Group 1

Software Design Document

Aishwarya Agarwal (130101003)
Alamanda Nikhil Teja (130101005)
Sowmya k (130101028)
Kanhaiya (130101033)
Kodali Hari Krishna Sai (130101037)
K M Sai Krishna(130101039)
Kunal Jain (130101042)
Sudhanshu(130101074)
Bhanu Prakash(130101076)
Vivek Kumar (130101079)

Date: (18/01/2015)

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Purpose

To develop a captcha generator which is capable of differentiating between a machine's input and human input. This can also be used by various developers who wish to use a captcha input in their softwares/websites.

1.2 Scope

- Initial versions will be there to check different types of distorted captcha's.
- Can be used by Developers who want to distinguish between human users and robot users.
- Later version can also be used either as a user manual or a plugin by other developers working on Captcha generation.
- Later versions can also include the self learning captcha.

1.3 Overview

Checks the randomly generated captcha and states to the user whether it is right or wrong.

2. SYSTEM OVERVIEW

The software runs on minimal hardware. It has no hard dependencies, works on windows platform.

3. Software ARCHITECTURE

3.1 Architectural Design

This software is based on a modular design. Various objects, classes, functions, event handlers, etc. bind the software together.

3.2 Decomposition Description

Event Handlers

- *OK button on click*

```
ON_COMMAND(IDC_BUTTON1,start)
```

On clicking OK button it checks the entered text (by user) with the captcha generated and a dialogue box appears stating whether the user is correct or incorrect.

- *Refresh button on click*
  It overwrites the existing captcha with a freshly generated image. Now, a new captch appears.

Functions

- *Initialise*
  `Function void Captcha::init()`
  This function initializes the array letters. It stores the start and end points of the characters already in font.hpp.

- *Background generation*
  `Function void Captcha::backgroundgen()`
  This is to set the Background colour as we want. It accesses the RGB values of every pixel.

- *Random generate*
  `Function void Captcha::gen_random(char *s, const int len)`
  This generates a random 5 letter string consisting of alphabets lowercase (a-z).

- *Captcha generation*
  `Function void Captcha::captchagen()`
  The random string that is generated is mapped to a bitmap by assigning RGB values to each pixel.

- *Noise addition*
  `Function void Captcha::ran_noiseadd()`
  It adds random pixels on the bitmap to create noise in it.

- *Saving*
  `Function void Captcha::bitsave()`
  This is to save the Bitmap image.

Libraries Used

- *Font.hpp*
  It contains a 2-d array ‘im’ which contains structural information of the shape of each characrer.
- *Md5.hpp*
  This library contains the md5 hash class for hashing.
- *Bitmap_image.hpp*
  It contains bitmap_image class which allows us to generate a bitmap image and save it.
- *Resource.hpp*

- *Mfc static library*

3.3 Design Rationale

We used object orientated approach because –

- It helps in organizing code
- Easier to work when there are many people involved in software development
- Easier to debug as a particular functionality has his own function

4. DATA DESIGN

4.1 Data Description

Globally defined:

```
CEdit * pInput;    - Input textbox

string str;        - contains the present captcha string which is randomly
                     generated. This is cleared after the string is hashed.
```

4.2 Data Dictionary

```
int letters[26][2];    -       letters stores the range of each character
                               present in font.hpp.

bitmap_image image;    -       it's an instance of bitmap_image class

string saveFile;       -       it's the name of the image you want to store.


Captcha(string _saveFile);     this is a constructor of captcha class.

string _str;             -     Randomly generated string

string hashedString;     -     stores the output of the hashed string str
```

5. HUMAN INTERFACE DESIGN

5.1 Overview of User Interface

The user interface is minimalistic. More emphasis is laid on functionality.
The UI contains one form having captcha generated in between it and two button, one is Refresh and other is OK.

5.2 Screen Images

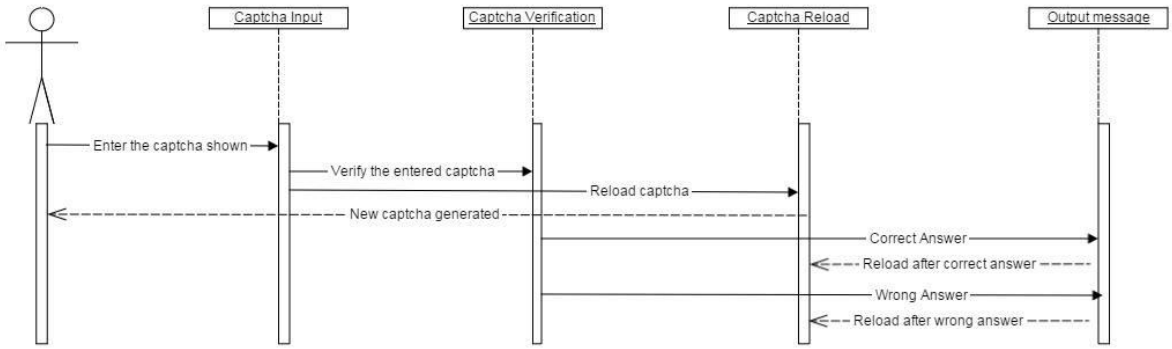
Refer to user manual for same.

5.3 Screen Objects and Actions

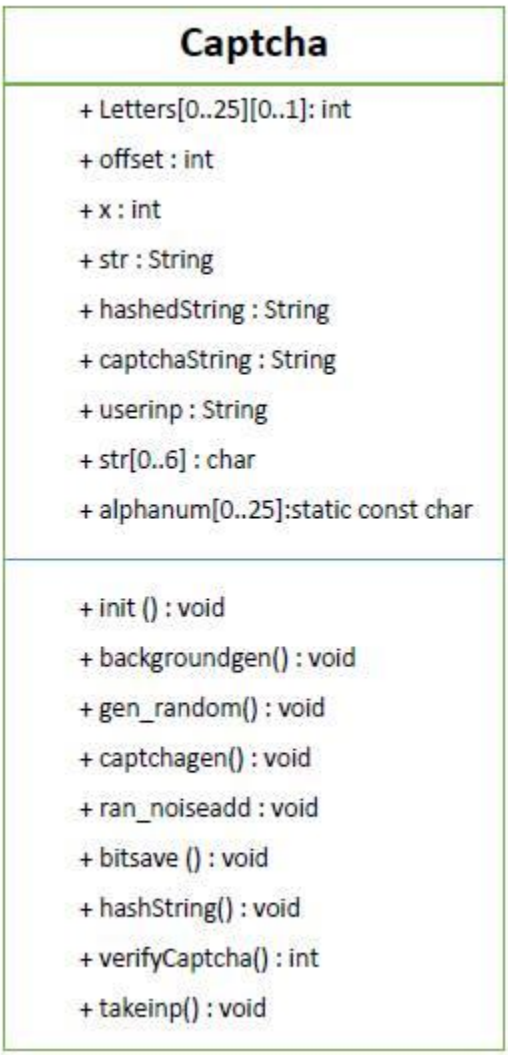
Refer to user manual for same.

6. UML Diagrams

6.1 Sequence UML



6.2 Class UML


Captcha
+ Letters[0..25][0..1]: int
+ offset : int
+ x : int
+ str : String
+ hashedString : String
+ captchaString : String
+ userinp : String
+ str[0..6] : char
+ alphanum[0..25]:static const char
+ init () : void
+ backgroundgen() : void
+ gen_random() : void
+ captchagen() : void
+ ran_noiseadd : void
+ bitsave () : void
+ hashString() : void
+ verifyCaptcha() : int
+ takeinp() : void