"
        cursor.execute(sql_fetch_blob_query)
        record = cursor.fetchall()
        for row in record:
            print("Id = ", row[0], "Name = ", row[1])
            name = row[1]
            photo = row[2]
            id= row[0]
            #resumeFile = row[3]

            print("Storing employee image and resume on disk \n")
            photoPath = "C:\\Users\\HP\\Desktop\\test2\\word" + str(id) + "_" + str(name)+  ".jpg"
            writeTofile(photo, photoPath)



        cursor.close()
    except Error as e:

        print("Error while connecting to MySQL", e)

        #cursor.close()
```

Figure 1

```python
def numOfWord():
    try:
        connection = mysql.connector.connect(host='localhost',
                                             database='eyfad',
                                             user='root',
                                             password='Dareen588588')
        if connection.is_connected():
            db_Info = connection.get_server_info()
            print("Connected to MySQL Server version ", db_Info)
            cursor = connection.cursor()

            sql_select_query = "SELECT volunteerID FROM image"
            cursor = connection.cursor()
            cursor.execute(sql_select_query)
            record = cursor.fetchall()
            print("volunteerID ", record)
            for x in record:
                print(x[0])
            #cursor.close()
                sql_select_query = "SELECT numOfWords FROM volunteer WHERE  volunteerID= %s"
                val = (x)
                cursor = connection.cursor()
                cursor.execute(sql_select_query,val)
                record = cursor.fetchall()
                print("num of word ", record)
                for y in record:
                 print(y[0])
                 oldword=int(y[0])
                 print("----------")
                 print(type(oldword))
                 #oldword=0;
                 zz= oldword+40
                 print(type(zz),zz)
                 sql = "UPDATE volunteer SET numOfWords = %s WHERE volunteerID = %s"
                 val2 = (zz, x[0])

                 cursor.execute(sql, val2)
                 connection.commit()

                 print(cursor.rowcount, "record(s) affected")


        cursor.close()
    except Error as e:

        print("Error while connecting to MySQL", e)

        #cursor.close()

readBlobData()
numOfWord()
```

Figure 2

Testing

Export database images:

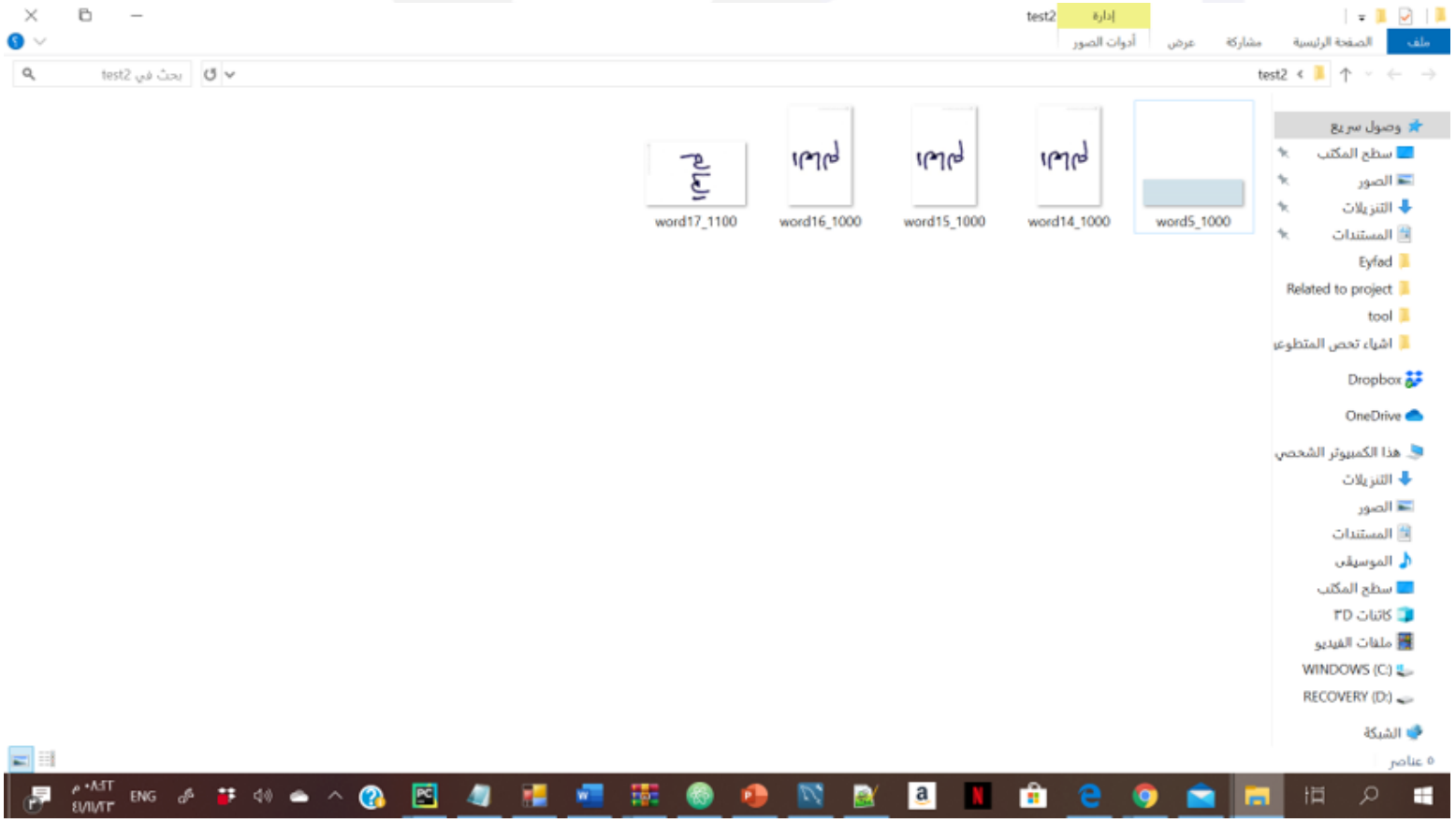


Figure 1

Implementation and coding ...

Website 07/14 update:

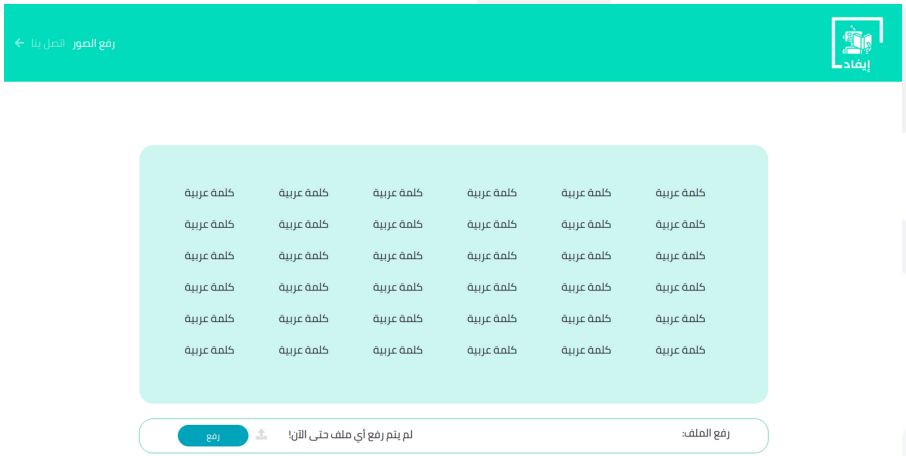


Figure 1

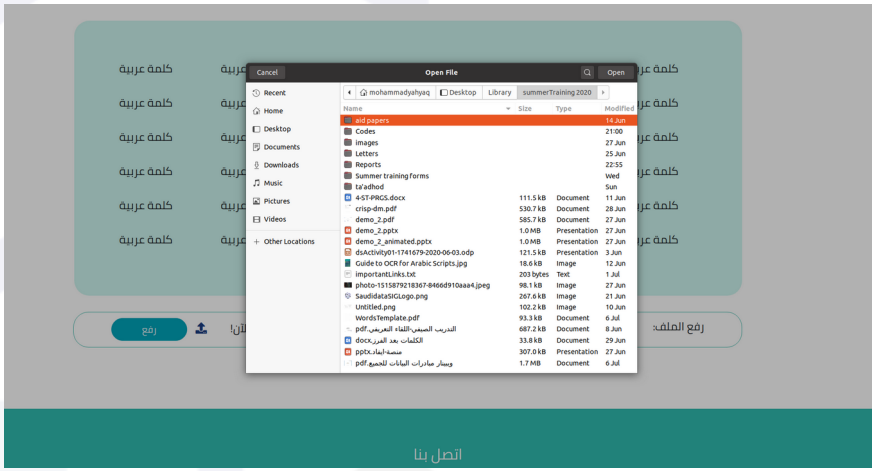


Figure 2

Figure 3

Figure 4

Implementation and coding ...

Website 07/14 update:

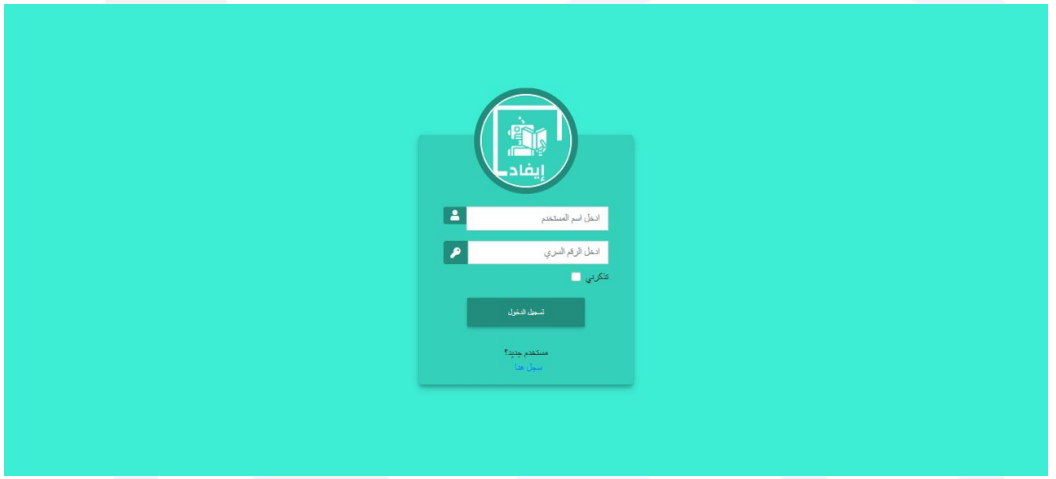


Figure 5

Figure 6

Implementation and coding ...

Website 07/21 update (uploading page):

Figure 1

Figure 2

Implementation and coding ...

Website 07/21 update (greeting based on account name):

Figure 1

Figure 2

Implementation and coding ...

Website 07/21 update (menu displays the user progress):

Figure 1

Figure 2

Implementation and coding ...

Website 07/21 update (handwritten samples page):

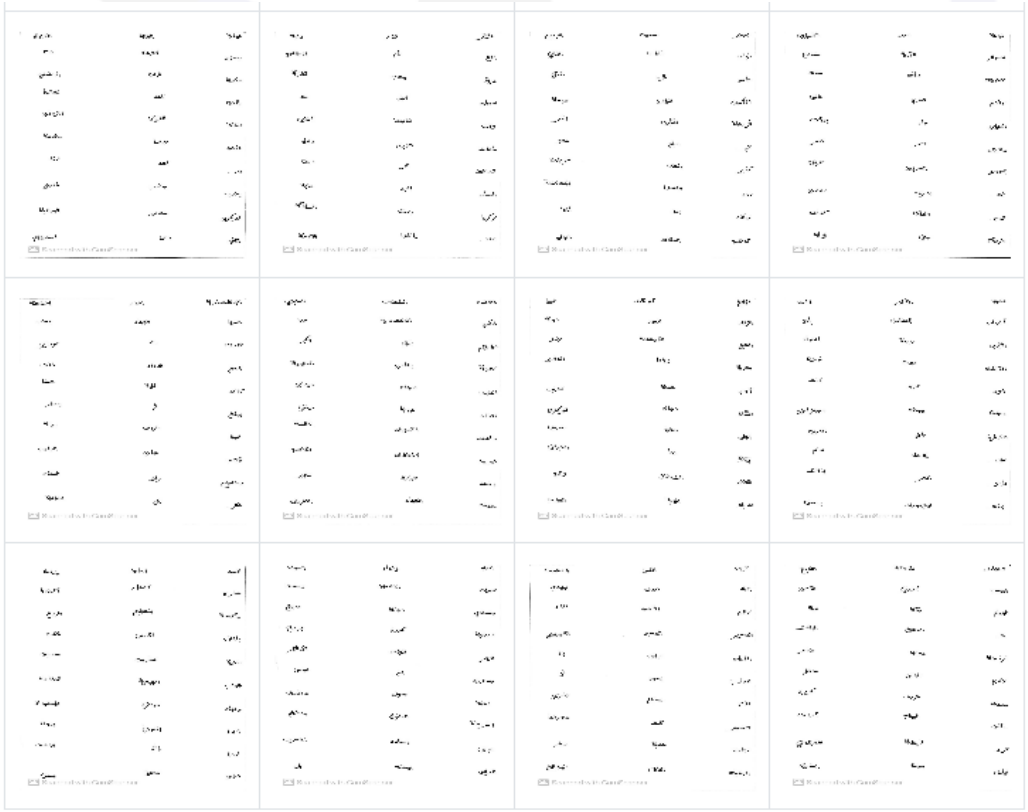


Figure 1

Implementation and coding ...

Website 07/21 update (downloading files into folder):

Figure 1

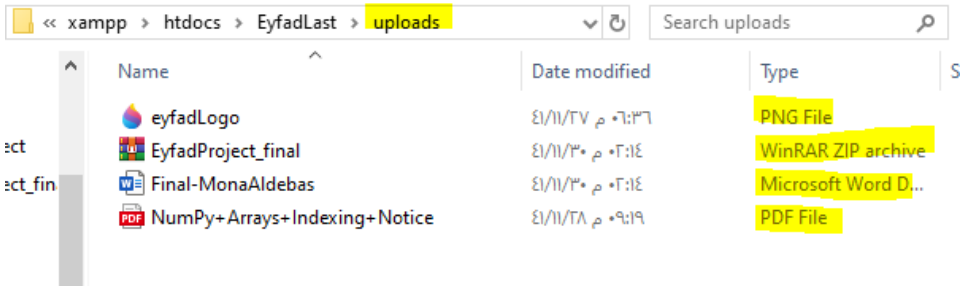


Figure 2

Implementation and coding …

Website 07/21 update (uploading files and it will be removed from the website and the folder):

Figure 1

Figure 2

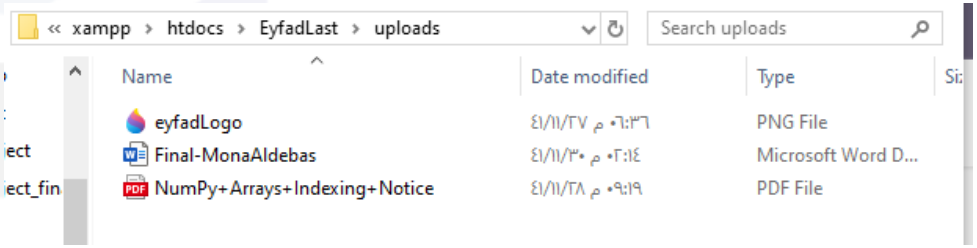


Figure 3

Implementation and coding ...

Image segmentation (updated version):

```python
import numpy as np
import cv2
import os

import math

files_list = os.listdir('template images')
# print(files_list)

for template in files_list:
    counter = 1
    imageCounter = 1
    with open("template images/" + template) as reader:
        for line in reader:
            name_pattern = template[:-4] + str(imageCounter)

            if counter % 3 == 1:
                image_list = list(eval(line)[:-1])  # this line of code will save the image into a list
                edge_size = int(math.sqrt(len(image_list)))  # we need this variable for reshape the numpy list
                image_np = np.array(image_list, np.uint8)
                image_np = image_np.reshape(edge_size,
                                            edge_size)  # now we reshaped the image to let it have the original dimensions
                # image_np = fix_rows(image_np)
                image_np = np.transpose(image_np)
                # image_np = fix_rows(image_np)
                # file_path = "list_of_images/" + str(name_pattern) + "a.png"
                # cv2.imwrite(file_path, image_np)
                # image_np = np.where(image_np == 0, 255, image_np)
                file_path = "list_of_images/" + name_pattern + ".jpg"
                cv2.imwrite(file_path, image_np)
                imageCounter += 1
            counter += 1

```

Figure 1

Implementation and coding ...

Image segmentation (updated version):

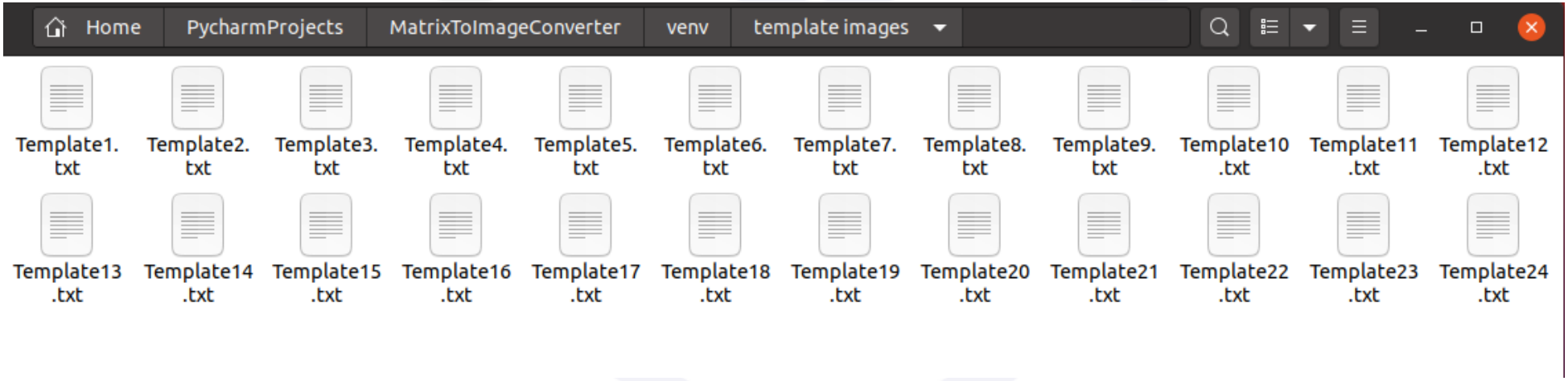


Figure 2

Testing

Image segmentation (updated version):

Figure 1

Testing

Image segmentation (updated version):

Figure 2

Testing

Image segmentation (updated version):

Figure 2

Implementation and coding ...

Website 07/29 update:

Figure 1

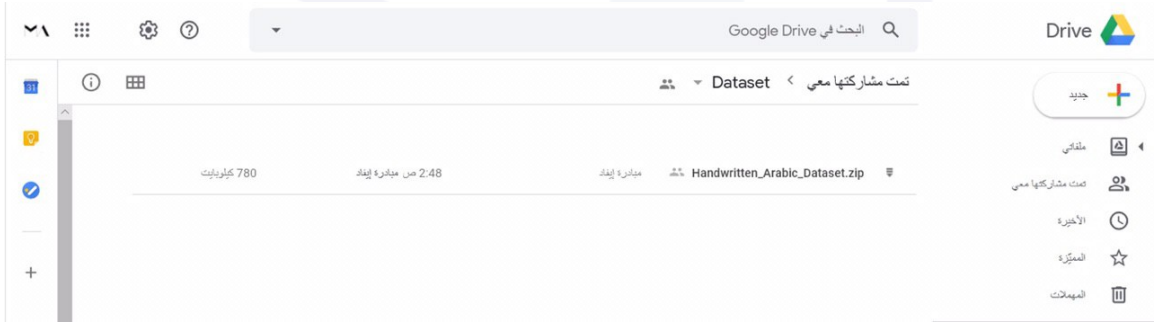


Figure 2

Implementation and coding ...

Updating the ground truth generator:

```python
import os

images=[]
groundtruth=[]
path1='train' #need to have folder call train
path2='metaData_train' #need to have folder call metaData_train

path3='test' #need to have folder call test
path4='metaData_test' #need to have folder call metaData_test
def Num_of_dots(word):
    Dictionary_dots = {"ب": 1, "ت": 2, "ث": 3,
                       "ج": 1, "خ": 1, "ذ": 1, "ز": 1,
                       "ش": 3, "ض": 1,
                       "ظ": 1, "غ": 1,
                       "ف": 1, "ق": 2,
                       "ن": 1, "ي": 2
                       }


    count=0
    for x in word:
        if x in Dictionary_dots:
            count+=Dictionary_dots[x]

    print("Number of dots:",count)

    return count

def Num_of_possition(word):
    Dictionary_letters = {"أ": 1, "ا": 1, "آ": 1, "ء": 1, "إ": 1,
                       "ب": 2, "ت": 3, "ث": 4,
                       "ج": 5, "ح": 6, "خ": 7,
                       "د": 8, "ذ": 9, "ر": 10, "ز": 11,
                       "س": 12, "ش": 13, "ص": 14, "ض": 15,
                       "ط": 16, "ظ": 17, "ع": 18, "غ": 19,
                       "ف": 20, "ق": 21, "ك": 22, "ل": 23,
                       "م": 24, "ن": 25, "ه": 26, "و": 27, "ي": 28
                       }

    string_possition=""
    for x in word:
        if x in Dictionary_letters:
            string_possition+= str(Dictionary_letters[x]) + "-"

    string_possition= string_possition[:-1]
    print(string_possition)
    return string_possition

```

Figure 1

Implementation and coding ...

Updating the ground truth generator:

```
46
47 def ground_truth_files(path1,path2):
48     myList = os.listdir(path1)
49     myList2 = os.listdir(path2)
50     noOfClasses = len(myList)
51     print("total number of classes detected", len(myList))
52     for x in myList:
53         i = 0
54         y = path1 + "/" + str(x)
55         print("y", y)
56         print("x", x)
57         os.mkdir(path2 + "/" + str(x))
58         myPicList = os.listdir(path1 + "/" + str(x))
59 
60         print(x)
61         myPicList = os.listdir(path1 + "/" + str(x))
62         for z in myPicList:
63             print("z", z)
64             z2 = os.path.splitext(os.path.basename(z))[0]
65             z3=z.split("_")
66             print("---------------7777-----",z3[0])
67             numberOfDot = Num_of_dots(x)
68             numberOfPosition = Num_of_possition(x)
69             file = open(path2 + "/" + str(x) + "/" + z2 + ".txt", "w", encoding='utf-8')
70             file.write('Word: ' + x + '\n'
71                        + 'Volunteer Identifier: ' + z3[0] + '\n'
72                        + 'Image Name: ' + z + '\n'
73                        + 'Number of letters: ' + str(len(x)) + "\n"
74                        + "Number of dots: " + str(numberOfDot) + '\n'
75                        + 'Position of letters: ' + numberOfPosition)
76 
77             file.close()
78 
79     print("-----------------")
80 
81 
82 ground_truth_files(path1,path2)
83 ground_truth_files(path3,path4)
```

Figure 2

Testing

Updating the ground truth generator:

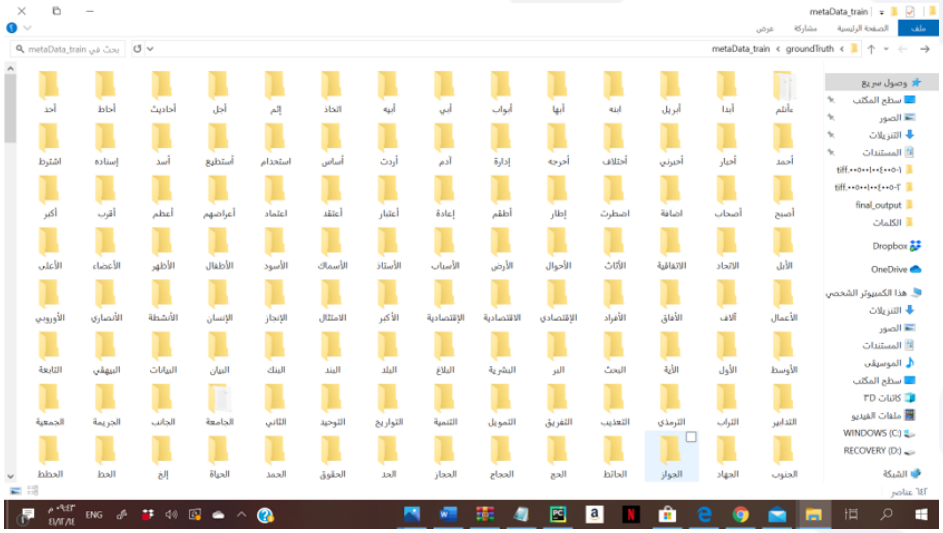


Figure 1

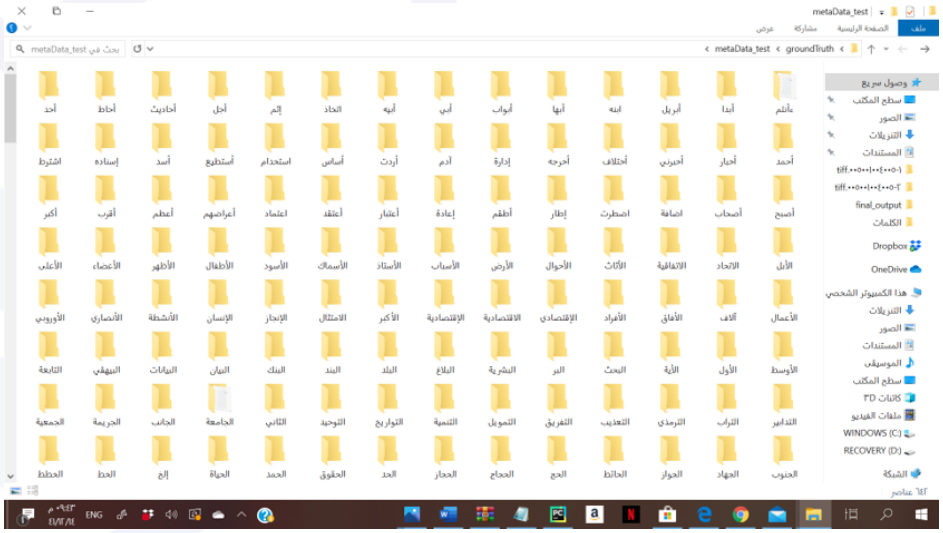


Figure 2

Testing

Updating the ground truth generator:

Figure 3

Implementation and coding ...

Data-set folders generator:

```python
import os

main_dirs = ["metaData_test", "metaData_train", "test", "train"]  # those are the subdirectories of the dataset file

os.mkdir("Handwritten_Arabic_Dataset")
for dir_name in main_dirs:
    os.mkdir("Handwritten_Arabic_Dataset/" + dir_name)
    with open("selected words.txt", "r") as reader:
        for line in reader:
            os.mkdir("Handwritten_Arabic_Dataset/" + dir_name + "/" + line[:-1])
```

Figure 1

Testing

Data-set folders generator:

Figure 1

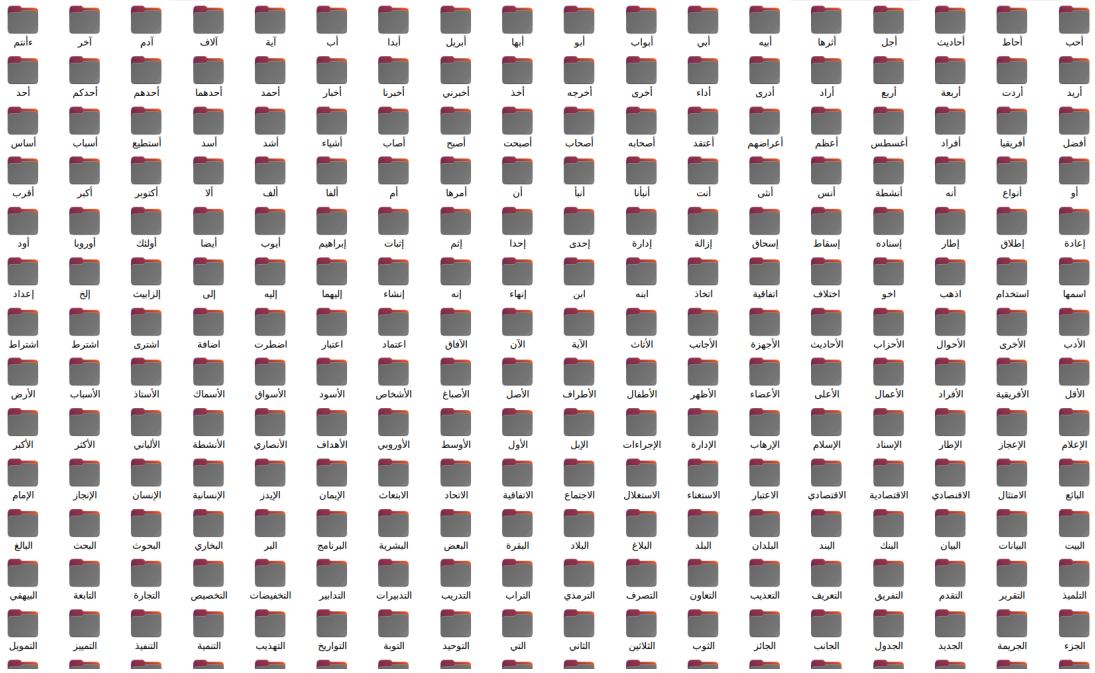


Figure 2

Implementation and coding ...

Image classification:

```python
import os
import io
import json
import numpy as np
import cv2
import requests


# this method will find the number for the word inside the selected words.txt file
def get_word_number(word):
    with open("selected words.txt", "r") as reader:
        counter = 1
        for line in reader:
            # print(counter, line[:-1])
            if line[:-1] == word:
                return counter
            counter += 1
        return -1


# # this peace of code for testing purposes
# while True:
#     word_number = get_word_number(input("Enter a word: "))
#     print(word_number)

writer_id = int(input("Enter the writer id: "))
set_type = input("Enter the set type: ")  # this gonna contain the set type (test, train, ...)

list_of_images = os.listdir("list_of_images")  # this list gonna contain
# print(list_of_images)

report_list = []  # this list gonna cotain the result of each image to report it at the end of the program
url_api = "https://api.ocr.space/parse/image"
for image in list_of_images:
    image_path = "list_of_images/" + image
    image_array = cv2.imread(image_path)  # this is the image exported in numpy array
    _, compressed_image = cv2.imencode(".jpg", image_array, [1, 90])
    file_bytes = io.BytesIO(compressed_image)  # this what we gonna pass it on the API

```

Figure 1

Implementation and coding ...

Image classification:

```
    result = requests.post(url_api, files={image_path: file_bytes}, data={"apikey": "helloworld", "language": "ara"})  # result is the delivered object from the API
    result = result.content.decode()  # this is the the text data in json format String
    result = json.loads(result)  # now we converted it into python data structure

    """
    This comment explains how to retrieve the data from the result
        result.get("ParsedResults")[0].get("ParsedText")
    """

    # # testing the tool
    # print(result.get("ParsedResults")[0].get("ParsedText"))
    # cv2.imshow("Tested Image", image_array)
    # cv2.waitKey(0)

    real_text = result.get("ParsedResults")[0].get("ParsedText").strip()
    image_correct_folder_path = "Handwritten_Arabic_Dataset/" + set_type + "/" + real_text
    if os.path.exists(image_correct_folder_path):
        word_number = get_word_number(real_text)
        if word_number != -1:
            exported_image = image_correct_folder_path + "/" + str(writer_id) + "-" + str(word_number) + ".jpg"
            cv2.imwrite(exported_image, image_array)
            report_list.append((exported_image, real_text))
        else:
            print("Error: the word " + user_word + " is not found on the file folder")
            exit()
    else:
        cv2.imshow(real_text, image_array)
        cv2.waitKey(0)

        print("Error: the word didn't recognised correctly by the OCR\n")
        while True:
            user_word = input("What was this word? ")
            new_path = "Handwritten_Arabic_Dataset/" + set_type + "/" + user_word
            if os.path.exists(new_path):
                word_number = get_word_number(user_word)
                if word_number != -1:
                    cv2.imwrite(new_path + "/" + str(writer_id) + "-" + str(word_number) + ".jpg", image_array)
                    report_list.append((new_path + "/" + str(writer_id) + "-" + str(word_number) + ".jpg", user_word))
                else:
                    print("Error: the word " + user_word + " is not found on the file folder")
                    exit()
```

Figure 2

Implementation and coding ...

Image classification:

```
81                break
82            else:
83                print("We didn't found this word in the data-set word")
84
```

Figure 3

Implementation and coding ...

Manual image classification:

```python
import os
import numpy as np
import cv2


# this method will find the number for the word inside the selected words.txt file
def get_word_number(word):
    with open("selected words.txt", "r", encoding='UTF8') as reader:
        counter = 1
        for line in reader:
            # print(counter, line[:-1])
            if line[:-1] == word:
                return counter
            counter += 1
        return -1


# # this peace of code for testing purposes
# while True:
#     word_number = get_word_number(input("Enter a word: "))
#     print(word_number)

writer_id = int(input("Enter the writer id: "))
set_type = input("Enter the set type: ").strip()  # this gonna contain the set type (test, train, ...)

list_of_images = os.listdir("list_of_images")  # this list gonna contain
# print(list_of_images)

report_list = []  # this list gonna cotain the result of each image to report it at the end of the program
for image in list_of_images:
    assert os.path.exists("list_of_images/" + image), "an image is not found"
    image_array = cv2.imread("list_of_images/" + image)
    cv2.imshow("Word", image_array)
    cv2.waitKey(0)

    word_number = -1
    is_misspelled = False  # this variable will be used to know if the word is misspelled or not
    user_word = ""
    while True:
        user_word = input("Enter the word you have seen: ").strip()
        if input == "0":
            is_misspelled = True
            break
        word_number = get_word_number(user_word)
        if word_number != -1:
            break
        else:
            print("\nThe word " + user_word + " is not exist on the dataset selected words...")
            print("Try again\n")
            cv2.imshow("Word", image_array)
            cv2.waitKey(0)
    if is_misspelled:
        continue
    assert word_number != -1, "word number is not calculated properly"
    cv2.imwrite("Handwritten_Arabic_Dataset/" + set_type + "/" + user_word + "/" + str(writer_id) + "-" + str(word_number) + ".jpg", image_array)
    os.remove("list_of_images/" + image)
```

Figure 1

Testing

Manual image classification:

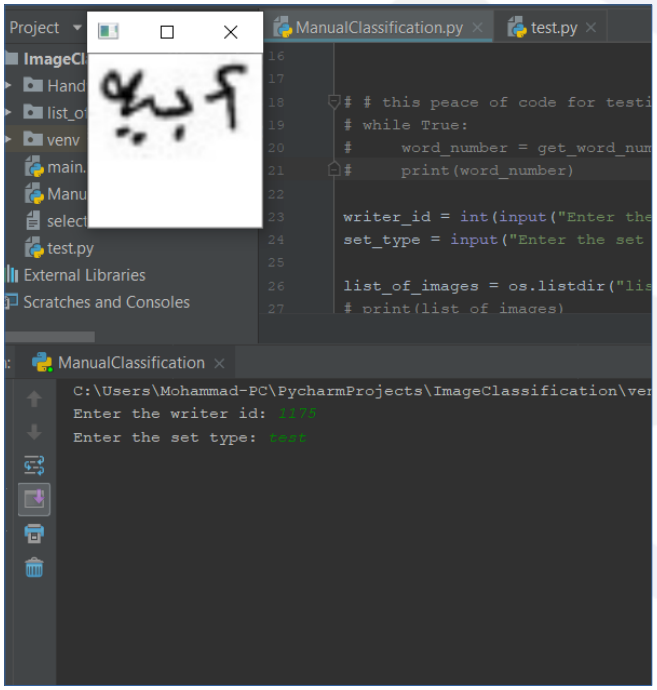


Figure 1

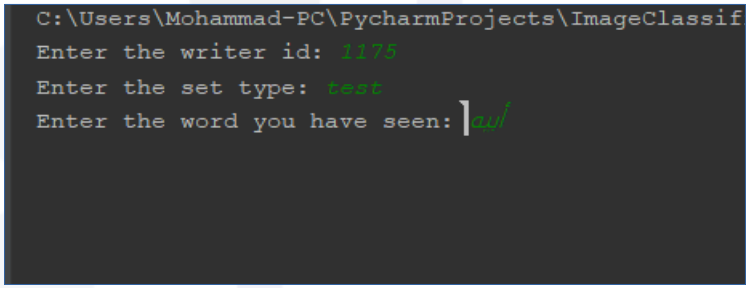


Figure 2

Testing

Manual image classification:

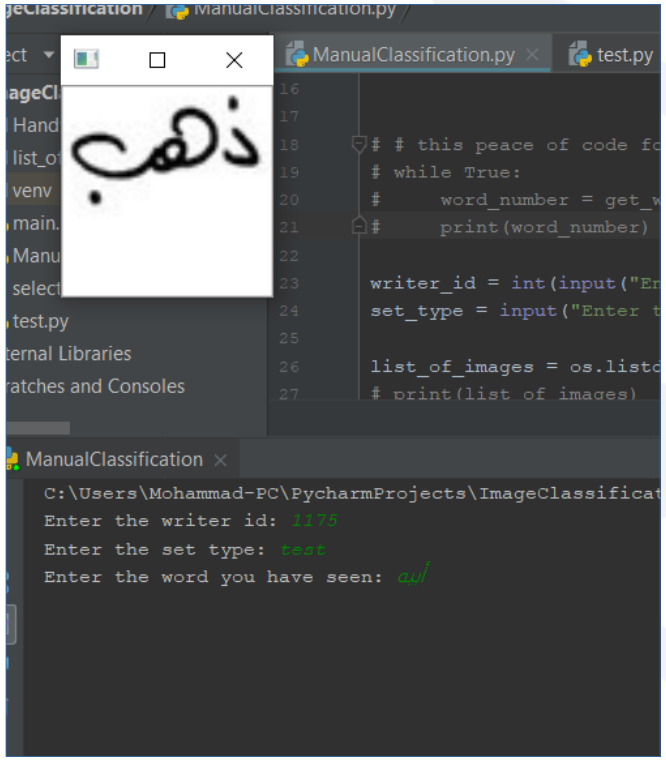


Figure 3

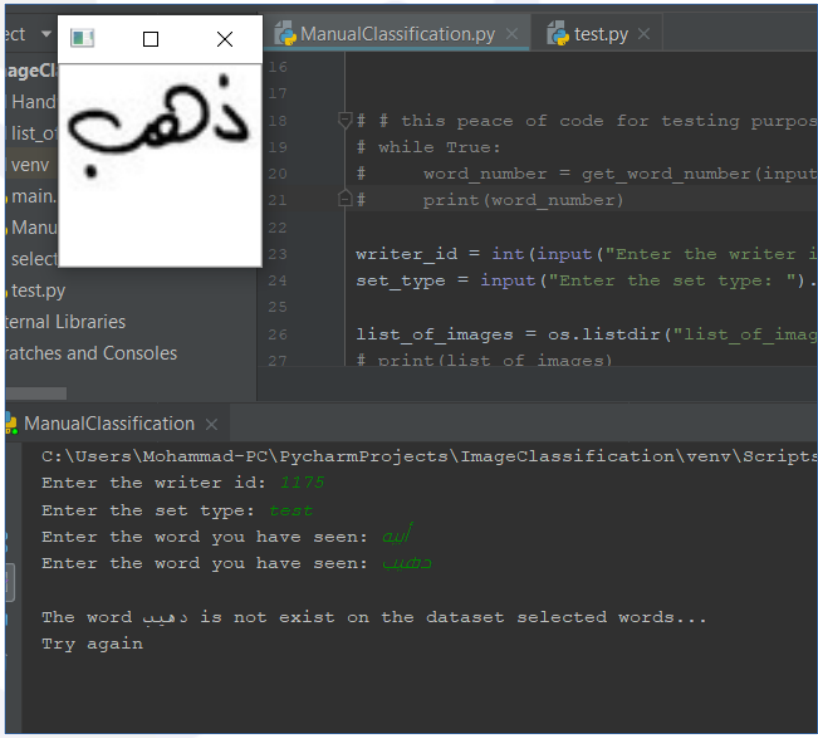


Figure 4

Implementation and coding ...

Website 08/04 update:

Create a page for participants and put their names with the prizes they own for their participation.

Figure 1

Implementation and coding ...

Website 08/04 update:

Figure 2

Implementation and coding ...

Website 08/04 update:

Figure 3

Implementation and coding ...

Dataset converter to CSV file:

```python
import ...

with open("EyfadDataset.csv", "w", newline="", encoding="utf-8") as f:

    def np_to_list(np_array):
        image_string = []
        dimensions = np_array.shape
        dimensions: Any = np_array.shape
                for k in range(dimensions[2]):
                    image_string.append(np_array[i][j][k])
        # for number in np_array:
        #     image_string += str(number)
        # image_string = image_string.replace("\n", " ")
        # image_string = image_string.replace(",", "")
        # image_string = image_string.replace("[", "{")
        # image_string = image_string.replace("]", "}")
        # print(image_string)
        # print(image_string[-1])
        return image_string[:-1]

    def find_word(selected_word_id):
        with open("selected words.txt", "r") as reader:
            count = 0
            for line in reader:
                if count == selected_word_id:
                    return line[:-1]
                count += 1


    dataset_writer = csv.writer(f)

    # first we should write the header
    dataset_writer.writerow(["writer_ID", "writer_name", "word_ID", "actual_word", "actual_image 128pxx128px (RGB 8bit)"])

    # now we will start to insert the data
    for folder in os.listdir("test"):
        if len(os.listdir("test/" + folder)) > 0:
            images_list = os.listdir("test/" + folder)
            for image in images_list:
                writer_id = int(re.split("[-.]", image)[0])
                writer_name = "Fatima Alogayyel"
                word_id = int(re.split("[-.]", image)[1])
                word = find_word(word_id)
                image_array = cv2.imread("test/" + folder + "/" + image)
                image_list = np_to_list(image_array)

                dataset_writer.writerow([writer_id, writer_name, word_id, word] + image_list)
```

Figure 1

Testing

Dataset converter to CSV file:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	A
1	writer_ID	writer_name	word_ID	actual_word	actual_image 128pxx128px (RGB 8bit)																																								
2	1175	Fatima Alogayyel	705	سمعت	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2
3	1175	Fatima Alogayyel	690	انتهى	254	254	254	255	255	255	255	255	255	254	254	254	255	255	255	255	255	255	254	254	254	255	255	255	255	255	255	255	255	255	255	255	255	255	254	254	254	254	254	2
4	1175	Fatima Alogayyel	65	التواريخ	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2
5	1175	Fatima Alogayyel	55	فرحة	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2
6	1175	Fatima Alogayyel	1356	جدا	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2
7	1175	Fatima Alogayyel	1562	الجهاد	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2
8	1175	Fatima Alogayyel	1361	المدينة	248	248	248	252	252	252	253	253	253	254	254	254	254	254	254	252	252	252	245	245	245	252	252	252	250	250	250	248	248	248	253	253	253	255	255	255	255	255	255	2
9	1175	Fatima Alogayyel	1585	قيد	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2
10	1175	Fatima Alogayyel	1047	بك	255	255	255	254	254	254	255	255	255	255	255	255	253	253	253	254	254	254	255	255	255	253	253	253	254	254	254	254	254	254	254	254	254	255	255	255	255	255	255	2
11	1175	Fatima Alogayyel	512	القيامة	250	250	250	250	250	250	251	251	251	252	252	252	254	254	254	254	254	254	254	254	254	254	254	254	254	251	251	251	250	250	250	250	250	250	251	251	251	252	252	2
12	1175	Fatima Alogayyel	383	وزارة	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2
13	1175	Fatima Alogayyel	87	النساء	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2
14	1175	Fatima Alogayyel	1074	الكبرى	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2
15	1175	Fatima Alogayyel	1533	التقدم	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	254	254	254	253	253	253	252	252	252	249	249	249	248	248	248	247	247	247	247	247	247	247	247	247	2
16	1175	Fatima Alogayyel	799	إطار	253	253	253	255	255	255	252	252	252	252	252	252	252	252	252	252	252	252	254	254	254	255	255	255	251	251	251	255	255	255	254	254	254	252	252	252	252	252	252	2
17	1175	Fatima Alogayyel	1369	الدورة	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2
18	1175	Fatima Alogayyel	148	الخاسة	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	254	254	254	254	254	2
19	1175	Fatima Alogayyel	834	شرط	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2
20	1175	Fatima Alogayyel	465	عبدالعزيز	254	254	254	254	254	254	254	254	254	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2
21	1175	Fatima Alogayyel	1419	البلد	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2
22	1175	Fatima Alogayyel	572	لغات	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2
23	1175	Fatima Alogayyel	838	وسط	255	255	255	255	255	255	254	254	254	254	254	254	254	254	254	254	254	254	254	254	254	254	254	254	254	254	254	254	254	254	254	254	254	254	254	254	254	255	255	2
24	1175	Fatima Alogayyel	272	معرفة	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2
25	1175	Fatima Alogayyel	379	الحجاز	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2
26	1175	Fatima Alogayyel	537	كفالة	255	255	255	254	254	254	255	255	255	252	252	252	254	254	254	255	255	255	253	253	253	254	254	254	253	253	253	252	252	252	255	255	255	251	251	251	254	254	254	2
27	1175	Fatima Alogayyel	67	يخرج	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2
28	1175	Fatima Alogayyel	3	إلى	255	255	255	253	253	253	254	254	254	255	255	255	253	253	253	249	249	249	250	250	250	254	254	254	254	254	254	254	254	254	254	254	254	253	253	253	252	252	252	2
29	1175	Fatima Alogayyel	752	منكر	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2
30	1175	Fatima Alogayyel	300	الحارث	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	254	254	254	255	255	255	255	255	255	255	255	255	255	255	255	2
31	1175	Fatima Alogayyel	1377	داخلية	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2
32	1175	Fatima Alogayyel	1392	درهم	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2
33	1175	Fatima Alogayyel	1599	جديتا	254	254	254	253	253	253	252	252	252	252	252	252	254	254	254	254	254	254	254	254	254	254	254	254	253	253	253	252	252	252	250	250	250	250	250	250	253	253	253	2
34	1175	Fatima Alogayyel	1544	دعت	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2
35	1175	Fatima Alogayyel	722	رضي	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2
36	1175	Fatima Alogayyel	107	شكرا	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	254	254	254	255	255	2
37	1175	Fatima Alogayyel	94	عيسى	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2
38	1175	Fatima Alogayyel	874	خطوط	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	254	254	254	254	254	254	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2
39	1175	Fatima Alogayyel	303	تؤثر	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2
40	1175	Fatima Alogayyel	1153	السوالب	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2
41	1175	Fatima Alogayyel	539	الملائكة	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2
42	1175	Fatima Alogayyel	1178	الربيع	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2
43	1175	Fatima Alogayyel	1084	التدبيرات	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2
44	1175	Fatima Alogayyel	1009	بعضهم	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	2

Figure 1

Implementation and coding ...

Website 08/11 update:

Fixing the header problem.

Figure 1

Figure 2

Implementation and coding ...

Website 08/11 update:

Fixing the email link problem.

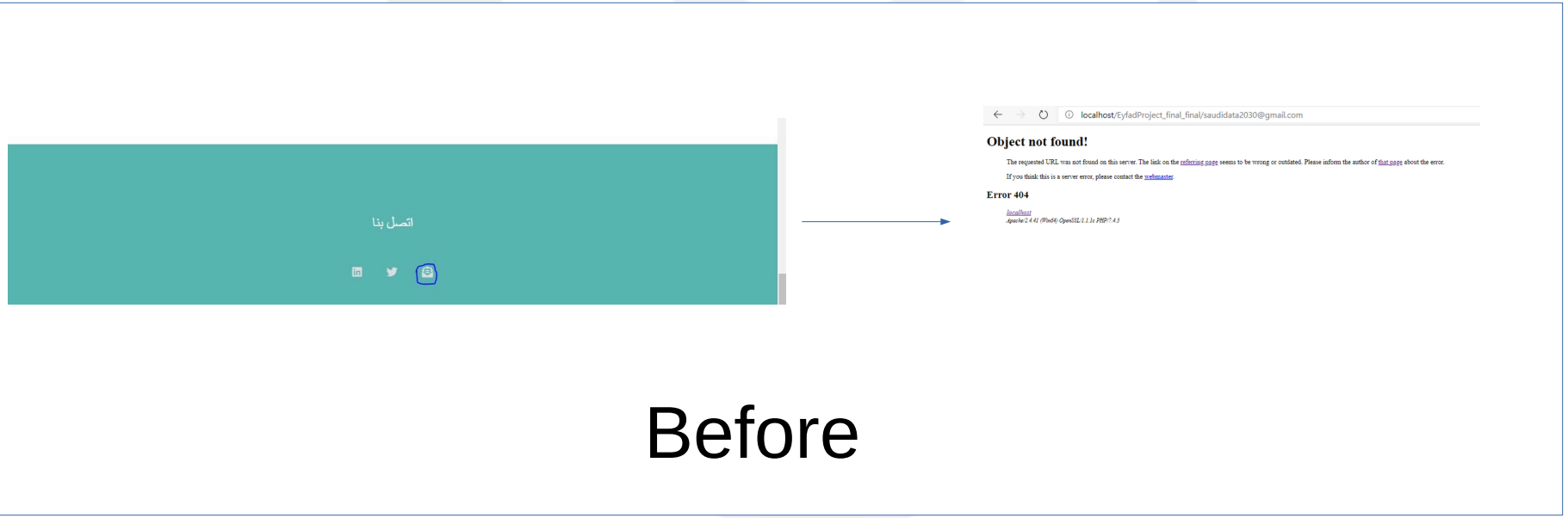


Figure 1

Implementation and coding ...

Website 08/11 update:

Fixing the email link problem.

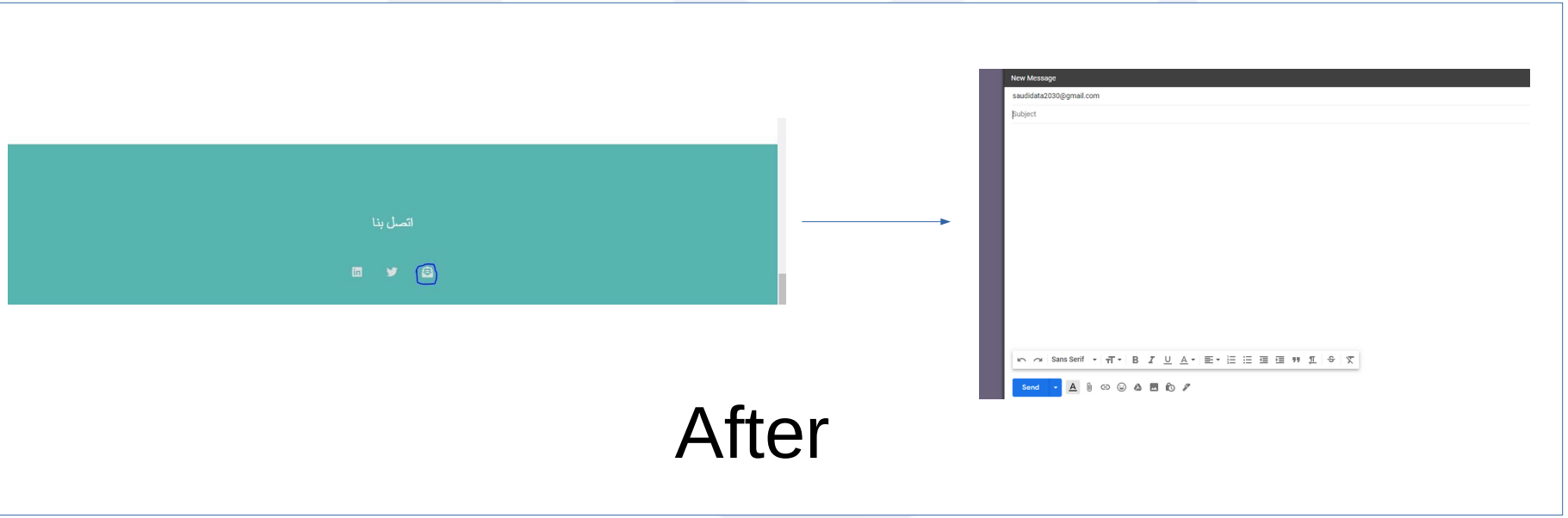


Figure 2

Implementation and coding ...

Website 08/11 update:

Fixing the email link problem.

```
<div id="contact-us" class="row">
  <div class="col-12">
    <a href="eifad@data.org.sa"><i class="fas fa-envelope-open-text icons"></i></a>
    <a href="https://twitter.com/Eyfad_SIG?s=21"><i class="fab fa-twitter icons"></i></a>
    <a href="https://sa.linkedin.com/in/saudidata2030"><i class="fab fa-linkedin icons"></i></a>
  </div>
</div>
</section>
```

Figure 3

```
</div>
<div id="contact-us" class="row">
  <div class="col-12">
    <a href="https://mail.google.com/mail/?view=cm&fs=1&tf=1&to=saudidata2030@gmail.com"><i class="fas fa-envelope-open-text icons"></i></a
    <a href="https://twitter.com/Eyfad_SIG?s=21"><i class="fab fa-twitter icons"></i></a>
    <a href="https://sa.linkedin.com/in/saudidata2030"><i class="fab fa-linkedin icons"></i></a>
  </div>
</div>
</div>
</section>

</body>
```

Figure 4

Implementation and coding ...

Website 08/11 update:

A page for explaining the meaning of the contributions.

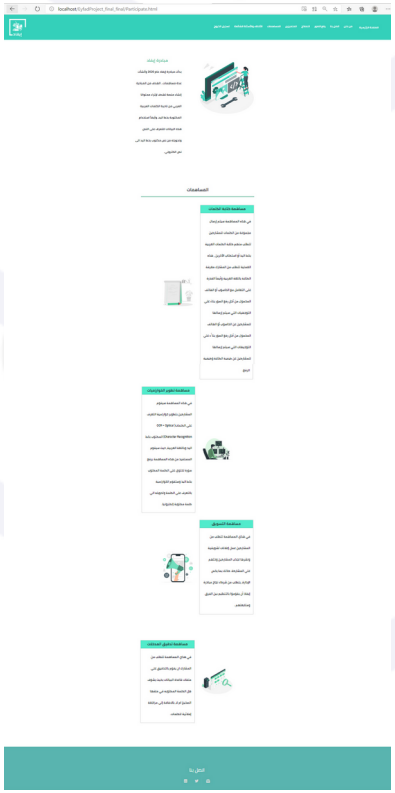


Figure 1

Implementation and coding ...

Website 08/11 update:

A page Frequent Asked Questions (FAQ).

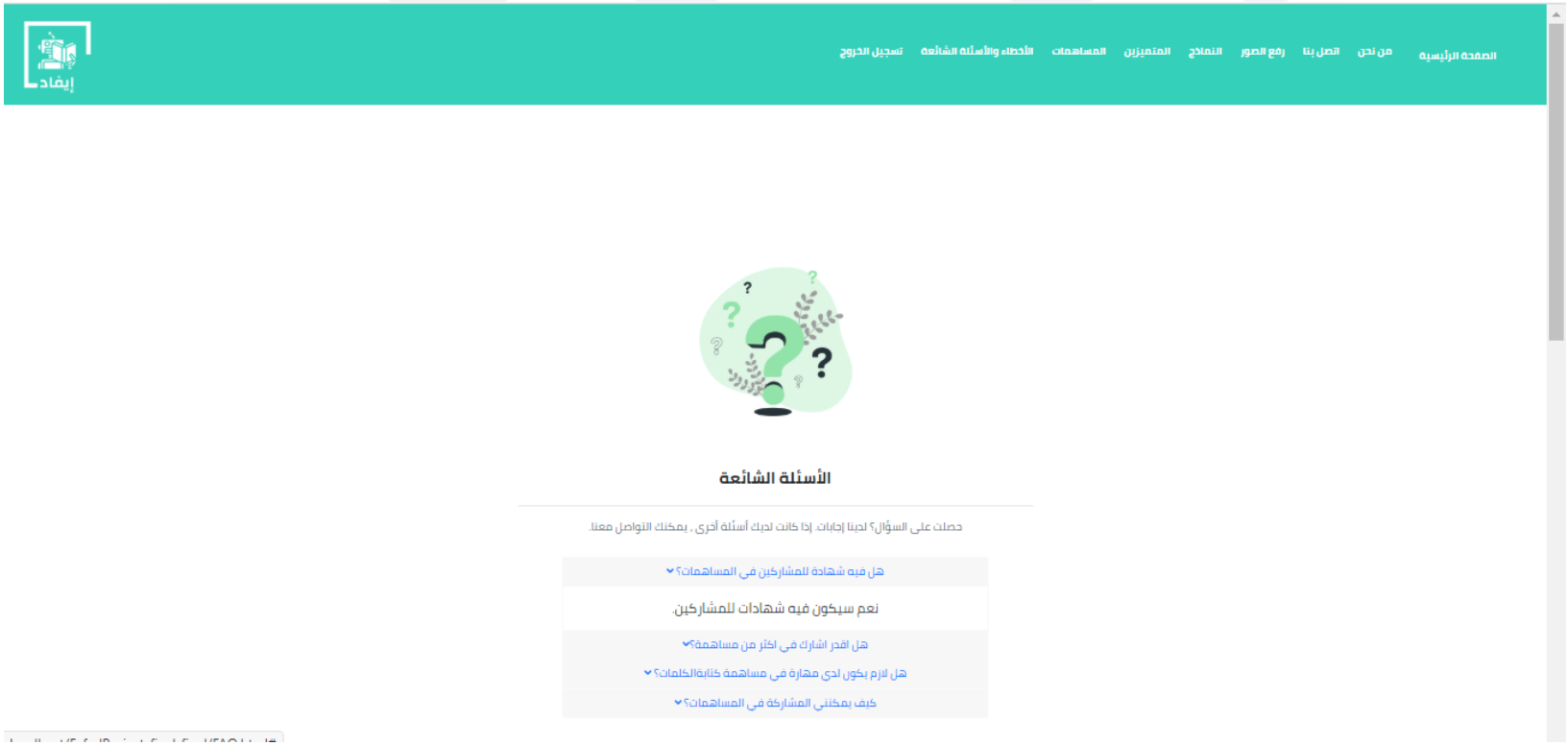


Figure 1

Implementation and coding ...

Website 08/11 update:

A page Frequent Asked Questions (FAQ).

Figure 2

Implementation and coding ...

Website 08/11 update:

A page Frequent Asked Questions (FAQ).

Figure 3

Implementation and coding ...

Website 08/11 update:

A page Frequent Asked Questions (FAQ).

Figure 4

Implementation and coding ...

Website 08/11 update:

A page Frequent Asked Questions (FAQ).

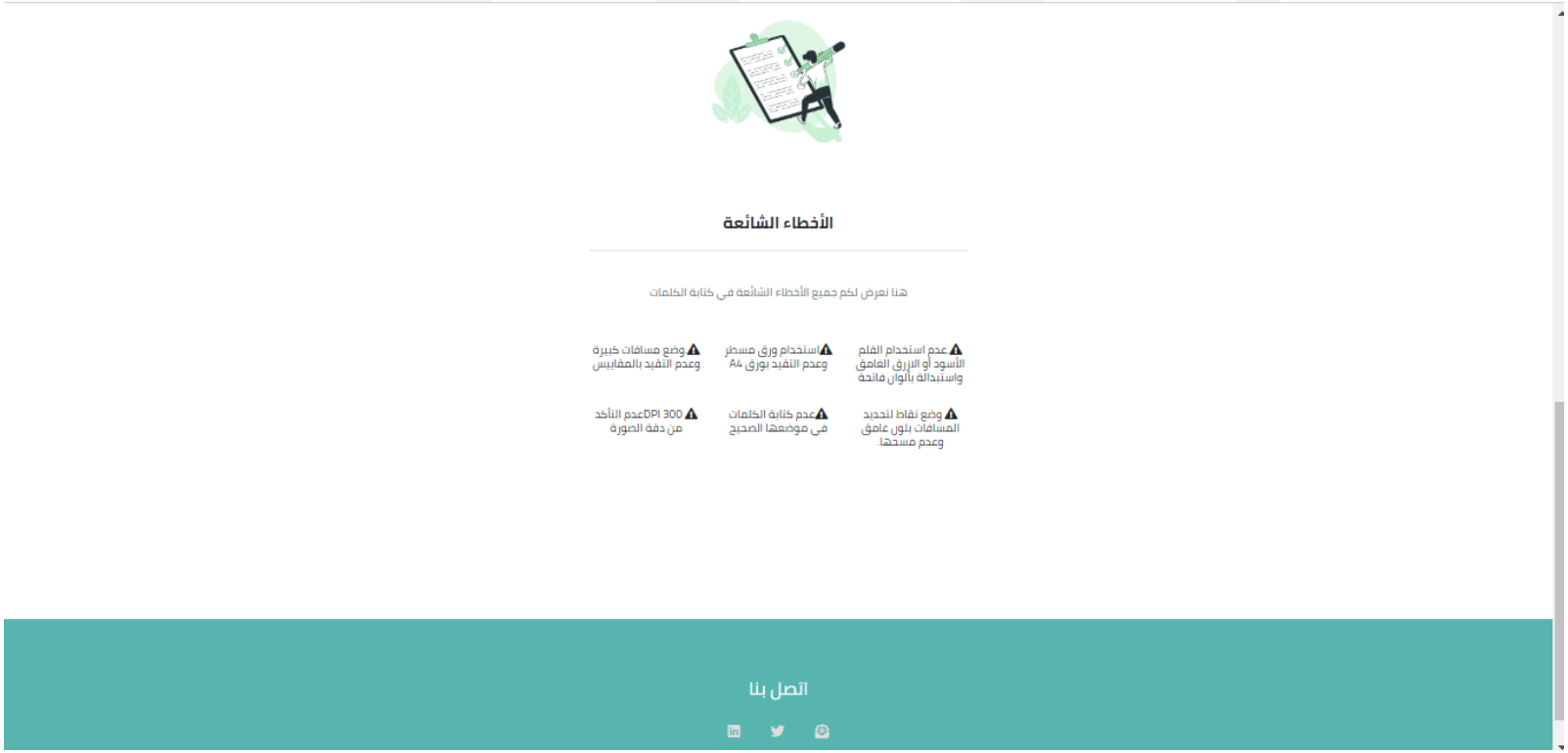


Figure 5

Deployment

Deployment ...

Resources

References