International Journal of Scientific Research in Science, Engineering and Technology (www.ijsrset.com)
© 2020 IJSRSET | Volume 7 | Issue 3 | Print ISSN: 2395-1990 | Online ISSN : 2394-4099
DOI : https://doi.org/10.32628/IJSRSET20738

File Based System-Authentication in Scripting Language of Web Based Development

Danial Kafi Ahmad [1 2*], Mariam Farida Ahmad[3]

[1]Faculty of Information Technology, INTI International University, Nilai, Malaysia

[2]Language Centre, National Defence University of Malaysia, Kuala Lumpur, Malaysia.

[3]Department of Statistics Malaysia.

[*] Corresponding author: danialkafi.ahmad@newinti.edu.my, 3181121@alfateh@upnm.edu.my

ABSTRACT

Web Based system were used widely across all regions. This shows the importance of the system in performing important tasks by means of automation. However, the nature of the Web Based system where by its connected via Commercial Interconnected Network (Internet) had caused the system's data to exposed to possible harmful threats in within the network. Therefore, it is vital to deploy a mechanism that could reduce the possibility of attacks from the threats, and one of it is the system authentication. This had shown that the authentication concept is vital and a good understanding of it is necessary. This could be done by study variety of authentication mechanism regardless of their efficiency. In this research experiment, a light authentication mechanism of login module were developed and were integrated with the File based system of Hypertext Preprocessor (PHP) and Hypertext Markup Language (HTML) of scripting language and does not involved any Database system in the authentication.

Keywords : Authentication, Software Development, Web-Based.

I. INTRODUCTION

Authentication were vital as a security mechanism in any system which available today. For example we could see the usage of login authentication for user to access the system in many E-Commerce websites and other inventory system [1]. This also applied to stand alone system which perform an automation task. However the usage of the login authentication mostly involved the deployment of Database System such as MySQL, Oracle etc. As authentication module is as important as the growth of the web based system, then it's vital to come up with more variety designs of login authentication module in web based that implement different platform and tools. Regardless of their efficiency of different design, it is always important to get findings on any problems or issues which related to user authentication of Web Based, so that it can be tested or manipulate for future used. In other words, it's a good idea to possess more knowledge in variety of aspects of user authentication in order to come out with a better design. In this research experiment, a user authentication module integrated with the built in FILE function of Hypertext preprocessor (PHP) and Hypertext Markup Language (HTML) were developed without involving the usage of Database system such as MySQL etc.

II. Methodology

SDLC

Software Development Lifecycle (SDLC) which consist of System Planning and Selection, Analysis, Design, and Implementation with Maintenance phases were deployed in order to allow the activities of getting the system requirement, classifying the system functionalities, writing codes or scripts, installation and configuration, as well as testing and maintenance. However, the phases in SDLC were executed in a Non-Plan Driven approach as to gain more flexibilities during development. In addition, Non-Plan Driven approach seems to be more suitable for small scale project [2]. Therefore incremental model of agile technique were chosen as the process model for the software development of this research. The model comprises of Specification, Development and Validation in which the phases allowed the activities of getting the requirement, designing, coding, installation and testing respectively [3]. The activities were executed dynamically as according to the condition of requirement during the development as due to the nature of Non-Plan Driven.

Software Process Model (Incremental)

Specification

Development Tools

In this research experiment, PHP and HTML were chose as the language to develop the program. This is due to the nature of the scripting languages which are suitable for Web-Based system [4]. In addition, the languages were classified as an open source which mean they are easily accessible by developers for any relevance purposes. Apart from possess good functionality, PHP and HTML had been proven as a language that were able to deliver Web-Based system in commercial industries as well as in education.

Architecture-Client Server

The architecture of Client-Server were implemented in the system design. The architecture involved the exchange of request and response in between the client (Web Browser) and the Server (XAMPP). This means that the code were developed and stored on the client side and only being executed when requested by the client, and in this experiment the exchange occurred in within the same machine.

Pseudocode

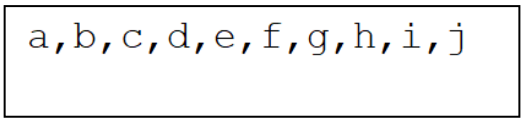
Structure and program of the authentication model were depict in a pseudocode [5]. The pseudocode had been practiced widely in any software development process. The benefits of developing pseudocode is the developer could have better understanding and picture of the system flow or semantic prior to the coding phase. However, pseudocode is not meant to be executed in automation mode as compared to the program or software code. This reflects the situation whereby its rather on general idea and does not imply the syntax rules in specific. [6] In testing, the pseudocode were treated as a tool to support for static testing. Figure 1 shows the pseudocode of the developed system (service page).

```
BEGIN
            Retrieve user input
                                    Open the file using fopen()
            Read the content of the file  using fread()
            Change the content of the file (string type) to (array type)
                                                using explode()
                        foreach (elements in the array)
            if elements is equal to input from user
                        then go to success page
            end foreach
            if (data from user is not null)
              then Display data is not match
            else
                                                Display data is empty
END
```

Figure 1. Pseudocode of service page.

Development Code

The File Based Login Module comprises of variables, array functions, looping control structure, conditional control structure and file functions. These could be seen through the usage of $user, $_POST["inputfromuser"], explode() which hold values of array after conversion from string, foreach() to iterate elements in within the array, and if() else() to check for the value from user. Figure 2 shows the textfile (username.txt) which consists of the string of (a,b,c,d,ef,g,h,I,j). Further explanation of the code could be read in Discussion chapter.

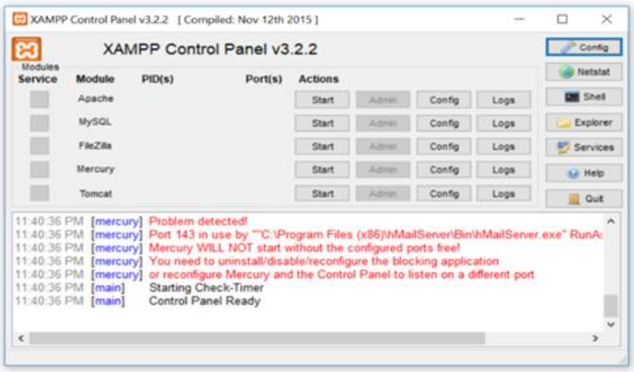


```
a,b,c,d,e,f,g,h,i,j
```

Figure 2. Text file of username.txt.

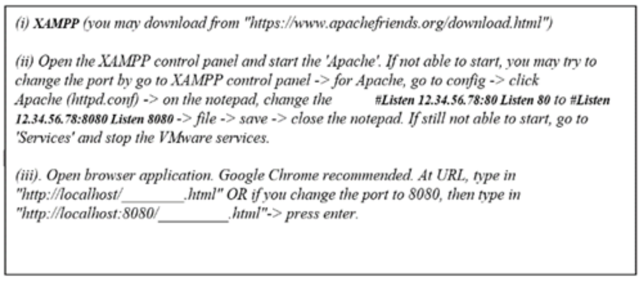
Installation and Configuration

Figure 3, figure 4, and figure 5 show the Graphical User Interface (GUI) of the XAMPP Control Panel, the configuration steps and the **httpd.conf** file. In this research, the port of 80 had been configured by change it to 8080 in order to allow the Apache software to start and run.



```
11:40:36 PM  [mercury]  Problem detected!
11:40:36 PM  [mercury]  Port 143 in use by ""C:\Program Files (x86)\hMailServer\Bin\hMailServer.exe" RunA
11:40:36 PM  [mercury]  Mercury WILL NOT start without the configured ports free!
11:40:36 PM  [mercury]  You need to uninstall/disable/reconfigure the blocking application
11:40:36 PM  [mercury]  or reconfigure Mercury and the Control Panel to listen on a different port
11:40:36 PM  [main]     Starting Check-Timer
11:40:36 PM  [main]     Control Panel Ready
```

Figure 3. Control Panel of XAMPP Package.

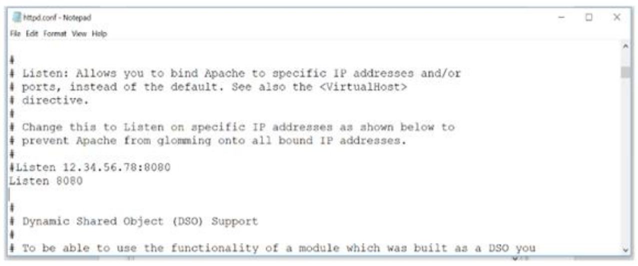


(i) ***XAMPP*** *(you may download from "https://www.apachefriends.org/download.html")*

(ii) Open the XAMPP control panel and start the 'Apache'. If not able to start, you may try to change the port by go to XAMPP control panel -> for Apache, go to config -> click Apache (httpd.conf) -> on the notepad, change the ***#Listen 12.34.56.78:80 Listen 80*** *to* ***#Listen 12.34.56.78:8080 Listen 8080*** *-> file -> save -> close the notepad. If still not able to start, go to 'Services' and stop the VMware services.*

(iii). Open browser application. Google Chrome recommended. At URL, type in "http://localhost/________.html" OR if you change the port to 8080, then type in "http://localhost:8080/________.html"-> press enter.

Figure 4. Configuration steps of port via *httpd.conf* file.



```
#
# Listen: Allows you to bind Apache to specific IP addresses and/or
# ports, instead of the default. See also the <VirtualHost>
# directive.
#
# Change this to Listen on specific IP addresses as shown below to
# prevent Apache from glomming onto all bound IP addresses.
#
#Listen 12.34.56.78:8080
Listen 8080


#
# Dynamic Shared Object (DSO) Support
#
# To be able to use the functionality of a module which was built as a DSO you
```

Figure 5. Configuration of port via httpd.conf script.

Validation

Static and Dynamic type techniques in testing were executed towards the login module. Both techniques were required in order to complete the testing process, whilst static were required in detecting any potential faults or error, and Dynamic were required to execute by means of automation in order to test or debug the program. Static techniques were executed via pseudocode as a tool in order to detect any potential defects in terms of the system or program flow. Rex Black stated that the static technique should be done before the dynamic technique [7].

Test Level

The four levels of testing which are Unit Testing, Integration Testing, System Testing and Acceptance Testing were executed. The following logical model which comprised of *Set {}* were used in visualizing the testing process of each levels. As according to the Susanna S. Epp, the elements or unit can be represented by using the *Set {}* [8].

Set U: {elements}

Let $x \in U$, component(x) -> result(y)

Figure 6. Unit Testing Model. [6]

Set U: {elements}

Let $x_b \in U$, n=<b<=n, n=<z<=n,

component(x_b) executed $\cap$ component(x_b) executed -> result(y_z)

Figure 7. Integration Testing Model. [6]

Set U: {elements}

Let $x_b \in U$, n=<b<=n, n=<y<=n

$$\frac{(component(xb = 1)\ \cap\ component(xb = n))}{((result(y1 = executable)) \cap \text{ result(yn = executable))}}$$

Figure 8. System Testing Model. [6]

Test Type and Technique

In order to achieve the Verification & Validation as opposed to the standard Software Development Process, both Static Technique and the Dynamic Technique, were deployed whereby the program were analyzed with and without automated execution. In addition both techniques were required in the process of testing [9].

III. DISCUSSION

Algorithm Coding

Figure 9 shows the scripts of HTML type for the first page of the program developed (Main.php). HTML tag were used as to contain the HTML scripts tags which are <form> and <input>. The action= "service.php" were used to redirect the input key in by the user to the service.php page by using method of post. POST method transmitted the data to the service.php page via the body of the program instead of Uniform Resource Locator (URL) of GET method. Some of the developers claim that POST is more secure compared to GET as it does not disclose the data transmitted via the URL [10]. Both input type of text and submit were deployed to accept input from user and send the data to service.php via POST respectively. Placeholder were used to indicate some instruction in within the user input text field.

```
<html>
<form action="service.php" method="POST">
<input type="text" name="inputfromuser" placeholder="Key in
your input">
<input type="submit" value="access">
</form>
</html>
```

Figure 9. main.php

Figure 10 shows the code or scripts of PHP type that perform validation of data and page redirects. The following program (service.php) comprise of foreach(explode(",",fgets(fopen("username.txt","r"))) as $user) which peform an iteration of the data from the text file (username.txt) which is stored in the htdocs folder in within XAMPP. The loop consist an argument which could be reflect by this line ($array_var as $string_var), and in this research the assumption is $array_var represent explode(",",fgets(fopen("username.txt","r"))) whilst $string_var represent $user. Both of the so called "sub argument" were holding a value of an array type with elements of (a,b,c,d,e,f,g,h,I,j) from the username.txt file and value of (a,b,c,d,e,f,g,h,I,j) of string type from the same file respectively. As for explode(",",fgets(fopen("username.txt","r"))) which represent by $array_var, the explode(); is responsible to convert the string retrieved from the username.txt to an array type. However, the conversion may happen with the present of the separator, as the script executor were required to know the breakpoint of the string in (username.txt) and create an array with respective elements. In this situation, the separator is the comma (,), whereby the array of elements were not crated with no separator included in the explode() as the program would not know which are the elements. The fgets(), fopen() were used in this line fgets(fopen("username.txt","r"))in order to retrieve the whole data in username.txt and retrieving the file (username.txt) from the the htdocs folder respectively. However the the username.txt were retrieved before its content (a,b,c,d,e,f,g,h,I,j) were retrieved. File mode of "r" which mean read were used in order to read the file's content.

```
<?php
foreach(explode(",",fgets(fopen("username.txt","r"))) as
$user){
if($user==$_POST["inputfromuser"]){header("location:success.ph
p");}
}
if($_POST["inputfromuser"]!=""){
echo "input not match with data in file..Loading...";}
else{echo "No input given...Loading...";}
header("refresh:2;url=main.php");
?>
```

Figure 10. service.php

The assignment of an each of array value (a,b,c,d,e,f,g,h,I,j) into $user occur until all the elements were being assigned. However, the conditional structure of if(){} else{} in within the foreachloop were executed to check if the value keyed in by the user ($_POST[“input from user”]) are equal or match to any value inside username.txt. The program would be redirected to success.php if any of the condition in within the loop iteration is TRUE. Figure 11 shows the program of success.php which display the string inidicated the page site and the page remains for 2 seconds before redirecting to main.php. In the case of no TRUE condition at all in within the iteration, therefore the the if(){} else{} outside foreach loop structure would check for TRUE condition of NULL value of $_POST[“inputfromuser”] and would display “Input not macth with data in file…Loading…” for FALSE NULL value and display “No input given…Loading…” for TRUE NULL value.

```
<?php
echo "This is success page...Loading...";
header("refresh:2;url=main.php");
?>
```

Figure 11. success.php

Graphical User Interface

Figures below shows the output of the developed module. Main Page consist of textbox and submit button created by the HTML script. Figure 13, figure 14 and figure 15 shows the string as an output of the different TRUE FALSE condition respectively.

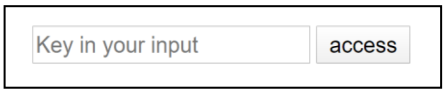


Figure 12. Main page GUI.

This is success page...Loading...

Figure13. Success page GUI.

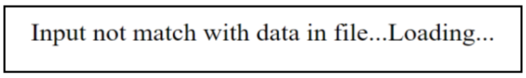

Input not match with data in file...Loading...

Figure14. No input not match GUI.

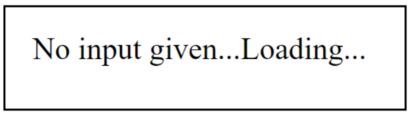

No input given...Loading...

Figure15. No input given GUI.

Testing (Validation)

The test levels of Unit Testing, Integration Testing, System and Acceptance Testing were deployed on the program as show in figure 16, and figure 17. The execution and validation of Integration and System Testing led to the acceptance of the system. Each of the page (**main.php, service.php, success.php**) were treated as a three different components.

Unit Testing (Authentication module)

Set U: {<html><form action="service.php" method="POST">
<input type="text" name="inputfromuser" placeholder="Key in your input">
<input type="submit" value="access"></form></html>,
<?php
foreach(explode(",",fgets(fopen("username.txt","r"))) as $user){
if($user==$_POST["inputfromuser"]){header("location :success.php");}
}
if($_POST["inputfromuser"]!=""){
echo "input not match with data in file..Loading...";}
else{echo "No input given...Loading...";}
header("refresh:2;url=main.php");
?>,
<?php echo "This is success page...Loading...";
header("refresh:2;url=main.php"); ?>}

Figure 16. Components of the login authentication module.

Let x ∈ U, component(x) -> result(y)
component(x= <html><form action="service.php" method="POST">

```
<input type="text" name="inputfromuser"
placeholder="Key in your input">
<input type="submit"
value="access"></form></html>)->
result(y=figure12)
```

Figure 17. Tested component of main.php.

```
component(x= <?php
foreach(explode(",",fgets(fopen("username.txt","r")))
as $user){
if($user==$_POST["inputfromuser"]){header("location
:success.php");}
}
if($_POST["inputfromuser"]!=""){
echo "input not match with data in file..Loading...";}
else{echo "No input given...Loading...";}
header("refresh:2;url=main.php");?> ->
($user==$_POST["inputfromuser"])IS TRUE)  ->
result(y= figure13)
```

Figure 18. Tested component of service.php (TRUE).

```
component(x= <?php
foreach(explode(",",fgets(fopen("username.txt","r")))
as $user){
if($user==$_POST["inputfromuser"]){header("location
:success.php");}
}
if($_POST["inputfromuser"]!=""){
echo "input not match with data in file..Loading...";}
else{echo "No input given...Loading...";}
header("refresh:2;url=main.php");?> ->
($user==$_POST["inputfromuser"])IS FALSE AND
($_POST["inputfromuser"]!="") IS TRUE )  ->
result(y= figure14)
```

Figure 19. Tested component of service.php (FALSE->TRUE).

```
component(x= <?php
foreach(explode(",",fgets(fopen("username.txt","r")))
as $user){
if($user==$_POST["inputfromuser"]){header("location
:success.php");}
}
if($_POST["inputfromuser"]!=""){
echo "input not match with data in file..Loading...";}
else{echo "No input given...Loading...";}
header("refresh:2;url=main.php");?> ->
($user==$_POST["inputfromuser"])IS FALSE AND
($_POST["inputfromuser"]!="") IS FALSE )  ->
result(y= figure15)
```

Figure 20. Tested component of service.php (FALSE->FALSE).

```
component(x= <?php echo "This is success
page...Loading...";
header("refresh:2;url=main.php"); ?>)  ->  result(y=
figure15 -> figure 13)
```

Figure 21. Tested component of success.php.

Integration and System Testing

<u>Condition: TRUE:</u>

```
Set U: {<html><form action="service.php"
method="POST">
<input type="text" name="inputfromuser"
placeholder="Key in your input">
<input type="submit" value="access"></form></html>,
<?php
foreach(explode(",",fgets(fopen("username.txt","r")))
as $user){
if($user==$_POST["inputfromuser"]){header("location
:success.php");}
}
if($_POST["inputfromuser"]!=""){
echo "input not match with data in file..Loading...";}
else{echo "No input given...Loading...";}
header("refresh:2;url=main.php");
?>,
<?php echo "This is success page...Loading...";
header("refresh:2;url=main.php"); ?>}
```

Figure 22. Components of the login authentication module for Integration and System Testing (TRUE).

Let $x_b \in U$

component(x_b) executed ∩ component(x_b) executed ->

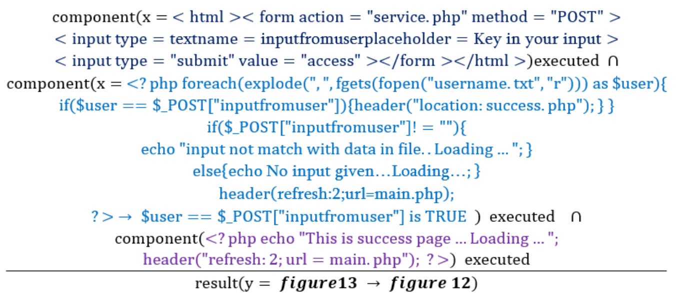
result(y_z)

component(x= <html><form action="service.php"
method="POST">
<input type="text" name="inputfromuser"
placeholder="Key in your input">
<input type="submit"
value="access"></form></html>)executed ∩

component(x= <?php
foreach(explode(",",fgets(fopen("username.txt","r")))
as $user){
if($user==$_POST["inputfromuser"]){header("location
:success.php");} }
if($_POST["inputfromuser"]!=""){
echo "input not match with data in file..Loading...";}
else{echo "No input given...Loading...";}
header("refresh:2;url=main.php");
?> -> $user==$_POST["inputfromuser"] is TRUE)
executed ∩
component(<?php echo "This is success
page...Loading...";
header("refresh:2;url=main.php"); ?>) executed

➔ result(y= ***figure13 -> figure 12***)

Figure 23. Integration Testing model with component executed (TRUE).



component(x = < html >< form action = "service. php" method = "POST" >
< input type = textname = inputfromuserplaceholder = Key in your input >
< input type = "submit" value = "access" ></form ></html >)executed ∩
component(x = <? php foreach(explode(", ", fgets(fopen("username. txt", "r"))) as $user){
if($user == $_POST["inputfromuser"]){header("location: success. php"); } }
if($_POST["inputfromuser"]! = ""){
echo "input not match with data in file. . Loading ... "; }
else{echo No input given…Loading…; }
header(refresh:2;url=main.php);
? > → $user == $_POST["inputfromuser"] is TRUE) executed ∩
component(<? php echo "This is success page ... Loading ... ";
header("refresh: 2; url = main. php"); ? >) executed

result(y = ***figure13*** → ***figure*** **12**)

Figure 24. System Testing model with component as a system executed (TRUE).

Condition: FALSE->TRUE:

Set U: {<html><form action="service.php"
method="POST">
<input type="text" name="inputfromuser"
placeholder="Key in your input">
<input type="submit" value="access"></form></html>,
<?php
foreach(explode(",",fgets(fopen("username.txt","r")))
as $user){
if($user==$_POST["inputfromuser"]){header("location
:success.php");}
}
if($_POST["inputfromuser"]!=""){
echo "input not match with data in file..Loading...";}
else{echo "No input given...Loading...";}
header("refresh:2;url=main.php");
?>,
<?php echo "This is success page...Loading...";
header("refresh:2;url=main.php"); ?>}

Figure 25. Components of the login authentication module for Integration and System Testing (FALSE->TRUE).

Let $x_b \in U$

component(x_b) executed ∩ component(x_b) executed ->

result(y_z)

component(x= <html><form action="service.php"
method="POST">
<input type="text" name="inputfromuser"
placeholder="Key in your input">
<input type="submit"
value="access"></form></html>)executed ∩

component(x= <?php
foreach(explode(",",fgets(fopen("username.txt","r")))
as $user){
if($user==$_POST["inputfromuser"]){header("location
:success.php");} }
if($_POST["inputfromuser"]!=""){
echo "input not match with data in file..Loading...";}
else{echo "No input given...Loading...";}

header("refresh:2;url=main.php");
?> -> $user==$_POST["inputfromuser"] is FALSE
AND ($_POST["inputfromuser"]!="") IS TRUE) ->
result(y= ***figure15 -> figure 13***)

Figure 26. Integration Testing model with component executed (FALSE->TRUE).

component(x = < html >< form action = "service. php" method = "POST" >
< input type = textname = inputfromuserplaceholder = Key in your input >
< input type = "submit" value = "access" ></form ></html >)executed ∩
component(x = <? php foreach(explode(", ", fgets(fopen("username. txt", "r"))) as $user){
if($user == $_POST["inputfromuser"]){header("location: success. php"); } }
if($_POST["inputfromuser"]! = ""){
echo "input not match with data in file. . Loading ... "; }
else{echo No input given…Loading…; }
header(refresh:2;url=main.php);
? > → $user == $\$_{\text{POST[inputfromuser]is}}$FALSE AND
($_POST["inputfromuser"]! = "") IS TRUE)

result(y = ***figure*15** → ***figure* 13**)

Figure 27. System Testing model with component executed (FALSE->TRUE).

Condition: FALSE->FALSE:

Set U: {<html><form action="service.php" method="POST">
<input type="text" name="inputfromuser" placeholder="Key in your input">
<input type="submit" value="access"></form></html>**,**
<?php
foreach(explode(",",fgets(fopen("username.txt","r"))) as $user){
if($user==$_POST["inputfromuser"]){header("location :success.php");}
}
if($_POST["inputfromuser"]!=""){
echo "input not match with data in file..Loading...";}
else{echo "No input given...Loading...";}
header("refresh:2;url=main.php");
?>**,**
<?php echo "This is success page...Loading...";
header("refresh:2;url=main.php"); ?>**}**

Figure 28. Components of the login authentication module for Integration and System Testing (FALSE->FALSE).

Let $x_b \in U$
component(x_b) executed ∩ component(x_b) executed -> result(y_z)

component(x= <html><form action="service.php" method="POST">
<input type="text" name="inputfromuser" placeholder="Key in your input">
<input type="submit" value="access"></form></html>)executed ∩

component(x= <?php
foreach(explode(",",fgets(fopen("username.txt","r"))) as $user){
if($user==$_POST["inputfromuser"]){header("location :success.php");} }
if($_POST["inputfromuser"]!=""){
echo "input not match with data in file..Loading...";}
else{echo "No input given...Loading...";}
header("refresh:2;url=main.php");
?> -> $user==$_POST["inputfromuser"] is FALSE
AND ($_POST["inputfromuser"]!="") IS FALSE) ->
result(y= ***figure15 -> figure 13***)

Figure 29. Integration Testing model with component executed (FALSE->FALSE).

component(x = < html >< form action = "service. php" method = "POST" >
< input type = textname = inputfromuserplaceholder = Key in your input >
< input type = "submit" value = "access" ></form ></html >)executed ∩
component(x = <? php foreach(explode(", ", fgets(fopen("username. txt", "r"))) as $user){
if($user == $_POST["inputfromuser"]){header("location: success. php"); } }
if($_POST["inputfromuser"]! = ""){
echo "input not match with data in file. . Loading ... "; }
else{echo No input given…Loading…; }
header(refresh:2;url=main.php);
? > → $user == $\$_{\text{POST[inputfromuser]is}}$FALSE AND
($_POST["inputfromuser"]! = "") IS FALSE)

result(y = ***figure*15** → ***figure* 13**)

Figure 30. System Testing model with component executed (FALSE->FALSE).

IV. CONCLUSION AND RECOMMENDATION

Authentication as a security mechanism to protect data were very important in any software system especially the Web Based due to its nature of massive data and information exchange among web system within a network. This is due to the concept of authentication which allow only authorized user to access a system. Therefore, in this research, the authentication module integrated with the file based of PHP scripting language were develop in order to create variety of authentication module design that could be improvised and enhanced for future used. The developed module also could be used in education for teaching and learning as it provide the important concept of programming which can be delivered to the students. In conclusion, since the login module authentication are vital, therefore more research and development of the module are suggested to be promoted and developed rapidly.

V. ACKNOWLEDGEMENT

The main author would like to express his gratitude to the Office of Research Development & Consultancy of INTI International University and Faculty of Information Technology of INTI IU for the supports in terms of resources allocation for the research, and also to Assoc. Prof. Dr. Norshima Zainal Shah, late Lt. Kol. Dr. Muhammad Salim bin Muhammad Tufail (B) and Dr. Hoo Yann Seong from the National Defence University of Malaysia for the continuous support to the author in doing research. The authors would like to express their gratitude to Dr. Ahmad Selamat (P.I.S) for the advice on mathematics concept and their modelling logic.

VI. REFERENCES

[1]. Balaji, N.P., Sreenivasulu, U. and Reddy, C.V., 2011. Web-Based System—Authentication to Single Log-on to Several Applications. International Journal of Computer Science and Telecommunications, Journal.

[2]. Bass, J.M., 2019. Agile on a large scale. ITNOW, 61(1), pp.56-57.

[3]. I. Sommerville, "Software Engineering 9th Edition," Pearson, 2010.

[4]. Mishra, A., 2014. Critical Comparison of PHP and ASP .NET for Web Development. International Journal of Scientific & Technology Research, 3(7), pp.331-333.

[5]. Scanlan, D.A., 1989. Structured flowcharts outperform pseudocode: An experimental comparison. IEEE software, 6(5), pp.28-36.

[6]. Ahmad, D.K., 2020. Simplification of Arithmetic and Variable Script in Hypertext Preprocessor (PHP, International Journal of Computer Science and Mobile Computing, 9(5), pp21-34.

[7]. R. Black, D. Graham, E. V. Veenendar, and I. Evans, "Foundations of Software Testing, ISTQB Certification," Cengage Learning EMEA, third edition, pp 35-121, 2012.

[8]. S.S. Epps, "Discrete mathematics with applications," Cengage learning, pp. 76-264, 2010.

[9]. N. Honest, "Role of Testing in Software Development Life Cycle", International Journal of Computer Sciences and Engineering, Vol.7, Issue.5, pp. 886-889, 2019.

[10]. Shiflett, C., 2004. PHP security. ApacheCon.(Las Vegas, USA, 2004) http://shiflett. org/php-security. pdf [referred 25.05. 2011].

Cite this article as :

Danial Kafi Ahmad, Mariam Farida Ahmad., " File Based System-Authentication in Scripting Language of Web Based Development, International Journal of Scientific Research in Science, Engineering and Technology(IJSRSET), Print ISSN : 2395-1990, Online ISSN : 2394-4099, Volume 7, Issue 3, pp.39-47, May-June-2020. Available at doi : https://doi.org/10.32628/IJSRSET20738
Journal URL : http://ijsrset.com/IJSRSET20738