

TEXT BASED CAPTCHA END TERM REPORT

By

GROUP-2

(Section: K19TS)

<u>Name</u>	<u>Registration No.</u>	<u>Roll No.</u>
Priyanshi Soni	11902073	RK19TSB40
Ajay Jangir	11906202	RK19TSB43



School of Computer Science Engineering
Lovely Professional University,
Jalandhar November-2020



STUDENT DECLARATION

This is to declare that this report has been written by us. No part of the report is copied from other sources. All information included from other sources have been duly acknowledged. If any part of the report is found to be copied, we shall take full responsibility for it.

Priyanshi Soni (Roll no- 40)

Ajay Jangir (Roll no- 43)



TABLE OF CONTENTS

TITLE	PAGE NO
1. BONAFIDE CERTIFICATE	04
2. BACKGROUND AND OBJECTIVES OF PROJECT ASSIGNED	05
3. IMPLEMENTATION OF PROJECT	07
4. TECHNOLOGY AND FRAMEWORK USED	16
5. SWOT ANALYSIS	16



BONAFIDE CERTIFICATE

This is to Certify that project report “Text Based Captcha” is the bonafide work of “ **Priyanshi Soni & Ajay Jangir** ” who carried out the project work under my supervision.

Neha Bagga

Asst. Professor ID: 23407

School of Computer Science and Engineering

Lovely Professional University, Jalandhar.



Background and Objectives of Project Assigned

Background:

Captcha is stands for Completely Automated Public Turing test to tell Computer and Human Apart. As the increase of automated bots systems or software that misuse and corrupt the public web services, the user must required going through and solving a Turing test problem, before they use web services. This Turing test is called Captcha. Many CAPTCHAs have been proposed in the literature that text-graphical based, audio-based, puzzle-based and mathematical questions-based. The design and implementation of CAPTCHAs fall in the realm of Artificial Intelligence.

Motivation:

People have developed techniques, systems, programs and software systems that can replace a normal human being to do a job; such kinds of jobs include entering of data into systems, generate data automatically, handling events that occur on or within a system. As a matter of fact, web sites must ensure that the services are supplied to legitimate human users rather than bots to prevent service abuse. To thwart automated attacks, services often ask users to solve a puzzle before being given access to a service.

Goals and objectives:



CAPTCHAs are designed to be simple problems that can be quickly solved by humans, but are difficult for computers to solve. Using Captchas, services can distinguish legitimate users from computer bots while requiring minimal effort by the human user . In the procedure, a computer or a program creates a test for its user, who is expected to be a human. The test is meant for the humans, that is, it is to be solvable only by humans and not any other machine, system or program. The user is required to provide a correct response to the test and then the user is permitted to access the work. When a correct response is received, it is presumed that the response arrived because of a human user. CAPTCHA techniques have been classified into four categories: - Text based Captcha. - Audio based Captcha. - Image based Captcha. - Video based Captcha. Each type is suitable to serve different group of users. CAPTCHA's code is a series of characters (uppercase and lowercase) and numbers.

- ♣ Multiple randomizing functions are used to generate a random code (stream of characters and numbers) in each challenge in order to make it not susceptible to a dictionary attack.

- ♣ The length of the code is varied (minimum length is 6 characters-numbers).

- ♣ Multiple font types are handled to prevent intrusion using image processing techniques when a consistent font is used.

- ♣ String/code are rotated at various angles.



♣ Lines are utilized to prevent segmentation. The numbers and the length of lines and their positions are varied each time in order to distort the text image randomly before being presented to the user.

Outcome of the Project:

As a contribution toward improving the web security in the field of an automated challenge and response against attacks issued by automated programs, we proposed a simple text-based CAPTCHA. Two main goals have been considered to be achieved that is: simplicity of solving the technique for a human as well as the time that a human actually needs to find the solution.

Implementation of Project

The project is to generate a random text based captcha using GUI. The program is generating random captcha every time when you **start** it. Now you have to **verify** it by entering the captcha. If you write wrong, an alert will pop up in the tkinter screen and saying '**Entered Captcha is wrong**' and then a **new captcha** will automatically generated and now you have to write new captcha. Now If your entered captcha is right , an alert will pop up and say '**Entered captcha is right**' then also a new captcha is generated, now you have to write new captcha & the process is **carry on**. This is how it works.



Now, **Let's Understand the code !**

```
from tkinter import * #importing tkinter to use GUI
import random #importing random to generate random numbers
import string #importing string
import tkinter.messagebox as tmsg
#importing tkinter messagebox as tmsg to show alert
```

```
root=Tk() #initialising tkinter with its entry widget root to create a
root window
```

```
root.geometry("700x600") #Setting dimensions of root window
```

```
root.title("Text Based Captcha")
#Putting title 'Text Based captcha' which is our Project name
```

Now we have to generate random numbers & alphabets to make a captcha.

What we will do, we will generate random no's and random alphabets's apart and after we will concat them both to make our desired captcha.

These random no's and random alphabets will also having random size & for this we have created below two variables z & p.

```
z=random.randint(3,5) #generating random numbers from 3 to 5
p=random.randint(3,5) #generating random numbers from 3 to 5
```

```
concat=' ' #creation of empty string, which we will use later
```




```
list1=[]      #creation of empty list1 to store the generated random
numbers
list2=[]      #creation of empty list2 to store the generated random
alphabets
```

```
for i in range(0,z):
    list1.append(str(random.randint(0,9)))
#Now it will take the random size generated by z and will generate the
random numbers from 0 to 9 and after typecasting it to string we will
storing it in list1
```

```
for i in range(0,p):
    list2.append(random.choice(string.ascii_letters
)) #Now it will take the random size generated by p and will generate
the random alphabets from (a to z , A to Z) and after generating we will
storing it in list2
```

```
for i in range(0,random.randint(2,4)):
    concat=concat+list1[random.randint(0,len(list1)
-1)]+list2[random.randint(0,len(list2)-1)]
```

#Now we are again generating random no in for loop to get everytime random size for our captcha. Inside for loop we are taking random no and alphabet from our stored lists and and storing it in our empty string concat, which we have initialised earlier.

Now we are having our generated captcha in variable 'concat'. We are good to go further .

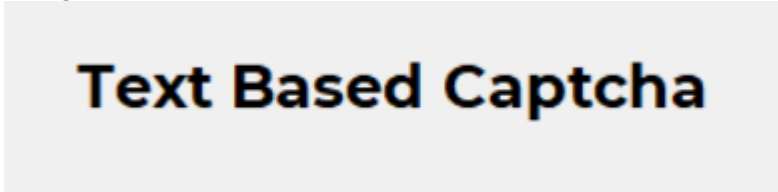
```
Label(root,text="Text Based Captcha",font="Montserrat
at 18 bold").pack(pady=70,side=TOP)
```



#Created a label in root window, which will show our project title heading. We doesn't create variable for the label and pack it in one line, because we are not going to do anything with this label further.

We have used font family called 'Montserrat' and set a size 16 and weight bold for the text. Then after I pack my label and align this label to the TOP.

Output:



```
photo=PhotoImage(file="captcha4.png",height=200,width=450)
```

#Created a widget PhotoImage in photo variable and set a default picture namely 'captcha4.png', having height 200 and width 450.

```
labelPhoto=Label(image=photo,text=f"{concat}",font="Aswell 28 overstrike",fg="#575757",compound="center")
```

```
labelPhoto.pack()
```

#Created a label to show the image, and we have to show the generated captcha

On the image, So we have used a property called compound='center' which shows my text on the image and will center it. Now the text shown is in variable called 'concat' but we have to write generated captcha in double quotes, so we use something called fstring.

We have used font-family called Aswell to give it a captcha style and overstrike to strike my text.

Output:





Now we have successfully generated our captcha and shown it to user as well. Now we require a entry widget so that user can enter the captcha and then we can verify it whether it is correct or not.

`uservalue=StringVar()` #This is the creation of string variable which we are creating for our entry widget which is specifying that entry widget can will accept the values having datatype string.

```
entry= Entry(root,textvariable=uservalue,relief=SUNKEN,
borderwidth=4,font="Montserrat 12",justify=CENTER)
```

```
entry.pack(ipady=10,pady=60)
```

#Entry widget created inside root window, having datatype string(textvariable=uservalue). We have provided borderwidth and style SUNKEN to it. That justify=CENTER will write our text from center of entry box, and ipady we have used to give it a height we want.



Output:



The below described function is call when button is clicked. We have created button below to it and defining it here.

```
def fun1():  
    concat2= '' #Creation of empty string  
    list3=[] #Creation of empty list  
    list4=[] #Creation of empty list
```

#This labelPhoto['text'] will give us the the text of that variable, and we have store our generated captcha there so we will get the value of our captcha by this...Now we are checking whether generated captcha and the value entered by user in entry box is same or not...if same then captcha will be verified.

```
if(labelPhoto['text']==uservalue.get()):  
    list5=['captcha.png', 'captcha3.png', 'captch  
a2.png', 'captcha4.png'] #Here we are storing name of random  
images which we are using in it.
```

```
    tmsg.showinfo("Captcha Verification", "Enter  
ed Captcha is Right") #Alert will pop up here
```



Output:



```
for i in range(0,z):  
    list3.append(str(random.randint(0,9)))
```

#We are again generating random no's like we did above and storing it in a list

```
for i in range(0,p):  
    list4.append(random.choice(string.ascii  
_letters)) #We are again generating random alphabet's like we did  
above and storing it in a list
```

```
for i in range(0,random.randint(2,4)):
```

```
    concat2=concat2+list3[random.randint(0,  
len(list1)-1)]+list4[random.randint(0,len(list2)-  
1)]
```

#Now We are taking randomly data from both the lists and concating it to our empty string which will gives a new captcha which stored in 'concat2'. This captcha is also having random size as we defined range of for loop randomly.

```
photo['file']=f"{list5[random.randint(0,2)]  
}"
```



#As we have generated our new captcha now finally we are changing the image randomly. It is taking the image names from list5 which we have created in the starting of if block. We are changing photo with the property Photo['file'] . We are assigning random image from the stored images in list5. We have used fstring because we are using variable and we have to write string value.

```
labelPhoto['text']=f"{concat2}"#Assigning the  
generated captcha to text of label variable 'labelPhoto'  
uservalue.set('') #We are setting now entry widget  
value to the empty so that we can write new captcha.
```

We are doing the same process which we have done in if part. As we are generating random captcha in both the cases whether user have written captcha right or not.

```
else:  
    tmsg.showinfo("Captcha Verification","Enter  
ed Captcha is wrong") #Alert will pop up here
```

Output:



```
list5=['captcha.png','captcha3.png','captch  
a2.png','captcha4.png']
```

```

        for i in range(0,z):
            list3.append(str(random.randint(0,9)))

        for i in range(0,p):
            list4.append(random.choice(string.ascii
_letters))

        for i in range(0,random.randint(2,4)):

            concat2=concat2+list3[random.randint(0,
len(list1)-1)]+list4[random.randint(0,len(list2)-
1)]

        photo['file']=f"{list5[random.randint(0,2)]
}"
        labelPhoto['text']=f"{concat2}"
        uservalue.set('')

```

#Now we finally creating our submit button which is calling the above function.

```

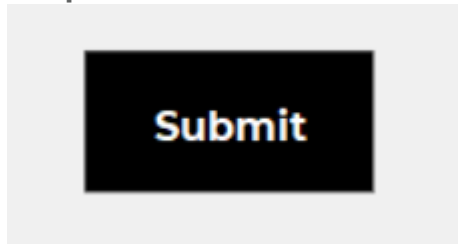
Button(root,text="Submit",fg="#fff",bg="#000", font
="Montserrat 12 bold",command=fun1,padx=20,pady=10)
.pack()

```

#Button created inside root window having background colour and text color value in hex code. That command=fun1 telling us to which function we have to call.



Output:



`root.mainloop()` #With the help of mainloop, we are running our application.

Technology and Framework Used

We used the Jupyter and Visual Studio code for coding purpose.

We used the concepts of- Python data types, functions, loops(for loop mainly) and Python GUI using Tkinter for designing our text-based captcha.

We downloaded the background images for captcha from Internet and downloaded the font styles for the successful completion of the project.

SWOT Analysis

Strength-

Text based CAPTCHA are the simplest type.

Text CAPTCHA are designed in every language.



The loading speed of these schemes is very fast.

These are very easy to use and easy to understand for humans.

Weakness –

To break these CAPTCHA is very easy most of the time.

A number of times simple OCR techniques are enough to crack them.

Generally these are designed in English language.

To make these challenges hard distortion is added that also creates problems for humans also.

Some alphabets and digits have very different shapes, but when they are distorted it becomes very difficult to recognise them.

It is found in many cases 8 may look like 6 or 9 and 7 may look like 1 or vice versa. It can happen due to some specific type of fonts or too much distortion.

Opportunities: Captchas are a must have for every website whether it is a online payment platform, e-shopping, gmails. It is a gatekeeper controlling your entry and judging whether you are a human or a spamming machine.

Threats: Poor user experience. **captchas** can be sometimes be time consuming and challenging, especially those with more complicated challenges such as text or image identification, which have earned notoriety for annoying



users. Users may potentially switch to websites without **captchas** due to frustration.

The Github Links for our Project are as follows:

https://github.com/priyanshi-2001/python_captcha (By Priyanshi Soni)

https://github.com/ajayjangir0044/Python-Project-text-_based_captcha

(By Ajay Jangir)



