IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

SECURITY OF WEB PAGES USING CAPTCHA TECHNIQUE

DASARI BENRJEE [#1], V.SARALA [#2]

[#1] MCA Student, Master of Computer Applications,

D.N.R. College, P.G.Courses & Research Center, Bhimavaram, AP, India.

[#2] Assistant Professor, Master of Computer Applications,

D.N.R. College, P.G.Courses & Research Center, Bhimavaram, AP, India.

ABSTRACT

In general for most of the web sites, there will be registration page for creating a profile and all the details need to be filled by the user who wish to register. But in some conditions the user may miss some fields empty and at that situation robots try to enter their illegal contents inside those fields and change the original contents. Hence there is no proper authentication for web pages in order to identify whether they are filled by user or robots. This project is a solution for improving the security of web forms from web bots (robots). This project delivers the best result than the existing system. The emphasis is on the CAPTCA (Completely Automated Public Turing test to tell Computers and Humans Apart) method that is most often used method. We are implementing a security method in java web programming.

1. INTRODUCTION

A CAPTCHA is a program that protects websites against bots by generating and grading tests that humans can pass but current computer programs cannot. For example, humans can read distorted text as the one shown below, but current computer programs can't The term CAPTCHA (for Completely Automated Turing Test To Tell Computers and Humans Apart) was coined in 2000 by Luis von Ahn, Manuel Blum, Nicholas Hopper and John Langford of Carnegie Mellon University. At the time, they developed the first CAPTCHA to be used by Yahoo

CAPTCHAs have several applications for practical security, including (but not limited to): Preventing Comment Spam in Blogs. Most bloggers are familiar with programs that submit bogus comments, usually for the purpose of raising search engine ranks of some website (e.g., "buy penny stocks here"). This is called comment spam. By using a CAPTCHA, only humans can enter comments on a blog. There is no need to make users sign up before they enter a comment, and no legitimate comments are ever lost!

Protecting Website Registration. Several companies (Yahoo!, Microsoft, etc.) offer free email services. Up until a few years ago, most of these services suffered from a specific type of attack: "bots" that would sign up for thousands of email accounts every minute. The solution to this problem was to use CAPTCHAs to ensure that only humans obtain free accounts. In general, free services should be protected with a CAPTCHA in order to prevent abuse by automated scripts.

Protecting Email Addresses From Scrapers. Spammers crawl the Web in search of email addresses posted in clear text. CAPTCHAs provide an effective mechanism to hide your email address from Web scrapers. The idea is to require users to solve a CAPTCHA before showing your email address. A free and secure implementation that uses CAPTCHAs to obfuscate an email address can be found at reCAPTCHA MailHide.

Online Polls. In November 1999, http://www.slashdot.org released an online poll asking which was the best graduate school in computer science (a dangerous question to ask over the web!). As is the case with most online polls, IP addresses of voters were recorded in order to prevent single users from voting more than once. However, students at Carnegie Mellon found a way to stuff the ballots using programs that voted for CMU thousands of times. CMU's score started growing rapidly. The next day, students at MIT wrote their own program and the poll became a contest between voting "bots." MIT finished with 21,156 votes, Carnegie Mellon with 21,032 and every other school with less than 1,000. Can the result of any online poll be trusted? Not unless the poll ensures that only humans can vote.

Preventing Dictionary Attacks. CAPTCHAs can also be used to prevent dictionary attacks in password systems. The idea is simple: prevent a computer from being able to iterate through the entire space of passwords by requiring it to solve a CAPTCHA after a certain number of unsuccessful logins. This is better than the classic approach of locking an account after a sequence of unsuccessful logins, since doing so allows an attacker to lock accounts at will.

Search Engine Bots. It is sometimes desirable to keep webpages unindexed to prevent others from finding them easily. There is an html tag to prevent search engine bots from reading web pages. The tag, however, doesn't guarantee that bots won't read a web page; it only serves to say "no bots, please." Search engine bots, since they usually belong to large companies, respect web pages that don't want to allow them in. However, in order to truly guarantee that bots won't enter a web site, CAPTCHAs are needed.
Worms and Spam. CAPTCHAs also offer a plausible solution against email worms and spam: "I will only

accept an email if I know there is a human behind the other computer." A few companies are already marketing this idea.

If your website needs protection from abuse, it is recommended that you use a CAPTCHA. There are many CAPTCHA implementations, some better than others. The following guidelines are strongly recommended for any CAPTCHA code:

Accessibility. CAPTCHAs must be accessible. CAPTCHAs based solely on reading text — or other visual-perception tasks — prevent visually impaired users from accessing the protected resource. Such CAPTCHAs may make a site incompatible with Section 508 in the United States. Any implementation of a CAPTCHA should allow blind users to get around the barrier, for example, by permitting users to opt for an audio or sound CAPTCHA.

Image Security. CAPTCHA images of text should be distorted randomly before being presented to the user. Many implementations of CAPTCHAs use undistorted text, or text with only minor distortions. These implementations are vulnerable to simple automated attacks.

Script Security. Building a secure CAPTCHA code is not easy. In addition to making the images unreadable by computers, the system should ensure that there are no easy ways around it at the script level. Common examples of insecurities in this respect include: (1) Systems that pass the answer to the CAPTCHA in plain text as part of the web form. (2) Systems where a solution to the same CAPTCHA can be used multiple times (this makes the CAPTCHA vulnerable to so-called "replay attacks"). Most CAPTCHA scripts found freely on the Web are vulnerable to these types of attacks.

Security Even After Wide-Spread Adoption. There are various "CAPTCHAs" that would be insecure if a significant number of sites started using them. An example of such a puzzle is asking text-based questions, such as a mathematical question ("what is 1+1"). Since a parser could easily be written that would allow bots to bypass this test, such "CAPTCHAs" rely on the fact that few sites use them, and thus that a bot author has no incentive to program their bot to solve that challenge. True CAPTCHAs should be secure even after a significant number of websites adopt them.

2. LITERATURE SURVEY

Literature survey is the most important step in software development process. Before developing the tool, it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, then next steps are to determine which operating system and language used for developing the tool. Once the programmers start building the tool, the programmers need lot of external support. This support obtained from senior programmers, from book or from websites. Before building the system the above consideration r taken into account for developing the proposed system.

Whether a captcha is based on pictures, text, sound, or puzzle-solving, certain similarities can be seen in terms of how captchas are attacked by malicious users. Typical attack models seen to date include: Bypass attacks Any attack that circumvents the need to solve the captcha at all. For example, network replay attacks, or cases where the captcha solution is exposed accidentally, perhaps through HTML or CGI form parameter values. Generally, any system that sends the decoded form of the captcha to a client program as part of the data stream is vulnerable to such an attack. Such attacks are not always a weakness of the captcha itself; they may instead be a weakness of the service using the captcha.

Challenge replay attacks If the captcha system can produce only a limited number of unique challenges, then the automated agent may record all or most of the possible challenges. A human associate provides a library of correct answers for the challenges. The automated agent can then replay the correct answer whenever it is faced with a particular challenge for which it knows the correct solution. Some image-based captchas are vulnerable to this weakness, particularly those based upon a finite library of photographs (e.g. the 'KittenAuth' captcha (Reimer, 2006) used a challenge library of 42 images). 4 Signal processing attacks The noise and perturbations that are commonly used to obfuscate captcha images or sounds are intended to be oneway; a computer should be able to add them, but not reverse them easily. In principle, only a human's flexible image and sound recognition capabilities should be capable of conveniently reversing the transformations and recovering the original message.

In practice, captcha researchers and malicious attackers have both proven highly capable of reversing captcha transformations approximately via mathematical heuristics and machine learning approaches (Chellapilla and Simard, 2005; Hocevar, 2004; Huang et al., 2008; Mori and Malik, 2003; Yan and Ahmad, 2007, 2008a). For text-based captchas, this is achieved by removing image noise and clutter items, and isolating individual characters within the captcha in order to allow optical character recognition (OCR) technologies a maximised opportunity for success. Attacking heuristics often have a parameterised design, so that their behaviour may be adjusted to attack several different but related forms of captcha. Mechanical Turk attacks Here, the problem of solving the captcha is automatically 'outsourced' to a paid human agent. They immediately solve the challenge and quickly return the answer to the automated agent in real time.

The automated agent then presents the human-provided answer, and is able to programatically exploit the online resource (Barr and Cabrera, 2006; Bursztein et al., 2010; The Economist Newspaper Ltd., 2008; Websense Security Labs, 2008b). A human 'Turk' agent working full time to support such attacks can solve thousands of captchas per hour, depending on the type of captcha. There is little that can be done to defend against such attacks, other than to perhaps raise the inconvenience of the captcha for all users in order to reduce the economic viability of this attack. Generally, an increase in inconvenience can be achieved either by increasing the difficulty of the captcha for humans, or by requiring users to regularly re-authenticate themselves as human. Trivial guessing attacks If there is an unlimited range of challenges, but a very limited range of possible answers (e.g. 'which of these 10 choices is correct?'), a high success rate may be achieved by an attacking program by merely guessing randomly from the available answers. Particularly, any graphical captcha that requires the user to select a correct position within 5 an image - but which has a wide error tolerance for user inaccuracy - may be vulnerable to a trivial guessing attack.

Brute force attacks If there is a somewhat limited range of possible answers - e.g. a numerical 4-digit captcha would have 10,000 possible answers - then it is possible for a distributed group of automated agents to attack the captcha by exhaustively trying answers at random or according to a selected sequence. This differs from the 'trivial guessing attack', in that it relies upon having access to a large number of attacking agents - i.e. a 'botnet' (Websense Security Labs, 2008b, 2009) - rather than relying upon having access to a poorly designed captcha. Hybrid attacks It is possible to combine these attacks. For example, if a signal processing attack can estimate 5 of 6 captcha characters with a high degree of confidence, a guess may be made on the remaining character, yielding a success rate of between 1.5% (mixed case alphanumerical characters) and 10% (numerical digits). For example, the 'Question-Based captcha' (Shirali-Shahreza and Shirali-Shahreza, 2007b) presents a mathematical problem, which can be broken by an attacker who uses OCR to recognise the numerical digits mentioned in the puzzle, combined with a random guess of one of the few possible ways in which the numbers may be combined arithmetically.

3. EXISTING SYSTEM

In the existing system almost all the websites try to use normal mode of data insertion into the database without using any third party verification mechanism. Hence there are lot of limitations which are occurred in the existing web pages. They are as follows:

LIMITATION OF EXISTING SYSTEM

The following are the limitation of existing system. They are as follows:

1. All the spammers try to enter illegal comments into their accounts such as Emails, Social Networks and so on.
2. Some hackers may generate the robots or hacking programs and may create some sort of web site designs and ask the users to fill their basic details including bank details.

4. PROPOSED SYSTEM

This project is a solution for improving the security of web forms from web bots (robots). This project delivers the best result than the existing system. The emphasis is on the CAPTCA (Completely Automated Public Turing test to tell Computers and Humans Apart) method that is most often used method. We are implementing a security method in java web programming.Here we can clearly differentiate whether the query is inserted by Humans or Robots into the system.

ADVANTAGES OF THE PROPOSED SYSTEM

The following are the advantages of the proposed system. They are as follows:

1) Security for the information which is entered during registration or login time.
2) The system can easily identify whether the user credentials are filled by the user or by System which writes robots for filling the data.
3) Only valid users can only see the captcha word and they can substitute the word as it is by typing from the keyboard.
4) As we know captcha word will be keep on dynamically changes for every individual application, there is a lot of security for the end users data

5. SOFTWARE PROJECT MODULES

Implementation is the stage where the theoretical design is converted into programmatically manner. In this stage we will divide the application into a number of modules and then coded for deployment. We have implemented the proposed concept on Java programming language with JEE as the chosen language in order to show the performance this proposed application. The front end of the application takes JSP,HTML and Java Beans and as a Back-End Data base we took My-SQL Server manner. The implementation will be divided into number of modules like 2 modules

1. Registration Module
2. Captcha Generation Module

Now let us discuss about each and every module in detail as follows:

5.1 Registration Module

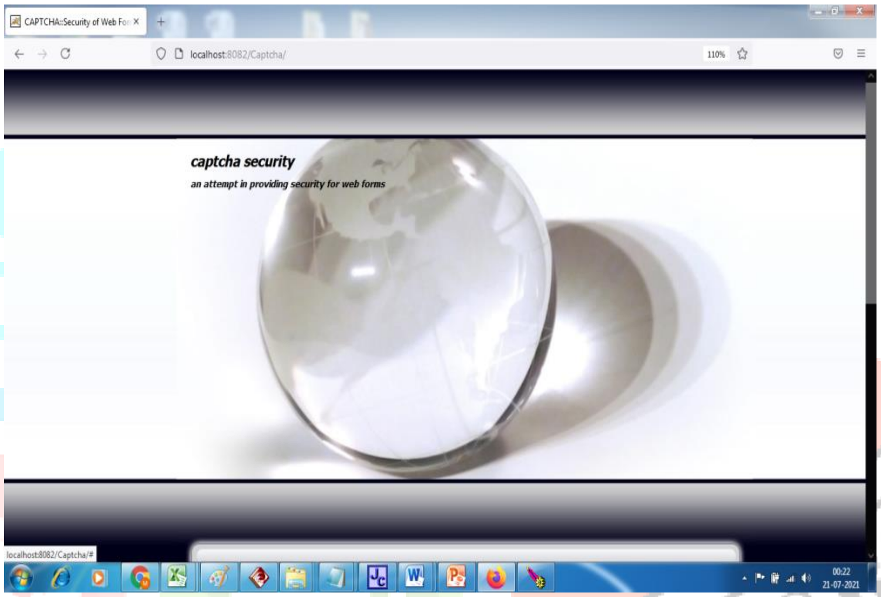
In this module, the user is one who is mainly fill all his valid details like Firstname,Last name,Email id,Phone number,Pincode,Address and so on.Once he fill all the details,he is asked to enter the CAPTCHA word which is present bottom of that page .If the user enters the CAPTCHA word correctly then he will get a message like registration is successful.If the same user don’t substitute the CAPTCHA word correctly,then he will get a message like “ Registration Failed Due to Wrong Captcha Substitution”.

5.2 Captcha Generation Module

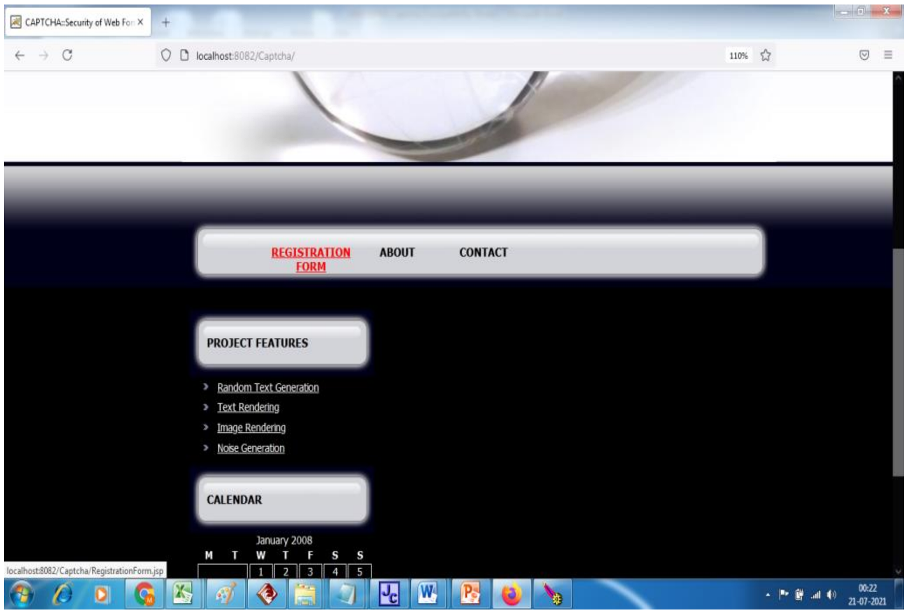
In this module, Once the user want to store his basic details into database,he is asked to enter captcha word.This word is randomly changed from one form to another.This is basically created in the form of image which contains sequence of letters and numbers.So in order to generate this captcha word randomly we are writing the logic for random generation of captcha codes and as we need to display the word in image file.For that reason we are using imaging jar file for integrating the properties required for image generation.

6. OUTPUT SCREENS

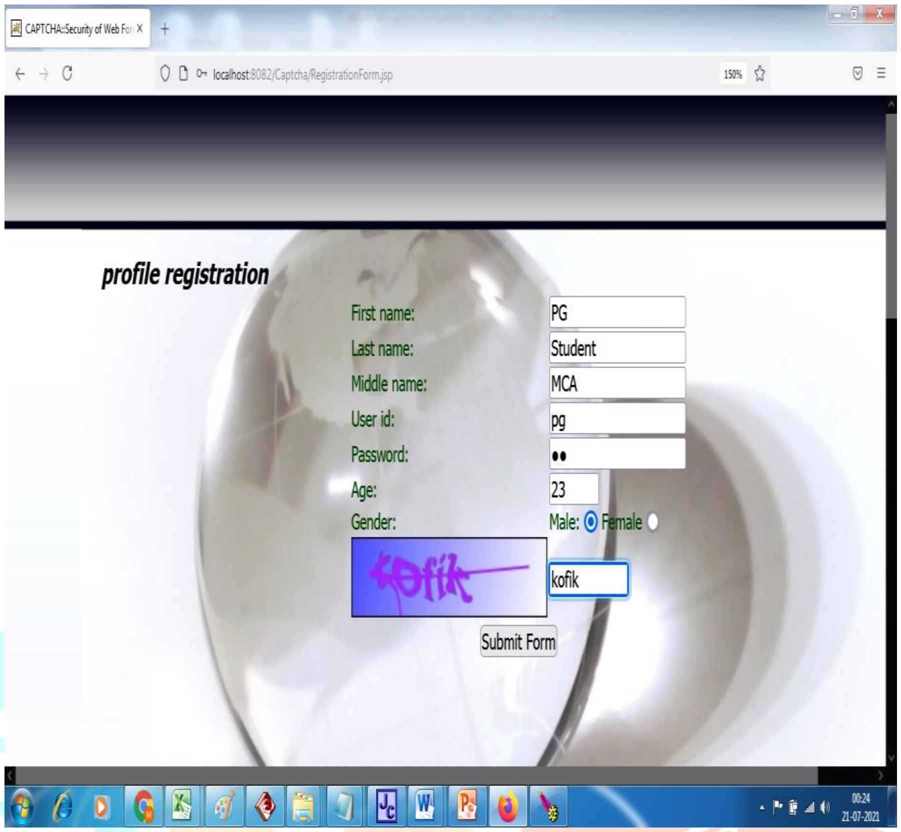
HOME PAGE


CAPTCHA::Security of Web For
localhost:8082/Captcha/
captcha security
an attempt in providing security for web forms
localhost:8082/Captcha/#

CUSTOMER REGISTRATION



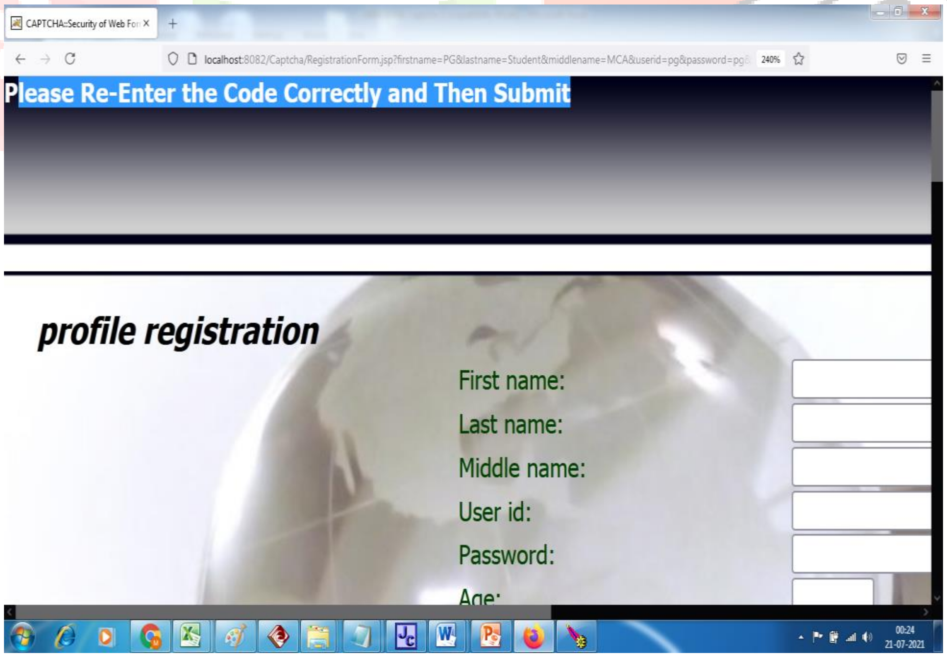
CUSTOMER REGISTRATION


CAPTCHA::Security of Web For
localhost:8082/Captcha/RegistrationForm.jsp
150%
profile registration

First name:	PG
Last name:	Student
Middle name:	MCA
User id:	pg
Password:	••
Age:	23
Gender:	Male: [✓ ●] Female ○
	kofik

Submit Form
00:24
21-07-2021

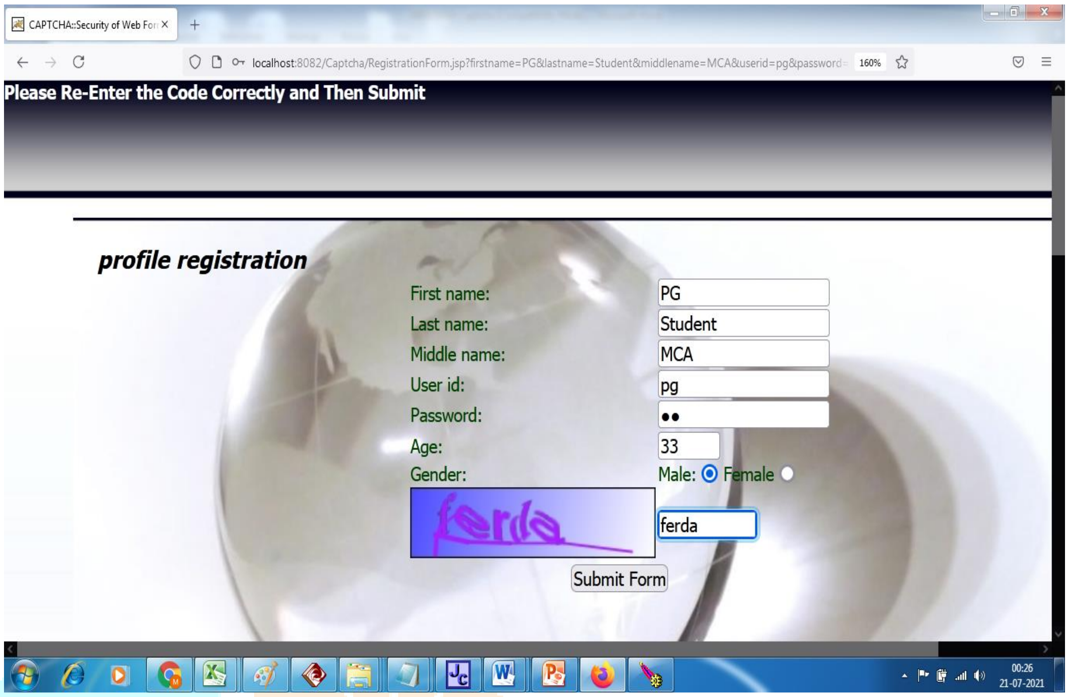
USER GOT ERROR


CAPTCHA::Security of Web For
localhost:8082/Captcha/RegistrationForm.jsp?firstname=PG&lastname=Student&middlename=MCA&userid=pg&password=pg
240%

Please Re-Enter the Code Correctly and Then Submit

profile registration
First name:
Last name:
Middle name:
User id:
Password:
Age:
00:24
21-07-2021

USER TRY TO RE-REGISTER

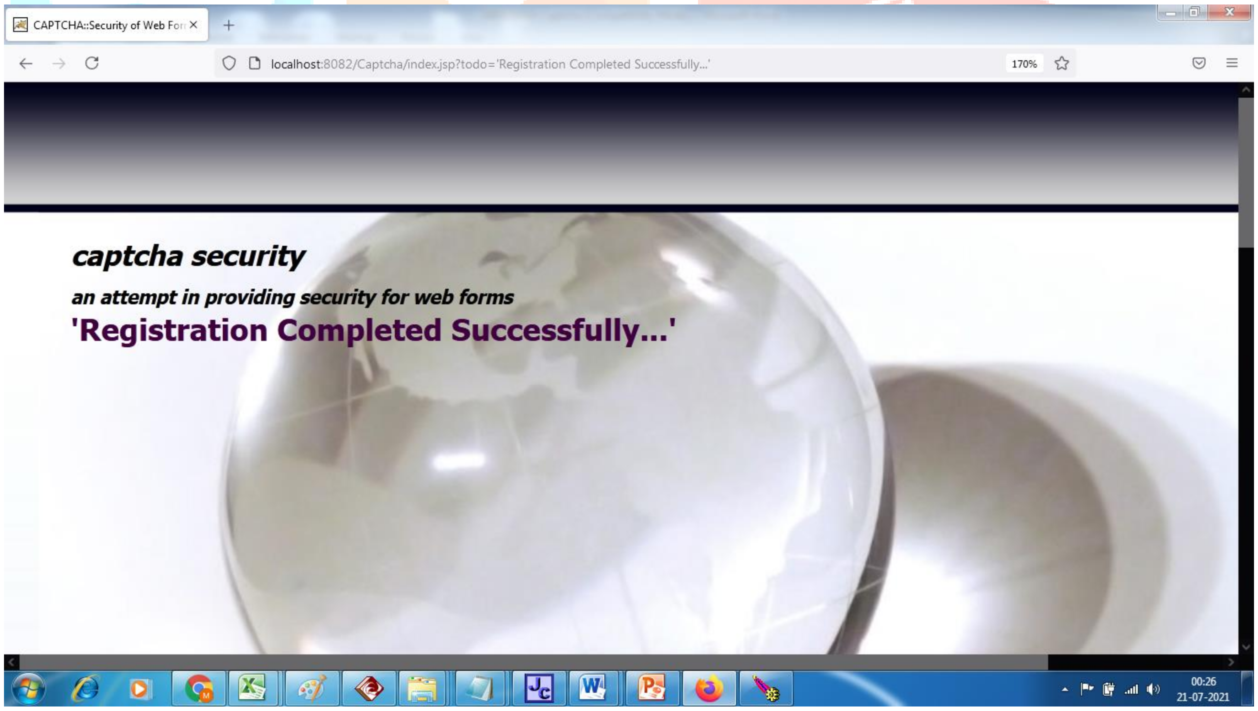

localhost:8082/Captcha/RegistrationForm.jsp?firstname=PG&lastname=Student&middlename=MCA&userid=pg&password=

Please Re-Enter the Code Correctly and Then Submit

profile registration

First name:	PG
Last name:	Student
Middle name:	MCA
User id:	pg
Password:	••
Age:	33
Gender:	Male: [✓] Female
ferda	ferda

USER GETS REGISTRATION SUCCESS


localhost:8082/Captcha/index.jsp?todo='Registration Completed Successfully...'

captcha security

an attempt in providing security for web forms

'Registration Completed Successfully...'

7. CONCLUSION

In this project we for the first time showed how the registration pages are applied with CAPTCHA technique in order to differentiate whether the process is done by human or machine and also we try to differentiate the similarity between several captcha codes which are uniquely generated for individual applications.

8. REFERENCES

[1] von Ahn, L., Blum, M., Hopper, N.J., Langford, J.: CAPTCHA, "using hard ai problems for security". In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 294–311. Springer, Heidelberg (2003)

[2] Przydatek, B., "On the (im)possibility of a text-only CAPCHA". In: First Workshop on Human Interactive Proofs (unpublished Abstract, 2002), available.

[3] Chan, T.-Y., "Using a text-to-speech synthesizer to generate a reverse turing test". In: Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence, p. 226. IEEE Computer Society Press, Los Alamitos (2003)

[4] Luis von Ahn, Manuel Blum and John Langford, "Telling Humans and Computers part (Automatically) or How Lazy Cryptographers do AI". To appear in Communications of the ACM.

[5] Liu, P., Shi, J., Wang, L., Guo, L., "An efficient ellipse-shaped blobs detection algorithm for breaking Facebook CAPTCHA". In: Yuan, Y., Wu, X., Lu, Y. (eds.) ISCTCS 2012. CCIS, vol. 320, pp. 420–428. Springer, Heidelberg (2013)

[6] Ahmad, A.S.E., Yan, J., Marshall, L., "The robustness of a new CAPTCHA". In: EUROSEC, pp. 36–41 (2010)

[7] [8] Ahmad, A.S.E., Yan, J., Ng, W.-Y., "CAPTCHA design: Color, usability, and security". IEEE Internet Computing 16(2), 44–51 (2012)

[9] Baecher, P., Büscher, N., Fischlin, M., Milde, B., "Breaking reCAPTCHA: A holistic approach via shape recognition". In: Camenisch, J., Fischer-Hübner, S., Murayama, Y., Portmann, A., Rieder, C. (eds.) SEC 2011. IFIP AICT, vol. 354, pp. 56–67. Springer, Heidelberg (2011)

[10] Bursztein, E., Martin, M., Mitchell, J.C., "Text-based CAPTCHA strengths and weaknesses". In: Chen, Y., Danezis, G., Shmatikov, V. (eds.) ACM Conference on Computer and Communications Security, pp. 125–138. ACM (2011)

[11] Huang, S.-Y., Lee, Y.-K., Bell, G., Ou, Z.-H., "An efficient segmentation algorithm for CAPTCHAs with line cluttering and character warping. Multimedia Tools and Applications" 48(2), 267–289 (2010)

[12] Wilkins, J., "Strong CAPTCHA" guidelines v1.2 (2009), http://www.bitland.net/captcha.pdf