CPS 595 - SOFTWARE ENGINEERING PROJECT

CAPTCHA - FINAL REPORT

Venkata Sai Prasanth Ippagunta

ippaguntav1@udayton.edu

101516438

CAPTCHA

CAPTCHA stands for **Completely Automated Public Turing Test to tell Computers and Humans Apart**.

It is used as a security check to ensure only humans can pass and are authenticated to perform an action. The computers and bots generally cannot pass through the CAPTCHA test.

The implementation and hard read nature makes it tough for bots to solve the CAPTCHA's.

CAPTCHA's are now implemented by several websites before processing any transactions or submitting registration and online forms.

Different Approaches for CAPTCHA

There are many ways and techniques to achieve CAPTCHA.

The 3 approaches for implementing the CAPTCHA in the project includes validation based on Text, Images (visual) and Object Collage (Object Segments).

Text CAPTCHA:

The user enters the input in the form of text for displayed CAPTCHA.

Visual CAPTCHA:

The user provides the input by clicking one or more image blocks which satisfies the given statement

Collage CAPTCHA:

The user provides input in the form of a number as the answer for given statement. The person will need to check the images carefully for perfect count of objects asked.

All the 3 demos are based on Single page validation which means the page redirects to itself on clicking buttons submit or check.

Implementation

Text CAPTCHA:

The text captcha takes user entered text as input and validates the text with database values.

The implementation of the program is as follows

1. The captcha is displayed from a set of 50 samples randomly using PHP's rand(min,max) function.
2. The main page 'index.php' is a form with text box field displayed along with buttons to submit and refresh.
3. The connection to the database is provided by mysqli functions defined in external file named db_connection.php.
4. The db_connection.php file is included in index.php using **require_once()** function. The main purpose of this function is to include external files in the main page.
5. The styles for the elements are defined in external stylesheet named 'stylesheet.css'.
6. The links for the css and js files are referenced from an external file named 'header.php'.
7. When user enters some text and click submit, the page will redirect to itself but with image name (id) for the displayed image as a query parameter.
8. The image name is then retrieved using PHP's super global variable **$_GET**.
9. Using 'Select' query the answer for respective image is fetched from the database.
10. The user input is then compared with answer from the database and a message will be displayed based on the result.
11. If the user clicks on submit without any input the message 'Please fill out the field' will be displayed.
12. If the provided input matches with the database values, a message 'Validation Success' is displayed.
13. If the entered text does not match with the database values, a message 'Validation failed' is displayed.
14. Using jquery a timer is also set for 60 seconds which indicates that the captcha will reload if no input is provided by the user.
15. The query is executed with '**LIMIT 1**' condition which ensures fetching of only one row.
16. The user input is checked for any malicious scripts using **htmlspecialchars()** function which escapes some special characters like <, > etc.
17. **mysqli_real_escape_string()** is used to escape any special characters to use in an SQL statement.

Visual CAPTCHA:

The visual captcha takes values of selected image blocks and compares it with the answer from database. The implementation of the program is as follows.

1. The images are displayed in a 3X3 matrix which are shuffled and changed on every reload.
2. 15 image set samples were collected with a unique name given to each folder (1-15 in my case).
3. The connection to the database is provided by mysqli functions defined in external file named db_connection.php.
4. The styles for the elements are defined in external stylesheet named 'styles.css'.
5. The links for the css and js files are referenced to their respective files in **<head>** tag.
6. Each image set consists 9 images which are randomly changed using PHP's **rand(min,max)** function.
7. The image names (1-9) are stored in an array and shuffled using PHP's **shuffle()** function so that on every reload the image set and images will keep changing.
8. Jquery is used to evaluate the values which checks for the equality on clicking the button 'check'.
9. The question for respective image sets are retrieved from database using folder name.
10. Each image has given an **id = img-check**. This id is used by the click function as id selector which means the respective jquery code will be executed on clicking the image.
11. Jquery's **toggleClass()** is used to create red border around the selected image. This function creates a red border on 1st click and removes it on 2nd click on the same image.
12. The second click function is also used with id selector 'img-check' which generates the final array of selected images which is used for comparing with the database values.
13. The third click function is assigned to a div element which displays custom messages of success or failure upon clicking the check button.
14. If res is undefined or when result length is 0, a message 'You need to select atleast one image' is displayed.
15. If the selected image block values matched with the database values, a message 'Validation Success' will be displayed.
16. If the selected block values does not match with the database values, a message 'Validation failed' will be displayed.
17. A timer is also set for 60 seconds which indicates that the captcha will reload if no input is provided by the user.
18. The query is executed with '**LIMIT 1**' condition which ensures fetching of only one row.
19. The user input is checked for any malicious scripts using **htmlspecialchars()** function which escapes some special characters like <, > etc.
20. **mysqli_real_escape_string()** is used to escape any special characters to use in an SQL statement.

Object Collage CAPTCHA:

The inputs for the collage captcha are the integers (the count of objects) which are checked with the values in the database.

The implementation of the program is as follows.

1. A set of 3 images is displayed which consists of different objects that needs to be identified by the user.
2. 20 sample image sets are used having unique name for the folders.
3. Each image set consists of 3 images with unique file names.
4. The db_connection.php file is included in index.php using **require_once()** function. The main purpose of this function is to include external files in the main page.
5. The styles for the elements are defined in external stylesheet named 'styles.css'.
6. The image sets are changed randomly using PHP's **rand(min,max)** function.
7. A text box under each image is displayed for providing input.
8. When user provide the input and click check, the page will redirect to itself but with folder name (fn) for the displayed image set as a query parameter.
9. The folder name is then retrieved using PHP's super global variable **$_GET**.
10. Using 'Select' query the answer for respective image set is fetched from the database.
11. The question for the respective image set is also retrieved from database using folder name.
12. The user input is then compared with answer from the database and a message will be displayed based on the result.
13. If the selected image set values matched with the database values, a message 'Validation Success' will be displayed.
14. If the selected image set values does not match with the database values, a message 'Validation failed' will be displayed.
15. Using jquery a timer is also set for 60 seconds which indicates that the captcha will reload if no input is provided by the user.
16. The query is executed with '**LIMIT 1**' condition which ensures fetching of only one row.
17. The user input is checked for any malicious scripts using **htmlspecialchars()** function which escapes some special characters like <, > etc.
18. **mysqli_real_escape_string()** is used to escape any special characters to use in an SQL statement.

EVALUATION

The evaluation or user study is important for any project as it determines the performance and user views which helps the developers to make improvements and fix glitches to the application if necessary.

EVALUATION OF CAPTCHA

In this project the user should evaluate 10 sets of each text, visual and collage CAPTCHA's.

The user has to register his name before starting the evaluation.

The evaluation pages are kept separate from regular CAPTCHA demo's to enable auto-redirection of CAPTCHA types.

Text CAPTCHA Evaluation

1. The evaluation process will proceed as mentioned below,
2. The user has to click on the button 'Evaluate Captcha' on the text captcha page which is the home page for the project.
3. After clicking he needs to input his name to start evaluating
4. On hitting start, the evaluation begins with text captcha. The user need to enter the characters shown, in the text field.
5. On clicking submit a message saying success/failure will be displayed and captcha will reload indicating the user is ready to give his second attempt.
6. The captcha will reload on page refresh or on clicking refresh but the current attempt will fail if he do so.
7. The user has 60 seconds to submit the captcha for each attempt. After 60 seconds if no input is provided the page will refresh and the current attempt fails.
8. After 10 attempts the user is directed to evaluate visual captcha.

Visual CAPTCHA Evaluation

1. The user needs to select the image blocks matching the question in each attempt.
2. After selecting and clicking check, a message saying success/failure is displayed and the captcha will reload.
3. The user has 60 seconds to select the image blocks.
4. The user needs to select at least one block to validate the captcha.
5. If no input is provided in 60 seconds the captcha will reload and the attempt will be failed.
6. The captcha will reload on page refresh or on clicking refresh button.
7. Either of these actions will result in failing the current attempt.
8. After 10 attempts the user will be directed to evaluate collage captcha.

Object Collage CAPTCHA Evaluation

1. The user is provided with a set of 3 images to evaluate.
2. User needs to clearly observe the objects in the images which matches with the question.
3. Inputs for all the images are mandatory. There is no partial inputting.
4. User needs to provide the object count in all the 3 images to check.
5. The user has 60 seconds to provide input, without which captcha reloads failing the attempt.
6. On page reload or on hitting refresh the captcha will reload failing the current attempt.
7. After 10 instances a report is generated for the user in a tabular format showing the overview of results for all the 30 attempts.
8. The report also displays the success rate of chosen user.

ADDING ADDITIONAL CAPTCHA'S

Adding additional captcha sets to *Text Captcha*

1. The .jpg captcha image should be copied to the **images** folder which can be found inside **text_captcha** folder inside main **captcha** folder.
2. The database is designed to auto_increment the id value upon inserting a new captcha.
3. Run the following command after switching to captcha database
    - INSERT INTO image_ans (id,answer) values (NULL,'*answer_to_captcha*');
4. Change the $i variable in captcha/text_captcha/index.php (line 65)
5. Open captcha/captcha_evaluate/text_captcha_evaluate.php and make following changes
    - Change second value of randomnum() to corresponding new file name
6. Run captcha/text_captcha/index.php to affect the changes.

Adding additional captcha sets to *Visual Captcha*

1. The captcha grid is designed in such a way that it can fit 9 image squares with file names ranging from 1 to 9.
2. The folder names are setup in the ascending order (1,2,3,4,5,…..)

3. Add a new folder and rename to corresponding number (for example 17 if the images folder has 16 folders), copy the 9 '.jpg' image to the folder.
4. The database is designed to auto_increment the folder_name value upon inserting a new captcha.
5. Run the following command after switching to captcha database
    - INSERT INTO visual_ans (folder_name,question,answer) values (NULL,'question for captcha',*correct_image_file_names*);
6. Open captcha/visual_captcha/index.php and make following changes
    - Change the $i variable to corresponding value (line 29)
    - Update required conditions for else if statement (line 167)
7. Open captcha/captcha_evaluate/visual_captcha_evaluate.php and make following changes
    - Change second value of randomnum() to corresponding new folder name (line 52)
    - Update required conditions for else if statement (line 224)
8. Run captcha/visual_captcha/index.php and visual_captcha_evaluate.php to affect the changes.

Adding additional captcha sets to *Collage Captcha*

1. The captcha grid is designed in such a way that it can fit 3 image squares with file names ranging from 1 to 3.
2. The folder names are setup in the ascending order (1,2,3,4,5,…..)
3. Add a new folder and rename to corresponding number (for example 21 if the images folder has 20 folders), copy the 3 '.png image to the folder.
4. The database is designed to auto_increment the folder_name value upon inserting a new captcha.
5. Run the following command after switching to captcha database
    - INSERT INTO collage_ans (folder_name,question,answer) values (NULL,'question for captcha',*answer_to_3_images*);
6. Open captcha/collage_captcha/index.php and make following changes
    - Change the $i variable to corresponding value (line 47)
7. Open captcha/captcha_evaluate/collage_captcha_evaluate.php and make following changes
    - Change second value of randomnum() to corresponding new folder name (line 50)
8. Run captcha/collage_captcha/index.php and visual_captcha_evaluate.php to affect the changes.