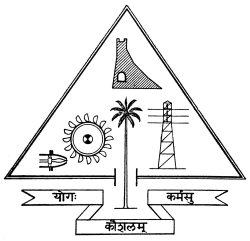
ADVANCED CAPTCHA



CS09-608(p) B.Tech Mini Project 2012

Done By

Shahana Hamza Mamutty(ETAJECS 049)

Sruthy K(ETAJECS 057)

Varun M(ETAJECS 063)

Guided by

M Jayasree

Assistant Professor

AND

Rani Koshy

Assistant Professor

Dept of Computer Science And Engineering
Government Engineering College
Thrissur-680009

ABSTRACT

It is a secure CAPTCHA that uses the property that the humans will be able to trace a tangled line more efficiently than a computer. It is added as an extra layer to the regular CAPTCHA to protect high security systems.

The below image is a rough sketch of the proposed CAPTCHA

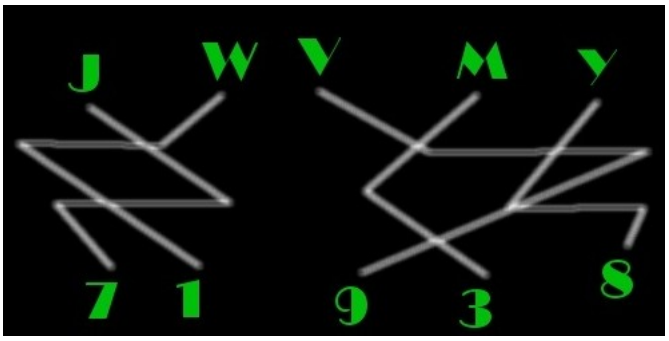


Figure 1: rough sketch 1

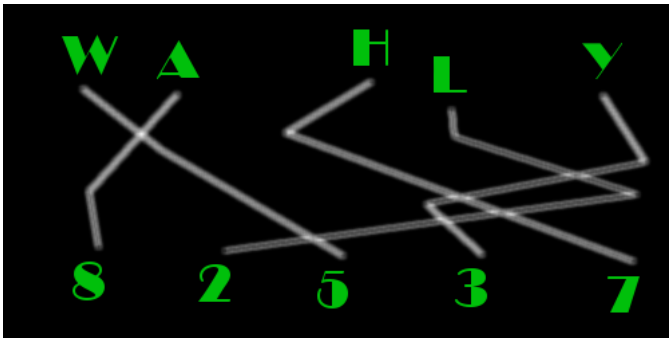


Figure 2: rough sketch 2

Letters above are mapped to numbers below by the black lines. The user traces this line and maps this. The lighter lines are for confusing any line tracer software. The lighter lines will be black in the final CAPTCHA. The lighter lines are for easy understanding of the reader of this material.

Randomness is another major feature of this idea. Sometimes more than one letter might be mapped to a single number. Perhaps there may be

numbers that does not have pre -images. Even there might be letters that does not have mappings. This is very effective to confuse the bot who tries to decode it.

ACKNOWLEDGEMENT

First of all we would like to express our sincere thanks to the almighty god for showering his grace upon us for the completion of this project.

We are very thankful to everyone who all supported us.We are equally grateful to M Jayasree(Assistant professor) and Rani Koshy(Assistant professor) for their moral support and guidance in different matters regarding the topic.He had been very kind and patient while suggesting the outlines of this project and clearing our doubts.We thank him for his overall supports.

Last but not the least we are also thankful to our friends who supported us and helped us to complete this project successfully.

Contents

List of Tables

List of Figures

Nomenclature

GUI	Graphical User Interface
CAPTCHA	Completely Automated Public Turing Test to tell Computers And Humans Apart
OS	Operating System

Chapter 1

Introduction

C APTCHA is the most popular technique to tell com- puters and humans apart. The Primary function of CAPTCHA is to present a test to the end user to deter- mine if the end user is a human or an automated program. This is generally the first line of defence in spa protection. It can also be used to prevent hogging of resources by an automated scripts without which the server will be over- loaded. CAPTCHAs have always been a target for hackers , and most of the earlier CAPTCHAs are now decodable by bots using optical character recognition. (E.g. EZ-Gimpy that was used against Yahoo! in creating mail accounts). Cracking of the existing CAPTCHA involves the following processes:

1. Processing of background
2. Segmentation
3. Character recognition

To counter this, modern CAPTCHA the letters are more closely packed to confuse segmentation, but even that could be cracked with powerful bots. This calls for a more effective method based on skills only an human could exhibit for security purposes.

Chapter 2

Requirement Analysis

2.1 Software Requirements

The following software platforms are used for this project

- Compiler for Python v2.7
- OpenCV library
- Qt library for creating GUI for demonstration purposes.
- Python django library for web deployment

2.2 Hardware Requirements

The minimum hardware requirements are:

- Network server with graphics processing capabilities.

2.3 Over all description

2.3.1 Product Perspective

This product targets the modification of an existing technology. The CAPTCHA helps to

distinguish between humans and bots by obfuscated character identification. The proposed product uses relational mapping among characters to achieve the same goal. The

functionality is achieved by using one of the unique properties of human brain The Top- Down approach.

2.3.2 Defenitions

CAPTCHA stands for Completely Automated Turing Test to Tell Computers and Humans

Apart. It is used to distinguish between human beings and automated bots.

Automated bots are softwares that are designed to run automatically. They can be used

for malicious purposes as well, so it is necessary to identify, distinguish and control their

operation.

2.3.3 Operating environment

The operational environment should satisfy the minimum hardware and software

requirements.

2.3.4 User documentation

Not provided, as the user is not involved in the functioning of the product. But

instructions are supplied along with the user interface, so that the user understands what

has to be done.

2.3.5 Purpose

The proposed product targets at the modification of the existing CAPTCHA system,

so as to make it more secure. Just like the existing CAPTCHA, the proposed product also

generates a unique image each time it is invoked. The image consists of domain entries,

range entries and oblique relational lines mapping the domain with the range. The end-user

is supplied with this image between domain and range. The user analyses the image and

provides the answer. The answer is send back to the server and validated. The user is

granted access if the answer is right, and denied otherwise.

2.3.6 User interface

The UI consists of an image, a question and an input field, displayed in a web-page. The

user analyses the image according to the question and inputs the required answer in the

provided input field.

2.3.7 Software interface

The software interface consists of the image and question generator, and an answer validator. Whenever a request is obtained, the generator generates an image and a question and sends it to the client. The client replies with an answer. That is compared with the actual answer already generated along with the question by the validator. If a match is encountered, then the validator acknowledges with a positive feedback.

2.3.8 Communication interface

The communication takes place via the Internet.

2.4 Detailed functionalities

2.4.1 Functional requirements

2.4.1.1 Relational mapping

Characters are selected at random and their relationship is established using the relational mapper module. It is here where the domain and range entries are selected and the relationship among them is established.

2.4.1.2 Image generation

The domain and range entries and their relationship is mapped on to an image.

The relational lines are made distorted so as to confuse bots from parsing it. The distorted lines are human-readable.

2.4.1.3 Question and answer generation

Some random human understandable questions are generated according to the relations and their corresponding answer is saved.

2.4.1.4 Communication module

The generated image and the question are sent to the requested client using the communication module.

2.4.1.5 Validation

The answer supplied by the user is validated against the previously stored answer. If the answer matches, a positive response is sent, else a negative answer is
sent.

2.4.2 Non-functional Requirements

2.4.2.1 Perfarmance Requirements

SPEED: sped of image generation, processing and validation must be fast so that
there must not be any time delay.
EASE OF USE: the system should be user friendly so that the users having less
technical knowledge can also easily understand and also the image must not be that
much complex so that the user cannot understand the mappings.

2.4.2.2 Security Requirements

The program source must not be accessible over the web, so the implementation
has to be done so. Also second chance for answering the same image must not be
permitted, as it could result in implementation of brute force cracking techniques by the
bot.

2.4.2.3 Software Quality Requirements

2.4.2.4 Scalability

The software must be scalable in order to incorporate future changes.

2.4.2.5 Maintainability

The software must be maintainable. Stability of the server must be ensured in order to honor the client requests.

2.4.2.6 Adaptability

The software must be adaptable with the evolution of technology.

2.4.2.7 Reliability

The server must be reliable in order to honor client requests.

2.4.2.8 Testability

The software should be properly tested under various circumstances in order to assure its reliability.

2.5 Scope of the project

The product has got the same scope as that of existing CAPTCHA. The malicious bots,

which can render many web-based services like e-mail etc. obsolete, need to be blocked.

CAPTCHA is a very effective technique for doing so. They help prevention of server

overloads.

2.6 Conclusion

With the advancement of technology, the existing CAPTCHA mechanisms face the threat

of being crackable. So the CAPCTHA has to be made more advanced so as to stand up to the

technology.

Chapter 3

Design And Implementation

3.1 System Design

3.1.1 System functions

Functions of the system mainly includes:

1. Listen to user requests
2. Receive user requests
3. Generate image and related data
4. Allow multiple requests by storing data to database
5. Transmission and reception of data (image, question, answer, acknowledgement)
6. Validation of answer input by the user

3.1.2 Data flow diagrams

Data flow diagrams for level0, level1 and level2 can be considered sa below:

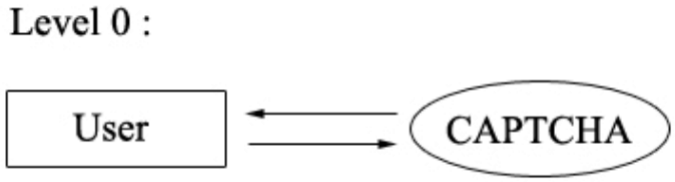


Figure 3.1: level 0 dataflow

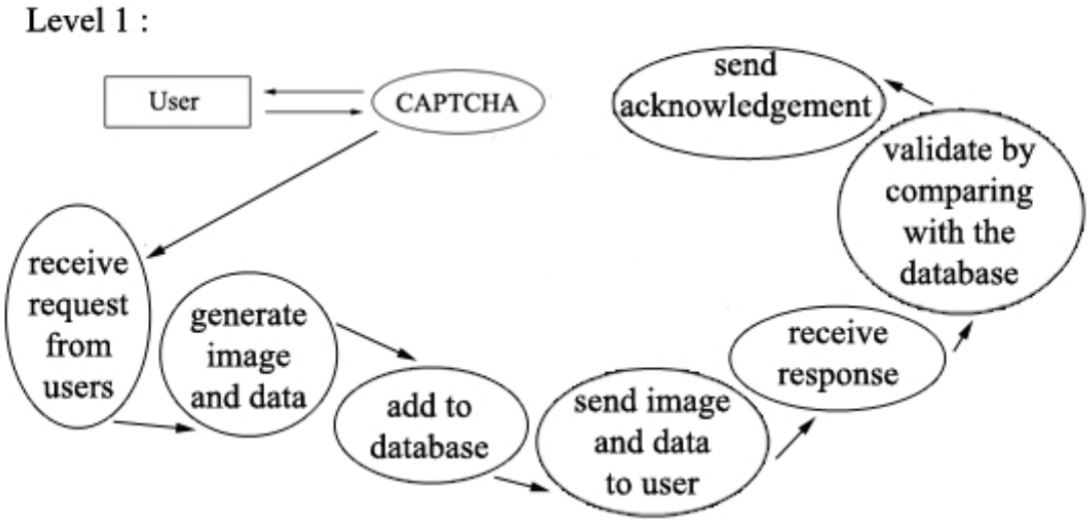


Figure 3.2: level 1 dataflow

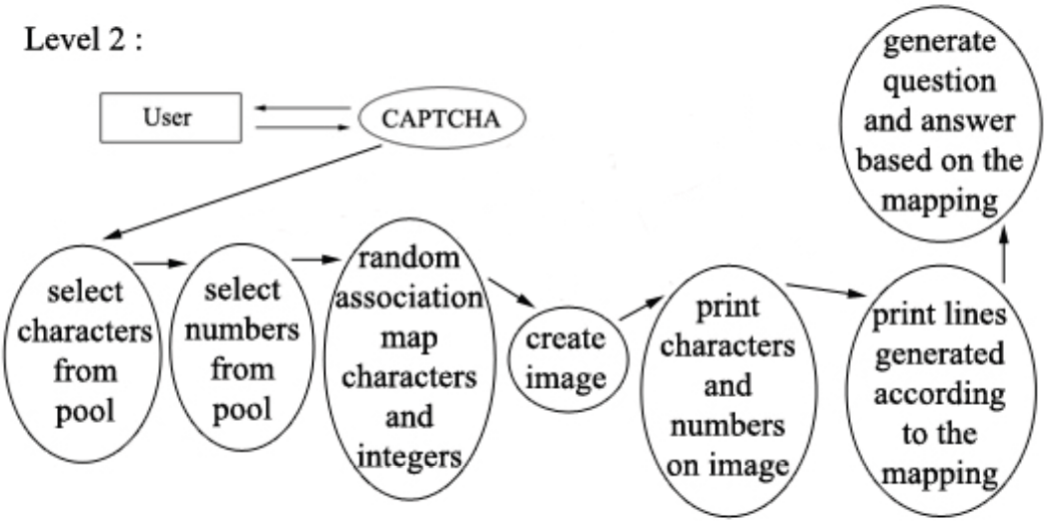


Figure 3.3: level 2 dataflow

3.2 Database Design

3.2.1 E-R diagram

E-R diagram can be simply considered as below:

Figure 3.4: Entity Relationship Diagram

3.3 GUI/User Interface Design

The UI is generated on a webpage. It consists of
1. The generated image from the server
2. The corresponding question
3. An input field to enter the answer
4. The submit button

Chapter 4

Coding

4.1 Algorithm for Captcha

4.1.1 Section-Generate Data

```
Begin

Alphabets = A,B,C.......,Z
Numbers = 1,2,3.....,9,0
Select five characters from alphabet.
Select five digits from numbers.
Randomly map selected characters and numbers and store them to a dictionary cal

End
```

4.1.2 Section-Generate Image

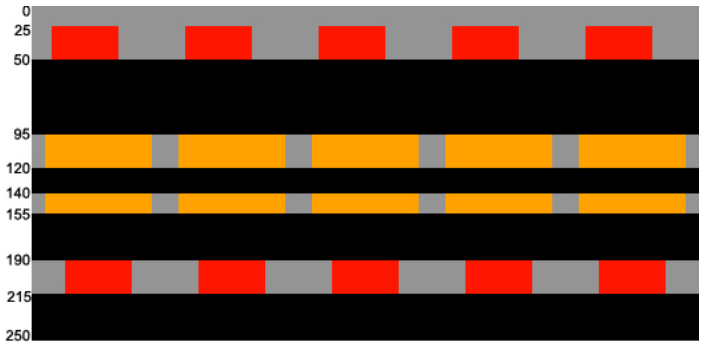


Figure 4.1: segmented form of image

```
Begin

Using Python Imaging Library create a new image of dimension 500 x 250.
```

```
Paint characters in first row of the image (marked in red).
Paint numbers in last row of the image (marked in red).
Randomly select the numbers of bends in the line (either one or two).
Select points from second or third row (marked in orange) such that the lines do
Image is Generated.

End
```

4.1.3 Section-Generate Question

```
Begin

Randomise the order of characters.
Generate the questions.
Generate the answers according to the new order.

End
```

4.1.4 Validation

```
Begin

Accept the input.
Compare the entered answers with the correct answer.
According to tha value returned provide or deny the access.

End
```

4.2 Code for Captcha

```
Begin

#!/usr/bin/python2.7

import random, sys
from PyQt4 import QtGui, QtCore
from PIL import Image, ImageFont, ImageDraw, ImageFilter

VERSION = '0.1'

def co_ordinate(x1,y1,x2,y2):
    x = random.randrange(x2-x1)+x1
    y = random.randrange(y2-y1)+y1
    return x,y
```

```
def XY(p):
    if p == 1:
        return co_ordinate(10,95,90,120)
    elif p == 2:
        return co_ordinate(110,95,190,120)
    elif p == 3:
        return co_ordinate(210,95,290,120)
    elif p == 4:
        return co_ordinate(310,95,390,120)
    elif p == 5:
        return co_ordinate(410,95,490,120)
    elif p == 6:
        return co_ordinate(10,140,90,155)
    elif p == 7:
        return co_ordinate(110,140,190,155)
    elif p == 8:
        return co_ordinate(210,140,290,155)
    elif p == 9:
        return co_ordinate(310,140,390,155)
    elif p == 10:
        return co_ordinate(410,140,490,155)


class captcha():

    def __init__(self):

        self.mapping = {}
        self.character_coordinates = {}
        self.integer_coordinates = {}
        self.characters = []
        self.font = ImageFont.truetype('font',40)

    def generate_data(self):

        #Generating Characters
        alphabets = ['A','B','C','D','E','F','G','H','J','K','L','M',
                     'N','P','Q','R','S','T','U','V','W','X','Y','Z']
        self.characters = random.sample(alphabets,5)
        print 'Domain : ', self.characters
        for i in range(5):
            m = random.randrange(50)+25
            n = random.randrange(25)+5
            self.character_coordinates[self.characters[i]] = (m+(i*100)+10)*100

        #Generating Numbers
        numbers = ['1','2','3','4','5','6','7','8','9']
```

```
        self.print_order = random.sample(numbers,5)
        print 'Range : ', self.print_order

        #Mapping Characters to Numbers
        self.map_order = random.sample(self.print_order,5)
        for i in range(5):
            self.mapping[self.characters[i]]=self.map_order[i]
        print 'Relation : ', self.mapping

    def generate_image(self):

        image = Image.new('RGB',(500,250),'#000000')
        draw = ImageDraw.Draw(image)

        box_flag = [0,0,0,0,0,0,0,0,0,0,0]
        numbers = ['-','-','-','-','-']
        numbers_flag = 0
        pq =[]

        for i in range(5):
            if i == 0:
                j = self.characters[i]
            elif i == 1:
                j = self.characters[4]
            else :
                j = self.characters[i-1]
            x = self.character_coordinates[j]
            m = x//100
            m = m-5
            n = x % 100

            print 'Line ',i,' :'

            intermediate_points = random.randrange(2) + 1

            print 'Bends :',intermediate_points
            p1, p2 = 0,0
            p1x , p1y = 0,0
            p2x , p2y = 0,0
            possible_box =[]

            if intermediate_points == 0:
                pq.append(m*1000 +n)
                if i == 0:
                    p2 = 1
                elif i == 1:
                    p2 =5
```

```
            else:
                p2 = i
        elif intermediate_points == 1:
            if i == 0:
                if box_flag[1] == 0:
                    possible_box.append(1)
                if box_flag[2] == 0:
                    possible_box.append(2)
                if box_flag[7] == 0:
                    possible_box.append(7)
                if box_flag[6] == 0:
                    possible_box.append(6)
            elif i == 1:
                if box_flag[4] == 0:
                    possible_box.append(4)
                if box_flag[5] == 0:
                    possible_box.append(5)
                if box_flag[9] == 0:
                    possible_box.append(9)
                if box_flag[10] == 0:
                    possible_box.append(10)
            else :
                if box_flag[i-1] == 0:
                    possible_box.append(i-1)
                if box_flag[i] == 0:
                    possible_box.append(i)
                if box_flag[i+1] == 0:
                    possible_box.append(i+1)
                if box_flag[i+4] == 0:
                    possible_box.append(i+4)
                if box_flag[i+5] == 0:
                    possible_box.append(i+5)
                if box_flag[i+6] == 0:
                    possible_box.append(i+6)

            p2 = random.sample(possible_box,1)
            p2 = p2[0]
            box_flag[p2] = 1
            p1x,p1y = XY(p2)
            print 'Bend : ', possible_box, p2
            draw.line([(m,n),(p1x,p1y)],'#FFFFFF', 3)
            pq.append(p1x*1000+p1y)

        elif intermediate_points == 2 :
            if i == 0:
                if box_flag[1] == 0:
                    possible_box.append(1)
```

```
        if box_flag[2] == 0:
            possible_box.append(2)
        if box_flag[7] == 0:
            possible_box.append(7)
        if box_flag[6] == 0:
            possible_box.append(6)
    elif i == 1:
        if box_flag[4] == 0:
            possible_box.append(4)
        if box_flag[5] == 0:
            possible_box.append(5)
        if box_flag[9] == 0:
            possible_box.append(9)
        if box_flag[10] == 0:
            possible_box.append(10)
    elif i == 2 :
        if box_flag[i-1] == 0:
            possible_box.append(i-1)
        if box_flag[i] == 0:
            possible_box.append(i)
        if box_flag[i+1] == 0:
            possible_box.append(i+1)
        if box_flag[i+4] == 0:
            possible_box.append(i+4)
        if box_flag[i+5] == 0:
            possible_box.append(i+5)
        if box_flag[i+6] == 0:
            possible_box.append(i+6)
        if box_flag[1] == box_flag[6] == 0 and possible_box.count(1)
            possible_box.remove(6)
        if box_flag[8] == 0:
            possible_box.remove(8)
        if box_flag[3] == 0:
            possible_box.remove(3)
        if box_flag[1] == box_flag[6] == 1:
            if box_flag[8] == 0:
                possible_box.append(8)
            if box_flag[3] == 0:
                possible_box.append(3)
            if len(possible_box) == 2 and box_flag[3] == box_flag[8]
                possible_box.remove(8)
    elif i == 3:
        for b in range(1,11):
            if box_flag[b] == 0:
                possible_box.append(b)
        if box_flag[1] == box_flag[6] == 0 and possible_box.count(1)
            possible_box.remove(6)
```

```
            if box_flag[5] == box_flag[10] == 0 and possible_box.count(5
                possible_box.remove(10)
            if box_flag[8] == 0:
                possible_box.remove(8)
            if box_flag[3] == 0:
                possible_box.remove(3)
            if box_flag[1] == box_flag[6] == box_flag[5] == box_flag[10]
                if box_flag[8] == 0:
                    possible_box.append(8)
                if box_flag[3] == 0:
                    possible_box.append(3)
        elif i == 4 :
            if box_flag[i-1] == 0:
                possible_box.append(i-1)
            if box_flag[i] == 0:
                possible_box.append(i)
            if box_flag[i+1] == 0:
                possible_box.append(i+1)
            if box_flag[i+4] == 0:
                possible_box.append(i+4)
            if box_flag[i+5] == 0:
                possible_box.append(i+5)
            if box_flag[i+6] == 0:
                possible_box.append(i+6)
            if box_flag[5] == box_flag[10] == 0 and possible_box.count(5
                possible_box.remove(10)
            if box_flag[8] == 0:
                possible_box.remove(8)
            if box_flag[3] == 0:
                possible_box.remove(3)
            if box_flag[5] == box_flag[10] == 1:
                if box_flag[8] == 0:
                    possible_box.append(8)
                if box_flag[3] == 0:
                    possible_box.append(3)
                if len(possible_box) == 2 and box_flag[3] == box_flag[8]
                    possible_box.remove(8)
        p1 = random.sample(possible_box,1)
        p1 = p1[0]
        box_flag[p1] = 1
        p1x,p1y = XY(p1)
        print 'First bend : ', possible_box, p1
        draw.line([(m,n),(p1x,p1y)],'#FFFFFF', 3)

        possible_box = []
        if p1 == 1:
            if box_flag[6] == 0:
```

```
            possible_box.append(6)
        if box_flag[7] == 0:
            possible_box.append(7)
        if box_flag[2] == 0:
            possible_box.append(2)
        if box_flag[2] == box_flag[6] == box_flag[7] == 1:
                if box_flag[3] == 0:
                    possible_box.append(3)
                if box_flag[8] == 0:
                    possible_box.append(8)
    elif p1 == 6:
        if box_flag[7] == 0:
            possible_box.append(7)
        if box_flag[2] == 0:
            possible_box.append(2)
        if box_flag[7] == box_flag[2] == 1:
                if box_flag[3] == 0:
                    possible_box.append(3)
                if box_flag[8] == 0:
                    possible_box.append(8)
    elif p1 == 2:
        if box_flag[1] == box_flag[7] == 1:
            possible_box.append(6)
        elif box_flag[6] == box_flag[7] == 1:
            possible_box.append(1)
        else:
            if box_flag[7] == 0:
                possible_box.append(7)
            if box_flag[1] == 0:
                possible_box.append(1)
            if box_flag[6] == 0:
                possible_box.append(6)
    elif p1 == 7:
        if box_flag[1] == box_flag[2] == 1:
            possible_box.append(6)
        elif box_flag[6] == box_flag[2] == 1:
            possible_box.append(1)
        else:
            if box_flag[1] == 0:
                possible_box.append(1)
            if box_flag[6] == 0:
                possible_box.append(6)
            if box_flag[1] == box_flag[6] == 1:
                if box_flag[3] == 0:
                    possible_box.append(3)
                if box_flag[8] == 0:
                    possible_box.append(8)
```

```
elif p1 == 3:
    if box_flag[8] == 0:
        possible_box.append(8)
    if box_flag[7] == 0:
        possible_box.append(7)
    if box_flag[2] == 0:
        possible_box.append(2)
    if box_flag[4] == 0:
        possible_box.append(4)
    if box_flag[9] == 0:
        possible_box.append(9)
elif p1 == 8:
    if box_flag[4] == 0:
        possible_box.append(4)
    if box_flag[7] == 0:
        possible_box.append(7)
    if box_flag[2] == 0:
        possible_box.append(2)
    if box_flag[9] == 0:
        possible_box.append(9)
elif p1 == 4:
    if box_flag[5] == box_flag[9] == 1:
        possible_box.append(10)
    elif box_flag[9] == box_flag[10] == 1:
        possible_box.append(5)
    else:
        if box_flag[9] == 0:
            possible_box.append(9)
        if box_flag[10] == 0:
            possible_box.append(10)
        if box_flag[5] == 0:
            possible_box.append(5)
elif p1 == 9:
    if box_flag[4] == box_flag[5] == 1:
        possible_box.append(10)
    elif box_flag[4] == box_flag[10] == 1:
        possible_box.append(5)
    else:
        if box_flag[5] == 0:
            possible_box.append(5)
        if box_flag[10] == 0:
            possible_box.append(10)
        if box_flag[5] == box_flag[10] == 1:
            if box_flag[3] == 0:
                possible_box.append(3)
            if box_flag[8] == 0:
                possible_box.append(8)
```

```
        if p1 == 5:
            if box_flag[4] == 0:
                possible_box.append(4)
            if box_flag[9] == 0:
                possible_box.append(9)
            if box_flag[10] == 0:
                possible_box.append(10)
            if box_flag[4] == box_flag[9] == box_flag[10] == 1:
                    if box_flag[3] == 0:
                        possible_box.append(3)
                    if box_flag[8] == 0:
                        possible_box.append(8)
        elif p1 == 10:
            if box_flag[4] == 0:
                possible_box.append(4)
            if box_flag[9] == 0:
                possible_box.append(9)
            if box_flag[4] == box_flag[9] == 1:
                    if box_flag[3] == 0:
                        possible_box.append(3)
                    if box_flag[8] == 0:
                        possible_box.append(8)

        p2 = random.sample(possible_box,1)
        p2 = p2[0]
        box_flag[p2] = 1
        p2x, p2y = XY(p2)
        pq.append(p2x*1000+p2y)
        print 'Second bend : ', possible_box, p2
        draw.line([(p1x,p1y),(p2x,p2y)],'#FFFFFF', 3)

    possible_box = []
    numbers_flag += 1
    if numbers_flag < 4:
        if p2 == 1 or p2 == 6:
            if numbers[0] == '-':
                possible_box.append(0)
            if numbers[1] == '-':
                possible_box.append(1)
        elif p2 == 2 or p2 == 7:
            if numbers[0] == '-':
                possible_box.append(0)
            if numbers[1] == '-':
                possible_box.append(1)
            if numbers[2] == '-':
                possible_box.append(2)
        elif p2 == 3 or p2 == 8:
```

```
                if numbers[1] == '-':
                    possible_box.append(1)
                if numbers[2] == '-':
                    possible_box.append(2)
                if numbers[3] == '-':
                    possible_box.append(3)
            elif p2 == 4 or p2 == 9:
                if numbers[2] == '-':
                    possible_box.append(2)
                if numbers[3] == '-':
                    possible_box.append(3)
                if numbers[4] == '-':
                    possible_box.append(4)
            elif p2 == 5 or p2 == 10:
                if numbers[3] == '-':
                    possible_box.append(3)
                if numbers[4] == '-':
                    possible_box.append(4)
        else:
            for b in range(5):
                if numbers[b] == '-':
                    possible_box.append(b)
        n = random.sample(possible_box,1)
        n = n[0]
        print 'Number position :', possible_box, n
        possible_box = []
        numbers[n] = self.mapping[j]

    for i in range(5):
        m = random.randrange(50)+25
        n = random.randrange(25)+180
        self.integer_coordinates[numbers[i]] = (m+(i*100)+10)*100 + n-180

    for i in range(5):
        if i == 0:
            j = self.characters[i]
        elif i == 1:
            j = self.characters[4]
        else :
            j = self.characters[i-1]
        p = pq[i] // 1000
        q = pq[i] % 1000
        y = self.integer_coordinates[self.mapping[j]]
        r = y // 100
        s = y %100 + 180
        draw.line([(p,q),(r,s)],'#FFFFFF', 3)
```

```
        image = image.filter(ImageFilter.BLUR)
        image = image.filter(ImageFilter.SMOOTH_MORE)
        draw = ImageDraw.Draw(image)

        for i in range(5):
            j = self.characters[i]
            x = self.character_coordinates[j]
            m = x//100
            n = x % 100
            y = self.integer_coordinates[self.mapping[j]]
            p = y // 100
            q = y %100 + 180
            draw.text((m-20,n-40),j,'#00C00B', self.font)
            draw.text((p-20,q+10),self.mapping[j],'#00C00B', self.font)

        del draw
        img = QtGui.QImage(image.tostring(),500,250,QtGui.QImage.Format_RGB888)
        del image
        return img

class GUI(QtGui.QWidget):

    def __init__(self):

        super(GUI, self).__init__()
        self.initUI()

    def initUI(self):

        label1 = QtGui.QLabel('Based on the image, answer the following questio

        self.label3 = QtGui.QLabel(self)

        label2 = QtGui.QLabel('Enter the mapping of ABCDE : ', self)
        answer = QtGui.QLineEdit(self)

        button1 = QtGui.QPushButton("Generate New Image", self)
        self.connect(button1,QtCore.SIGNAL('clicked()'),self.NewImage)
        button2 = QtGui.QPushButton("Validate", self)
        button2.setDefault(1)
        button3 = QtGui.QPushButton('About',self)
        self.connect(button3,QtCore.SIGNAL('clicked()'),self.about)

        layout = QtGui.QGridLayout(self)
        layout.setSpacing(10)
        layout.addWidget(label1,1,1,1,4)
        layout.addWidget(self.label3,2,1,2,4)
```

```
        layout.addWidget(label2,4,1,4,2)
        layout.addWidget(answer,8,1)
        layout.addWidget(button2,8,2)
        layout.addWidget(button1,8,3)
        layout.addWidget(button3,8,4)
        self.setLayout(layout)
        self.setWindowTitle('XCaptcha '+VERSION)
        self.NewImage()

    def NewImage(self):
        print 'DEBUG INFORMATION'
        img = captcha()
        img.generate_data()
        image = img.generate_image()
        self.label3.setPixmap(QtGui.QPixmap(image))

    def about(self):
        heading = '<center><h3>XCaptcha '+VERSION+'</h3></center>'
        line1 = 'Based on the paper by :'
        dev0 = '<ul><li><a href="mailto:arvind.einstein101@gmail.com">Arvind S.A
        line2 = 'Developed by :'
        dev1 = '<a href="mailto:shahanamamutty@gmail.com">Shahana Hamza Mammutty
        dev2 = '<a href="mailto:sruthikampurath@gmail.com">Sruthy K</a> (S6 CSE)
        dev3 = '<a href="mailto:mulloli@me.com">Varun M</a> (S6 CSE)'
        line3 = '<ul><li>'+dev1+'</li><li>'+dev2+'</li><li>'+dev3+'</li></ul>'
        QtGui.QMessageBox.information(self, "Information",heading+line1+dev0+lin

def main():
    app = QtGui.QApplication(sys.argv)
    ex = GUI()
    ex.show()
    sys.exit(app.exec_())

if __name__=='__main__' :
    main()
```

Chapter 5

Testing and Implementation

5.1 Testing methods

There are two main approaches to testing:

- Black box testing: (functional testing) It is based on the definition of what a program is intended to do i.e. it is based on the programs specification, rather than on its structure. The tester is only aware of what the software is supposed to do, but not how i.e. when he enters a certain input, he gets a certain output; without being aware of how the output was produced in the first place. Test cases are built around specifications and requirements, i.e., what the application is supposed to do. It uses external descriptions of the software, including specifications, requirements, and designs to derive test cases. These tests can be functional or non-functional, though usually functional. The test designer selects valid and invalid inputs and determines the correct output. There is no knowledge of the test object's internal structure.

- White box testing: White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing). In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs.

Table 5.1: Black box testing

Sl no	**Test Cases**	**Expected Output**	**Whether obtained the expected output or not**
1	The user opens a form in the web browser	• A web based form appears along with the CAPTCHA. • CAPTCHA consists of image,question and answer box.	Yes
2	User types in the answer box and clicks the validate button	• The page refreshes and displays whether the answer is right or not.	Yes

Table 5.2: White box Testing

Sl no	**Test Cases**	**Function**	**Expected Output**	**Whether obtained the expected output or not**
1	Function coordinate	This function takes in four arguments and returns a point which is enclosed within the rectangle bounded by the given arguments	A point is returned	Yes
2	Function XY(P)	This function converts the nomenclature surreal boxes in the CAPTCHA image in to its coordinate implementation and selects a point inside that box by calling the coordinate function	A point is returned	Yes

Sl no	Test Cases	Function	Expected Output	Whether obtained the expected output or not
3	Function generate data	Selects five characters from a port of five numbers from a pool of numbers and randomly maps themselves	returns the characters, numbers and mappings	Yes
4	Function generate image	According to tha caharacters, numbers and mappings by the generate data function(),generate an image by drawing lines between them	image is returned	Yes
5	Function generate question	Randomise the order of characters and generate the corresponding answer according to the new order	question is returned	Yes
6	Check Answer	Accept input from the user and compare with the correct answer	Yes	
7	Function Authorisation	According to the value returned, provide or deny the access	Yes	

5.2 Advantages and Limitations

5.2.1 Advantages

1. Its a new approach.
2. Its web friendly.
3. It is a Free software
4. It can run on any OS.
5. It has low implementation cost.
6. by consecutive operation the person gets used to the new system.

5.2.2 Limitations

1. It takes longer sorting time.

5.3 Future Extensions if possible

- Character wrapping.
- Instead of characters and numbers icons can also be used.
- Varying the pattern of question.

Chapter 6

Conclusion

A CAPTCHA was found to be highly useful, when the level of security required was elevated.The downside being a longer solving time.

Annexure

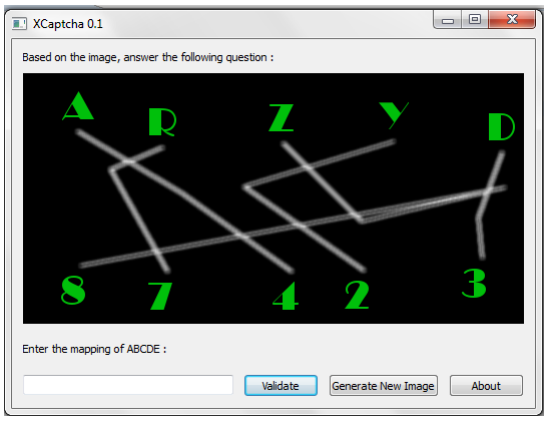


Figure 6.1: Annexure 1

Bibliography

[1] Peter Hall, Peter Andreae, and Milton Ngan, *Reconstruction of blood vessel networks from a few perspective projections*, Conference Proceedings on Second New Zealand International Two-Stream, 1995, pp. 369–372.

[2] Yong-Lin Hu, W.J. Rogers, D.A. Coast, C.M. Kramer, and N. Reichek, *Vessel boundary extraction based on a global and local deformable physical model with variable stiffness*, Magnetic Resonance Imaging **16** (1998), no. 8, 943–951.

[3] Alan Watt and Fabio Policarpo, *The computer image*, Addison-Wesley, 1998.