Addis Ababa Institute of Technology

Topic: Software Testing Plan Document

Submitted Date
Jul 2017

Instructor: Natinael

Group Members:

1. Fruit Teklu	**- ATR/7656/07**
2. Kidus Mamuye	**- ATR/6157/07**
3. Silas Getachew	**- ATR/8744/07**
4. Amante Diriba	**- ATR/3786/07**

Table of Contents

Table of Table.

Table of Figure

Internationally Left Blank

1. Introduction

This document will address the different standards that will apply to the unit, integration and system testing of the Web Application Ethio Virtual Academic Portal (EVAP). Throughout the testing process we will be applying the test documentation specifications described in the IEEE Standard for Software Test Documentation.

1.1 Overview

In this document, we have explicitly articulated the different testing strategies for our application in Detail. First section deal with Introduction, purpose and objective of the document. Second part Provide the Features to be tested or not from our application. The third section contain the main part of this document's which is the different testing approaches unit testing and Integration testing.

1.2 Purpose

The purpose of our test plan is to uncover and report as many bugs as possible to increase our system reliability and to ensure the right product is developed.

1.3 Objective of the Test

The objective of the test is to verify that the functionality Ethio Virtual Academic Web application works according to the specifications. The test will execute and verify the test scripts, identify, fix and retest all high and medium severity defects per the entrance criteria, prioritize lower severity defects for future fixing by defining various Testing strategies and testing tools used for complete Testing life cycle of this project.

2. Feature to be tested and not to be tested

2.1 Feature to be tested

The following are the list of area that is be focused on while testing our application:

Functional Requirements Testing includes testing:

- Creating user account
- Authentication of user account (during the login)
- Account form validation which includes testing how the system gives error feedback when user fills an inappropriate user name and e-mail address format.
- Log in account validation includes testing how the system gives error feedback when user fills in invalid user name and password.
- Log out section
- Sharing of resources
- Discussion panel section
- Assignment submission portal
- Searching for post
- Asking for help section

Non Functional Requirements Testing includes testing:

- Proper display of user interface during navigation across the pages.
- Hardware and software that the system will run on.
- Performance.
- Usability – includes testing whether the system's codes are reusable to the extent that they are mentioned to fit (to be).
- Reliability – includes testing how the system is fault tolerant and provides some suggestion for the frequently fault done by users while interacting with the system.

2.2 Feature **not** to be tested

Possible features and functionalities that are not to be tested; the reasons for why they are not going to tested and their possible contingencies are explicitly articulated as follows:

- Web site becomes unavailable - Testing will be delayed until this situation is rectified.
- Web testing software is not available/does not work (e.g. Web site uses cookies and tool cannot handle cookies) - This testing will require more manual and functionalities that are not specified to be handled in this version of the product.
- Testing staff unavailability - This testing will be delayed until the system is launched.
- Performance of the system –This testing will require more time and will not be tested if time does not permit.
- Security – some aspect of security issues such as user authentication and assuring that the system can be secure from cross site scripting trait will be tested as it has been considered as a functional requirement for our system. But other security issues beyond this will not be tested due to time limits. But this testing will be handled for the next version of this application.

3. Test Requirements

3.1 Facilities Required

We have used a remote web server facility to simulate a real production web server, hosting our web application at http://evapp.azurewebsites.net/ URL. This has assisted us to do a basic test to prove that the system can be installed, configured, and brought to an operational state.

3.2 Hardware Required

To run local testes we have used ordinary computers having Intel processors, and for the remote test the application was hosted on Intel-based shared server.

3.3 Software Required

Locally, the Application was configured to run on XAMPP Server to run the test locally. Remotely, it was configured to run on IIS and MySQL Servers. To depict some tests seeking diagrammatic representation such as Control Flow Testing, we have used "Edraw" to draw diagrams. To undertake system testing we have used "Selenium Web Automation Framework", driven by C#. Since the server-side part of our web application is written using PHP, we have used "PHPUnit Framework" to run unit tests.

4. Testing Approaches

4.1 Overview

In this section we mention different approaches and strategy of testing, Control flow, dataflow, domain testing used to generate test data, which given to unit testing in implementation of unit testing. Also clearly describe process of test case generation with its steps.

4.2 Unit Testing

Unit testing of software applications is done during the development (coding) of an application. The objective of unit testing is to isolate a section of code and verify its correctness. In procedural programming, a unit may be an individual function or procedure. The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. Unit testing is usually performed by the developer.

4.2.1 Test Data Generation

4.2.1.1 Control flow Testing

a. Registration

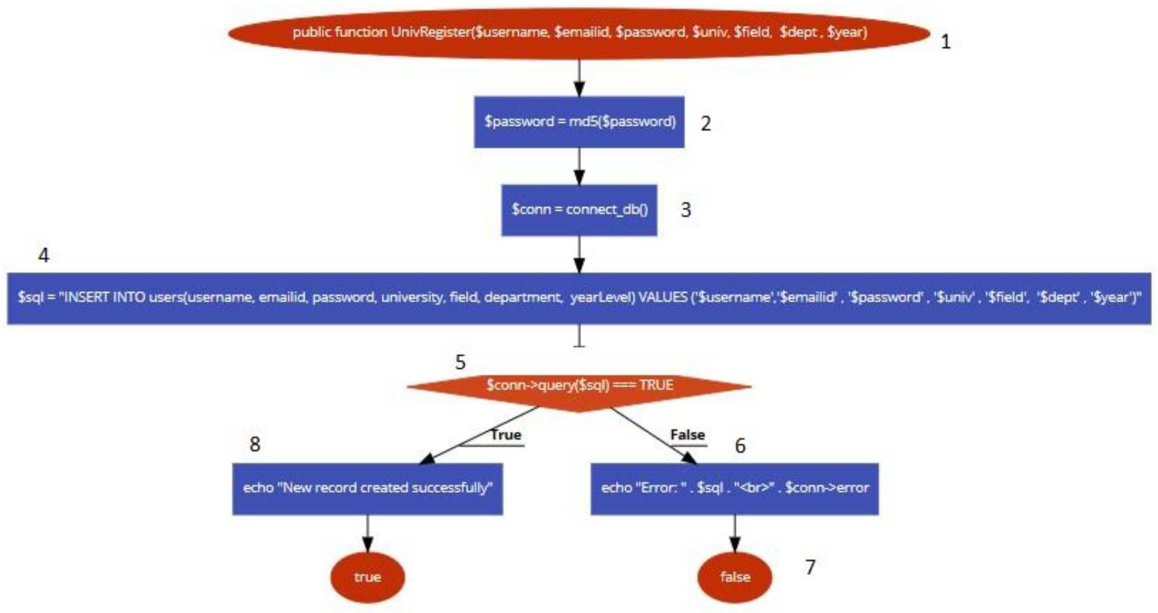

public function UnivRegister($username, $emailid, $password, $univ, $field, $dept , $year)
1
$password = md5($password)
2
$conn = connect_db()
3
4
$sql = "INSERT INTO users(username, emailid, password, university, field, department, yearLevel) VALUES ('$username','$emailid' , '$password' , '$univ' , '$field', '$dept' , '$year')"
5
$conn->query($sql) === TRUE
True
False
8
6
echo "New record created successfully"
echo "Error: " . $sql . "<br>" . $conn->error
true
false
7

FIGURE 1: CFD OF REGISTRATION

Path selection

a. All path selection
   1. 1-2-3-4-5[F]-6-7

2. 1-2-3-4-5[T]-8-9
   Input are <username,email,password,univ,field,dept,year>

TABLE 1: REGISTRATION INPUT GENERATION

Input	Path
<null,null,null,null,null,null,null>	1-2-3-4-5[F]-6-7
<Yes,yes,yes,yes,yes,yes>	1-2-3-4-5[F]-8-9

Generating Test case

Input vector=username,email,password,univ,field,dept,year
Predicate = $conn->query(sql)
Path predicate = node 5.

Path predicate expression

I. For 1-2-3-4-5[F]-6-7
   a. $conn->query(sql)==True====False.

II. For 1-2-3-4-5[T]-8-9
   a. $conn->query(sql)==True====True.

So, we can choice test case between this path.

b. Upload resource

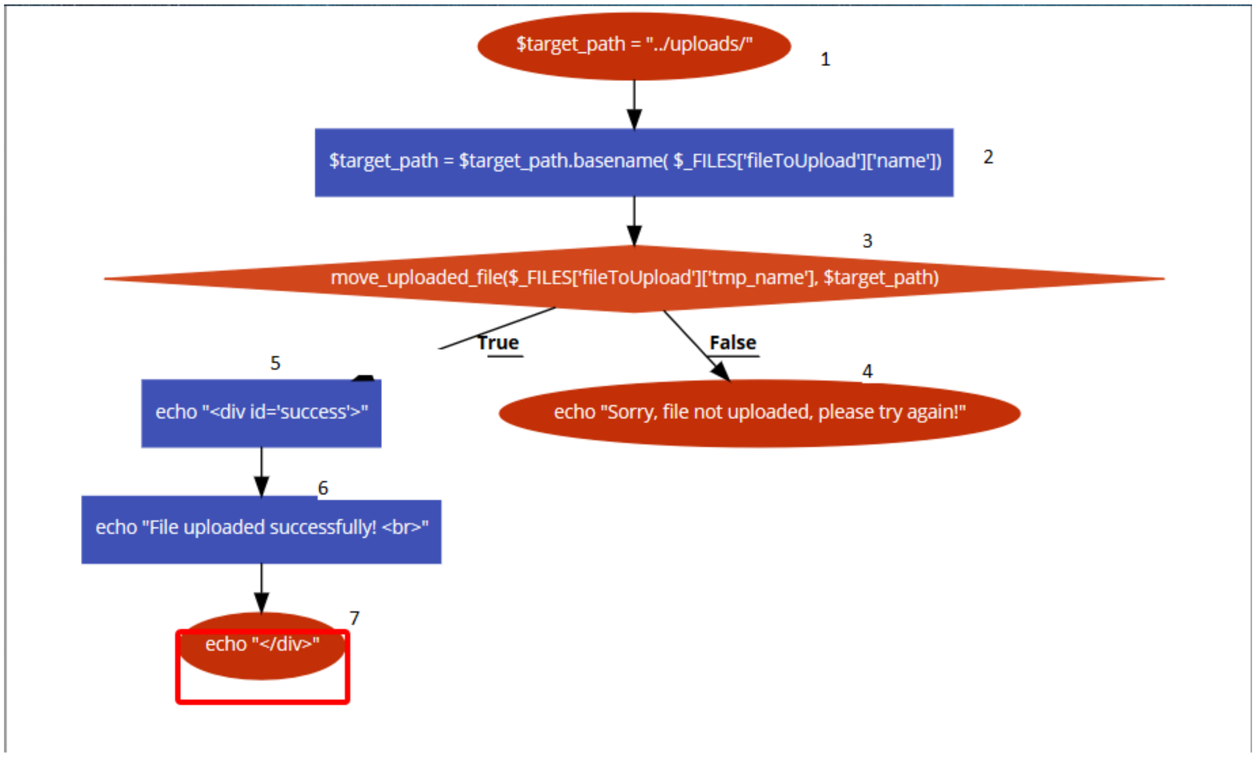

$target_path = "../uploads/"
1
$target_path = $target_path.basename($_FILES['fileToUpload']['name'])
2
3
move_uploaded_file($_FILES['fileToUpload']['tmp_name'], $target_path)
True
False
5
4
echo "<div id='success'>"
echo "Sorry, file not uploaded, please try again!"
6
echo "File uploaded successfully!
"
7
echo "</div>"

FIGURE 2:CFD OF UPLOAD RESOURCE UNIT

Test case Generation step.

Path selection

b. All path selection
   1. 1-2-3[T]-5-6-7
   2. 1-2-3[F]-4

Generating Test case

Input are <filename,path>

TABLE 2:UPLOAD RESOURCE INPUT GENERATION

Input	Path
<null,null>	1-2-3[F]-4
<Yes,yes>	1-2-3[T]-5-6-7

Input vector=filename, path

Predicate = move_upload_file('filename',path)

Path predicate = node 3.

Path predicate expression

I. For 1-2-3[F]-4
   a. move_upload_file('filename',path)==False

II. For 1-2-3[T]-5-6-7
   a. move_upload_file('filename',path)==True.

So we can choice test case between this paths.

c. creating database

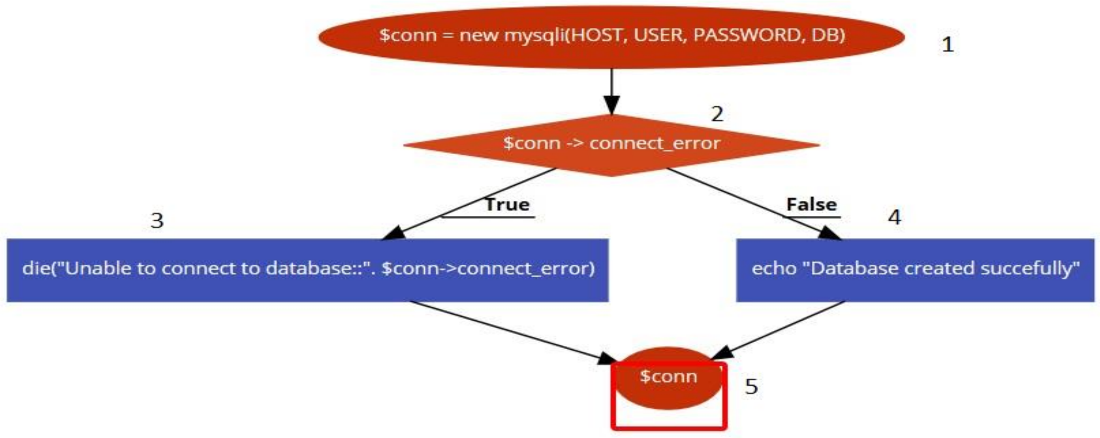

$conn = new mysqli(HOST, USER, PASSWORD, DB)
1
2
$conn -> connect_error
True
False
3
4
die("Unable to connect to database::". $conn->connect_error)
echo "Database created succefully"
$conn
5

FIGURE 3:CFD OF CREATING DATABASE UNIT

Path selection

All path selection

I. 1-2[T]-3-5
II. 1-2[F]-4-5

Input are <Host,User,password,database>

TABLE 3:CREATING DATABASE INPUT GENATION

Input	Path
Correct input<Yes,yes,yes,yes>	1-2[T]-3-5
Incorrect input<Yes,no,no,no> <No,no,no,no> <null,null,null,null> <No,Yes,no,Yes>	1-2[F]-4-5

d. Display answer (in discussion panel)

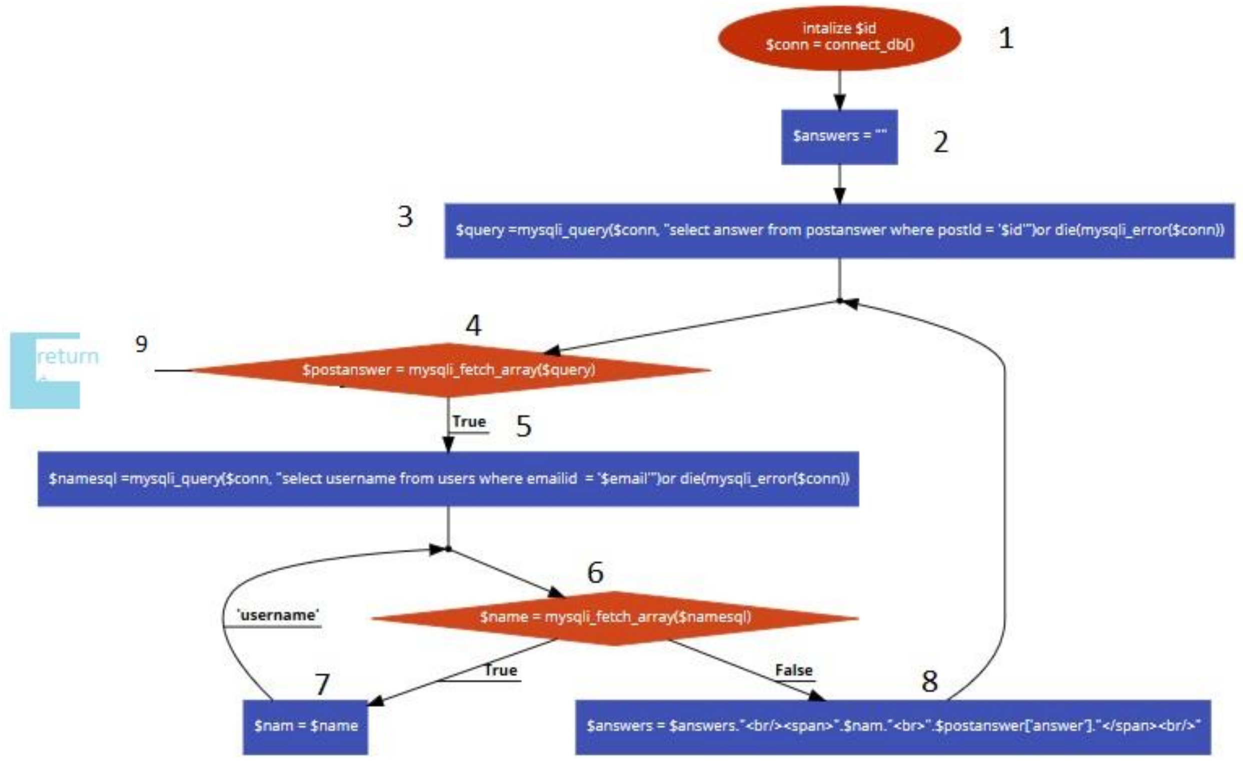

intalize $id
$conn = connect_db()
1
$answers = ""
2
3
$query =mysqli_query($conn, "select answer from postanswer where postId = '$id'")or die(mysqli_error($conn))
4
return
9
$postanswer = mysqli_fetch_array($query)
True
5
$namesql =mysqli_query($conn, "select username from users where emailid = '$email'")or die(mysqli_error($conn))
6
'username'
$name = mysqli_fetch_array($namesql)
True
False
7
8
$nam = $name
$answers = $answers."<br/><span>".$nam."<br>".$postanswer['answer']."</span><br/>"

FIGURE 4: DISPLAY ANSWER IN DISCUSSION PANEL UNIT

Path selection criteria

All path selection

a. 1-2-3-4[T]-5-6[T]-7-6[F]-8-4[F]-9
b. 1-2-3-4[F]-9
c. 1-2-3-4[T]-5-6[F]-8-4[T]-5-6[T]-7-6[F]-8-4[F]-9

e. Registration

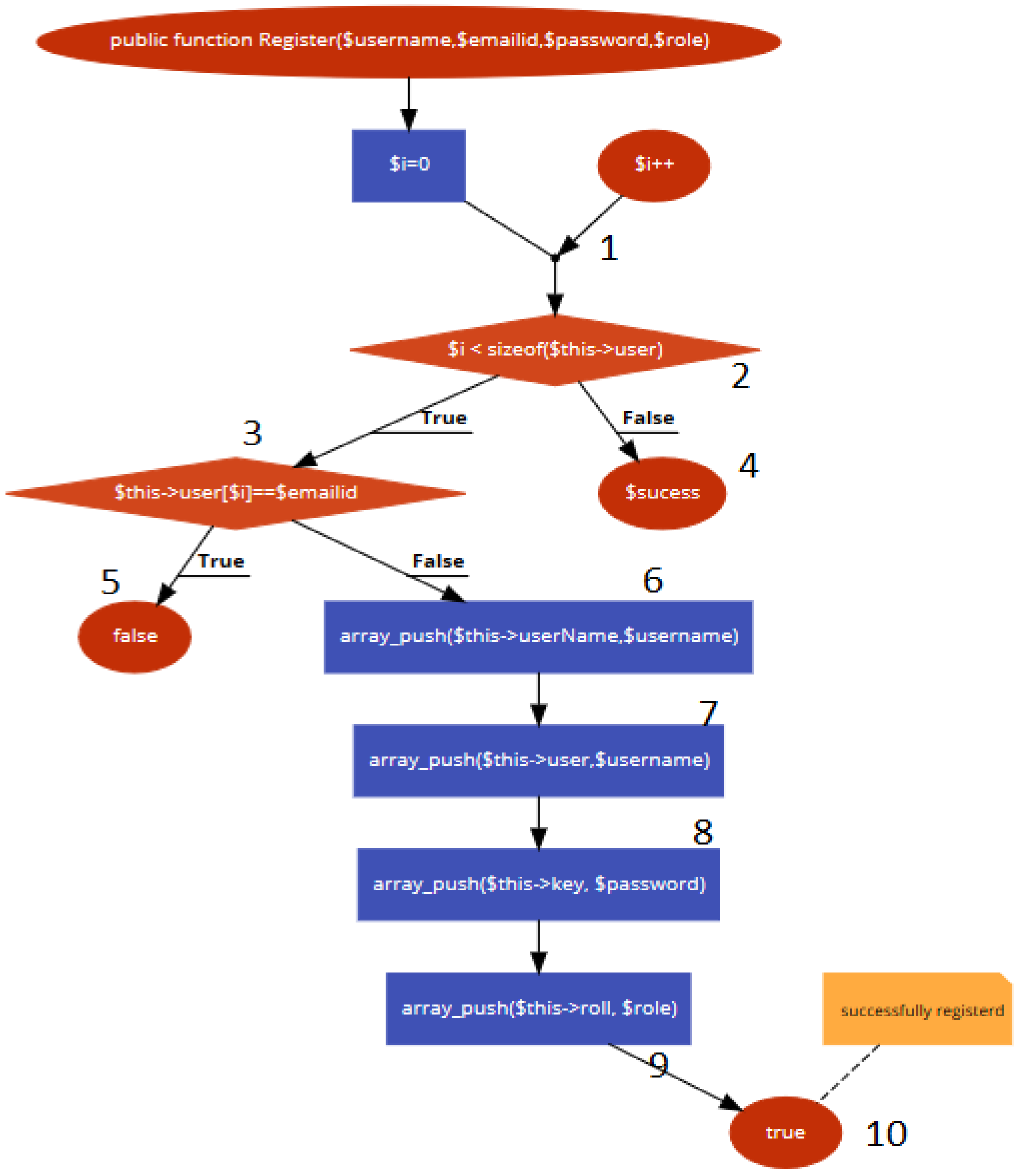

public function Register($username,$emailid,$password,$role)
$i=0
$i++
1
$i < sizeof($this->user)
2
True
False
3
4
$this->user[$i]==$emailid
$sucess
True
False
5
6
false
array_push($this->userName,$username)
7
array_push($this->user,$username)
8
array_push($this->key, $password)
array_push($this->roll, $role)
successfully registerd
9
true
10

FIGURE 5: REGISTRATION IN UNIT TESTING WITH ARRAYS.

Path selection

c. All path selection

I. 1-2[T]-3[F]-6-7-8-9-10

II. 1-2[T]-3[T]-5

III. 1-2[F]-4

Input are <username,email,password >

TABLE 4: REGISTRATION INPUT GENERATION

Input	Path
<null,null,null>	1-2[F]-4

<Yes,yes,yes>	1-2[T]-3[F]-6-7-8-9-10
<Yes,incorrect email,Yes>	1-2[T]-3[T]-5

Generating Test case

Input vector=filename, path
Predicate =$i<sizeof(this->$user). $this->$user[$i] ==emailID.
Path predicate = node 2,3.

Path predicate expression

1. For 1-2[F]-4
   $i>=sizeof (this->$user)
2. For 1-2[T]-3[F]-6-7-8-9-10
   ($this->$user[$i] ==emailID) ==True.
3. For 1-2[T]-3[T]-5
   ($this->$user[$i] ==emailID) ==False.

So, we can choice test case between this path.

4.2.1.2 Domain_Testing

Is a type of functional testing which tests the application by giving inputs and evaluating its appropriate outputs? It is a software testing technique in which the output of a system has to be tested with a minimum number of inputs in such a case to ensure that the system does not accept invalid and out of range input values.

TABLE 5: DOMAIN TESTING

Scenario	data	Test Case	Excepted outcome
InstructRegistration (username,email,password,	<Username,email,password> 1. <null,null,null> 2. <string,string,string) 3. <valid,invalid email,valid)	1.Test case input(null,null,null)	Must fill the field
		2. Test case 2 Input(string,string,string)	Succefully register
		3.Test case 3 Input(invalid email format	Invalid email format(must include@)
Login (emailId,password, role)	<email,password,role> 1.null,null,null 2.correct email,password,) 3. <email,incorrect password> 4.<incorrect email,pasword>	1.Test case 1	Must fill the field
		2.Test case 2	Redirect to home page
		3.Test case 3	Incorrect password
		4.Test Case 4	Incorrect email
Upload (fileToUpload,filename,path)	fileToupload,filename,path 1. <null, null,null> 2. <Yes,Yes,	1.Test Case 1	File is not upload
		2.Test Case 2	Incorrect path! Path is not exits.

4.2.1.2 Data flow Testing

Data flow testing is a family of test strategies based on selecting paths through the program's control flow in order to explore sequences of events related to the status of variables or data objects. Dataflow Testing focuses on the points at which variables receive values and the points at which these values are used.

4.2.2 Test Case Specification

TABLE 6: LOGIN TEST SPECIFICATION

Name: Log in.				
Purpose: to get a registered user logged into the system				
Test Data = Email(invalid email, valid email, empty) password(invalid , valid, empty)				
Module or unit on test= Login unit				
Assumption=user is login to the system				
Input	**Expected result**	**Data**	**Actual output**	**Pass/fail**
Empty Email and Valid Password	"please fill the email address field"	Empty username and a valid password	"please fill out this field " is displayed on the email address field	pass
Invalid Email and Valid password	"please enter email address"	Invalid Email and Any Valid confirmation username	" please enter email address"	Pass
Valid Email and Valid password	"Display home page"	If Email= valid, password=valid	"Display home page"	Pass
Empty Email and Empty Password	"please fill the field"	Empty Email and Empty password.	"please fill out this field"	Pass
Valid Email and Empty Password	"please fill the field"	Valid Email and Empty password.	"please fill out this field"	Pass
valid Email and invalid password	"Password doesn't match with email"	If email is some@gmail.com and password = "somepass" But user enters some@gmail.com and password = "otherpass"	"Password doesn't match"	pass

TABLE 7: TEST CASE SPECIFICATION FOR SHARE MATERIAL

Name: Share material				
Purpose: to make the user upload a material into the system store				
Test Data = Title(a valid string title,, empty) File(existing, empty)				
Assumption=user is login to the system				
Input	**Expected result**	**Data**	**Actual output**	**Pass/fail**
Empty title and an existing file	"Title is required"	Empty title and an existing file selected	"Title is required"	pass
A valid title and an empty file	"please enter email address"	Invalid Email and Any Valid confirmation username	" please enter email address"	Pass
A valid title and an existing file	"Display success message"	Title = "PHP tutorial" File selected = "PHP.pdf"	"Successfully uploaded"	Pass
Empty title and Empty file	"Title is required"	Empty title and Empty file	"Title is required"	Pass

TABLE 8: TEST CASE SPECIFICATION FOR DISCUSSION PANEL SECTION

Name: Discussion Panel section				
Purpose: to make sure that the user can successfully perform the discussion panel tasks.				
Test Data = Title(a valid string title, empty) File(existing, empty)				
Assumption=user is login to the system				
Input	**Expected result**	**Data**	**Actual output**	**Pass/fail**

Empty title and an existing file	"Title is required"	Empty title and an existing file selected	"Title is required"	pass
A valid title and an empty file	"please enter email address"	Invalid Email and Any Valid confirmation username	" please enter email address"	Pass
A valid title and an existing file	"Display success message"	Title = "PHP tutorial" File selected = "PHP.pdf"	"Successfully uploaded"	Pass

TABLE 9: TEST CASE SPECIFICATION FOR LOGOUT SECTION

Name: Logout Section				
Purpose: to make the user's session has been unset and destroyed and no longer can the user immediately login in without filling his /her password again.				
Pre-condition: the user is logged in.				
Test Data: variables that are stored in the user session id. Which are $_SESSION['name'] , $_SESSION['password']				
Assumption=user is login to the system				
Input	**Expected result**	**Data**	**Actual output**	**Pass/fail**
None	index.php page is displayed.	none	index.php page is displayed.	pass

TABLE 10: TEST CASE SPECIFICATION FOR SEARCH

Name: Search section				
Purpose: to make that user can search for certain content.				
Test Data = input("any object or primitive data type")				
Assumption=user is login to the system				
Input	**Expected result**	**Data**	**Actual output**	**Pass/fail**
searched input	"displaying posts	PHP book?	"displaying posts	pass

	related to the text PHP book ?”	If php book not found	related to the text What is PHP?” Display “No result found’	
Empty input	“please enter item to be searched!”	Empty input	“Display clear page”	Fail

TABLE 11 TEST CASE SPECIFICATION FOR BROWSE CONTENT BY LEVEL

Name: browse content by level				
Purpose: to browse the content by level				
Test Data = click the button				
Assumption: user login to the system.				
Input	**Expected result**	**Data**	**Actual output**	**Pass/fail**
On input data	“displaying content by level”	No input data	“display the content by level”	pass

TABLE 12: TEST PLAN SPECIFICATION FOR CREATE ACCOUNT SECTION

Name: Create account
Purpose: to make a user get registered in to the system
Test Data = User name(invalid email, valid email, empty) University(invalid , valid, empty) User email(invalid , valid, empty) Field of study(empty, choose from option) User password(invalid , valid, empty) Department (empty, choose from options) Confirm password(invalid , valid, empty) Year of study(empty , choose from options) Your role(empty, specified from options)

Input	**Expected result**	**Data**	**Actual output**	**Pass/fail**
Empty username and all other fields valid	“please fill the user name field”	Empty username and all others valid inputs	“please fill out this field ” on the user name field	pass
User password different from confirmation password	“passwords mismatch”	User password = “nopa” And confirm password = “pano”	“passwords not matchs”	Pass
User email empty and all other fields seted valid	“fill the email address field ”	If Email=””, all other fields set valid input	“fill out this field” on the email field	Pass
One of the password fields empty and others valid	“please fill the field”	Empty password field or empty confirmation field	“please fill out this field”	Pass
Valid Email and Empty Password	“please fill the field”	Valid Email and Empty password.	“please fill out this field”	Pass
All fields valid and password field value is equal to confirmation field value	“user account created”	Username = “Tefera” Email = tef@gmail.com Password=”pass” Confirmation = “pass” Role = “student” University= “Addis Ababa” Field of study=”Engineering” Department =“Electrical” Year of study=”4th”	User is registered to the system and success message displayed on screen	pass

Test case Specification based unit testing

TABLE 13:LOGIN TEST CASE SPECIFICATION BASED ON UNIT TESTING

Name: Log in.				
Purpose: to get a registered user logged into the system				

Test Data = Email(invalid email, valid email, empty)
password(invalid , valid, empty)
role(invalid , valid, empty)				

Module or unit on test= Login unit				
Assumptions =				
Input	**Expected result**	**Data**	**Actual output**	**Pass/fail**
Empty Email and Valid Password	False	Empty username and a valid password	False	pass
Invalid Email and Valid password	False	Invalid Email and Any Valid confirmation username	False	Pass
Valid Email and Valid password	True	If Email= valid, password=valid	True	Pass
Empty Email and Empty Password	False	Empty Email and Empty password.	False	Pass
Valid Email and Empty Password	False	Valid Email and Empty password.	False	Pass
valid Email and invalid password	False	If email is some@gmail.com and password = “somepass” But user enters some@gmail.com and password = “otherpass”	False	pass

Name: Create account

Purpose: to make a user get registered in to the system

Test Data = User name(invalid email, valid email, empty

User email(invalid , valid, empty)

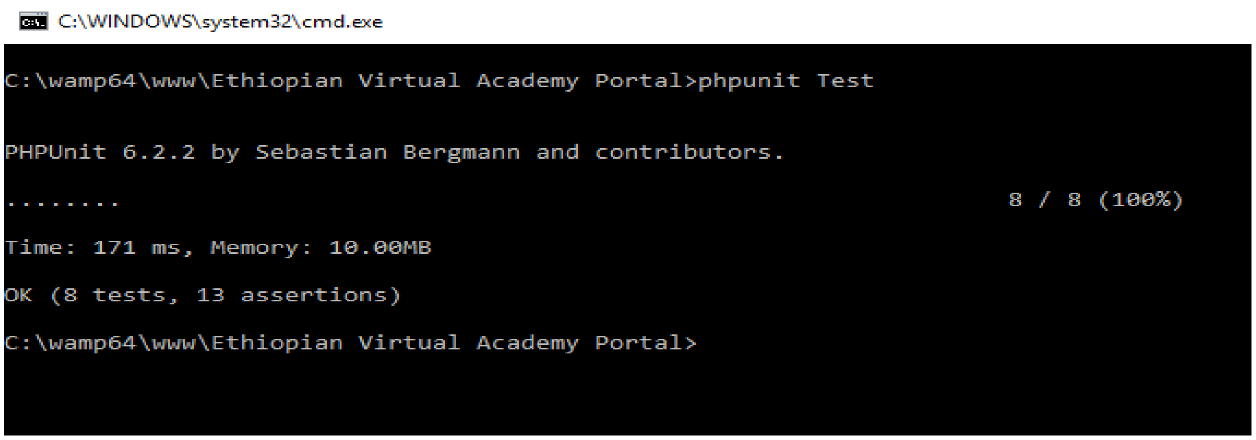
User password(invalid , valid, empty)

Your role(empty, specified from options)

Module or unit on test= Registration unit.

Input	**Expected result**	**Data**	**Actual output**	**Pass/fail**
Empty username and all other fields valid	False	Empty username and all others valid inputs	False	pass
User email empty and all other fields seted valid	False	If Email="", all other fields seted valid input	False	Pass
One of the password fields empty and others valid	False	Empty password field or empty confirmation field	False	Pass
All input are valid	True	All input are valid	True	Pass

Sample screenshot of the testing while succeeding in the assertion and testing:



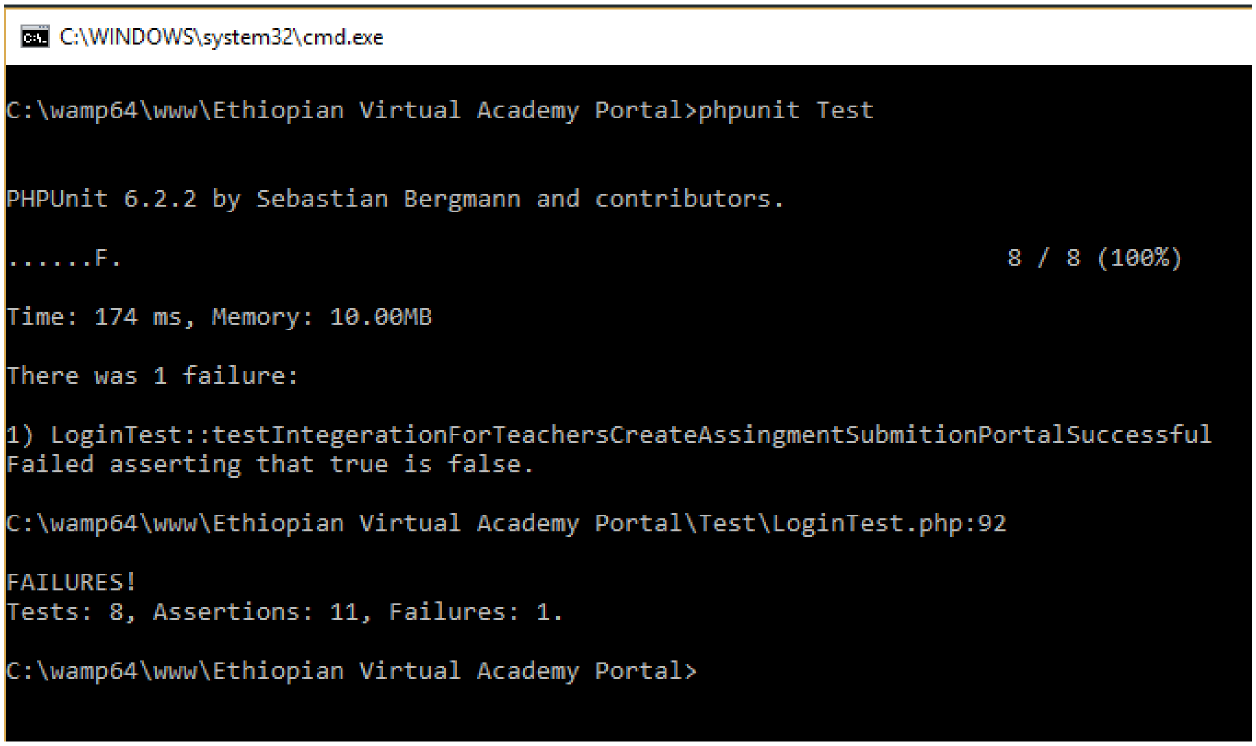
C:\WINDOWS\system32\cmd.exe

```
C:\wamp64\www\Ethiopian Virtual Academy Portal>phpunit Test

PHPUnit 6.2.2 by Sebastian Bergmann and contributors.
........                                                   8 / 8 (100%)
Time: 171 ms, Memory: 10.00MB
OK (8 tests, 13 assertions)
C:\wamp64\www\Ethiopian Virtual Academy Portal>
```

FIGURE 6: PHPUNIT TESTING PASS

Sample screenshot of the testing while failing the assertion and testing:



```
C:\WINDOWS\system32\cmd.exe

C:\wamp64\www\Ethiopian Virtual Academy Portal>phpunit Test


PHPUnit 6.2.2 by Sebastian Bergmann and contributors.

......F.                                                         8 / 8 (100%)

Time: 174 ms, Memory: 10.00MB

There was 1 failure:

1) LoginTest::testIntegerationForTeachersCreateAssingmentSubmitionPortalSuccessful
Failed asserting that true is false.

C:\wamp64\www\Ethiopian Virtual Academy Portal\Test\LoginTest.php:92

FAILURES!
Tests: 8, Assertions: 11, Failures: 1.

C:\wamp64\www\Ethiopian Virtual Academy Portal>
```

FIGURE 7: PHPUNIT TESTING FAIL

4.3 Integration Testing

Upon completion of unit testing, the units or modules are to be integrated which gives raise to integration testing. The purpose of integration testing is to verify the functional, performance, and reliability between the modules that are integrated.

Integration Strategies:

- Big-Bang Integration
- Top Down Integration
- Bottom Up Integration

4.3.1 Top down Testing

Top-down integration testing is an integration testing technique used in order to simulate the behavior of the lower-level modules that are not yet integrated. Stubs are the modules that act as temporary replacement for a called module and give the same output as that of the actual product.

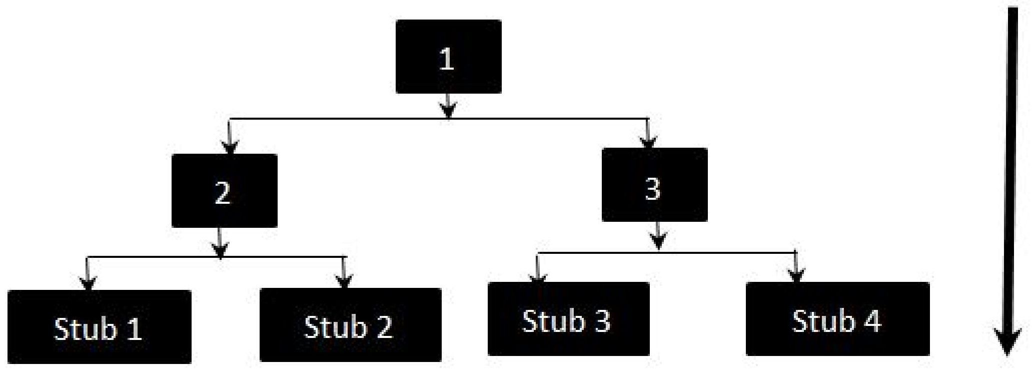


FIGURE 8:TOP DOWN TESTING

4.3.2 Bottom up Testing

Each component at lower hierarchy is tested individually and then the components that rely upon these components are tested.

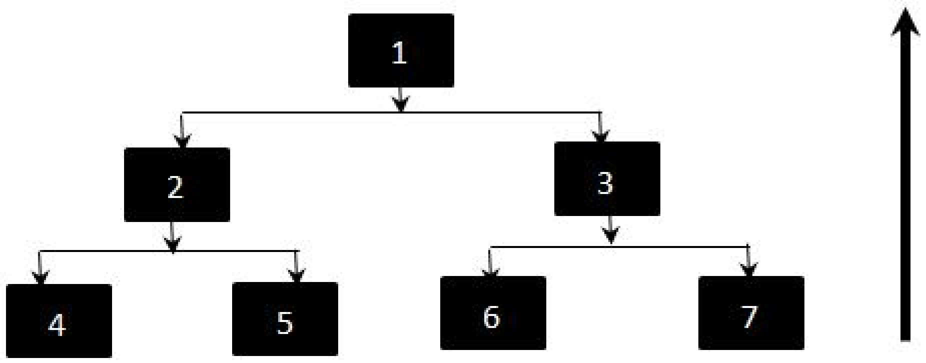


FIGURE 9: BOTTOM UP TESTING

4.3.3 Big Bang Testing

Big Bang Integration Testing is an integration testing strategy wherein all units are linked at once, resulting in a complete system. When this type of testing strategy is adopted, it is difficult to isolate any errors found, because attention is not paid to verifying the interfaces across individual units.

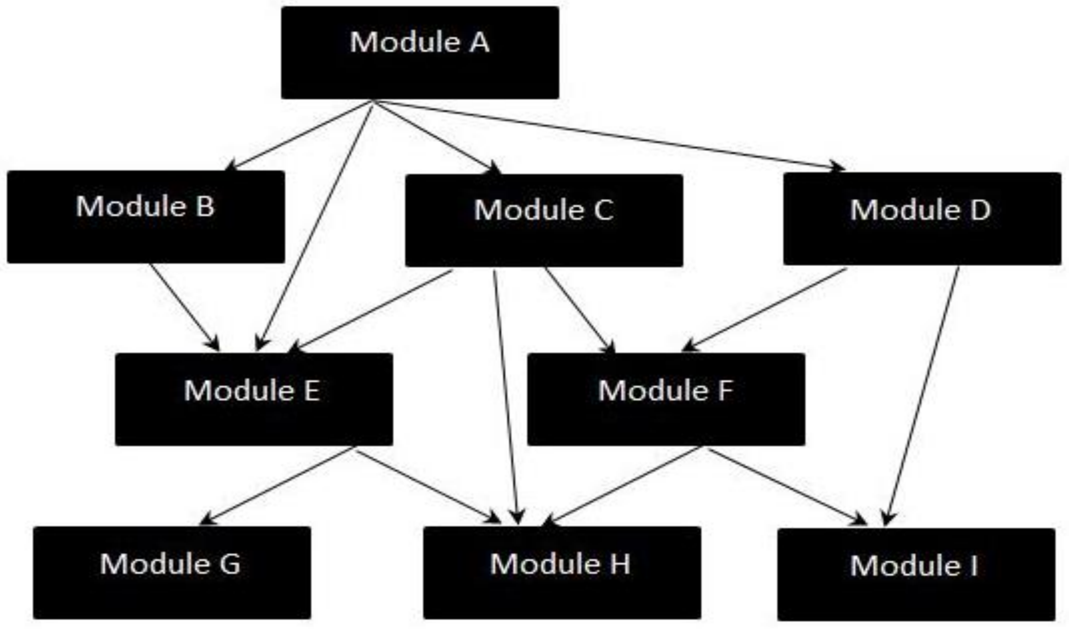


FIGURE 10: BIG BANG TESTING

4.3.4 Sandwich Testing

http://www.zyxware.com/articles/4139/what-is-sandwich-testing

4.4 System Testing

System testing of software is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.

4.4.1 Basic Test

The main objective of basic test we undertook is to provide an evidence that the system can be installed, configured, and brought to an operational state and we have proved that by hosting our application on publicly accessible web server.

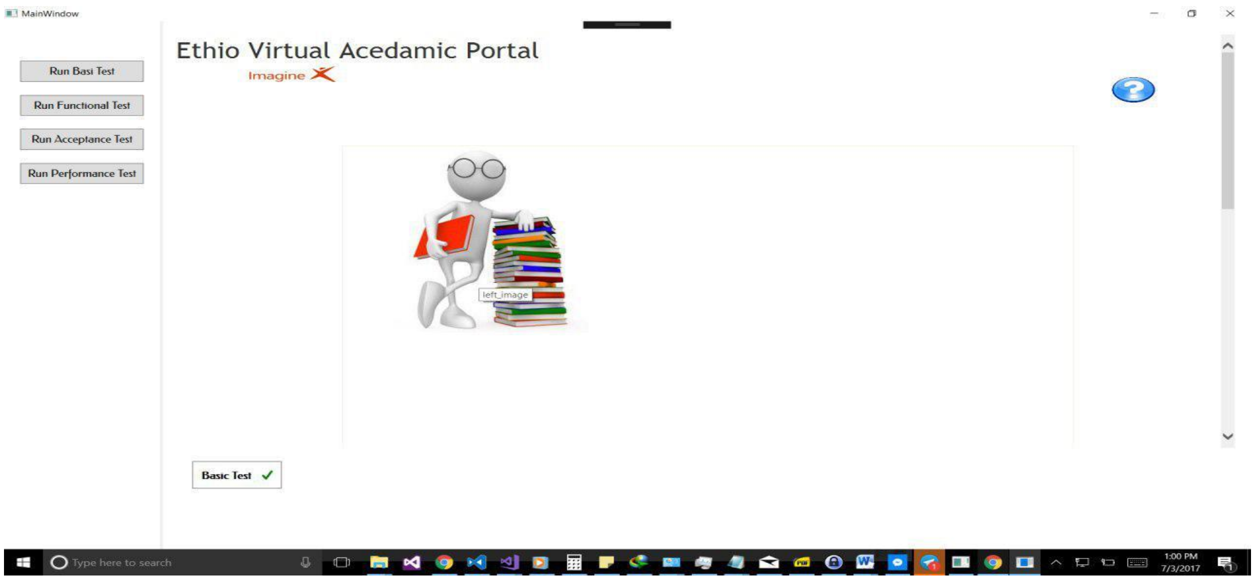


FIGURE 11: BASIC TESTING

4.4.2 Functional Test

Functional testing is carried out in order to find out unexpected behavior of the report. The characteristic of functional testing is to provide correctness, reliability, testability and accuracy of the report output/data.

What we did in the functional testing was testing the features and operational behavior of our web application product to ensure that it corresponds to specifications listed on the Software Requirements and Specifications (SRS) document.

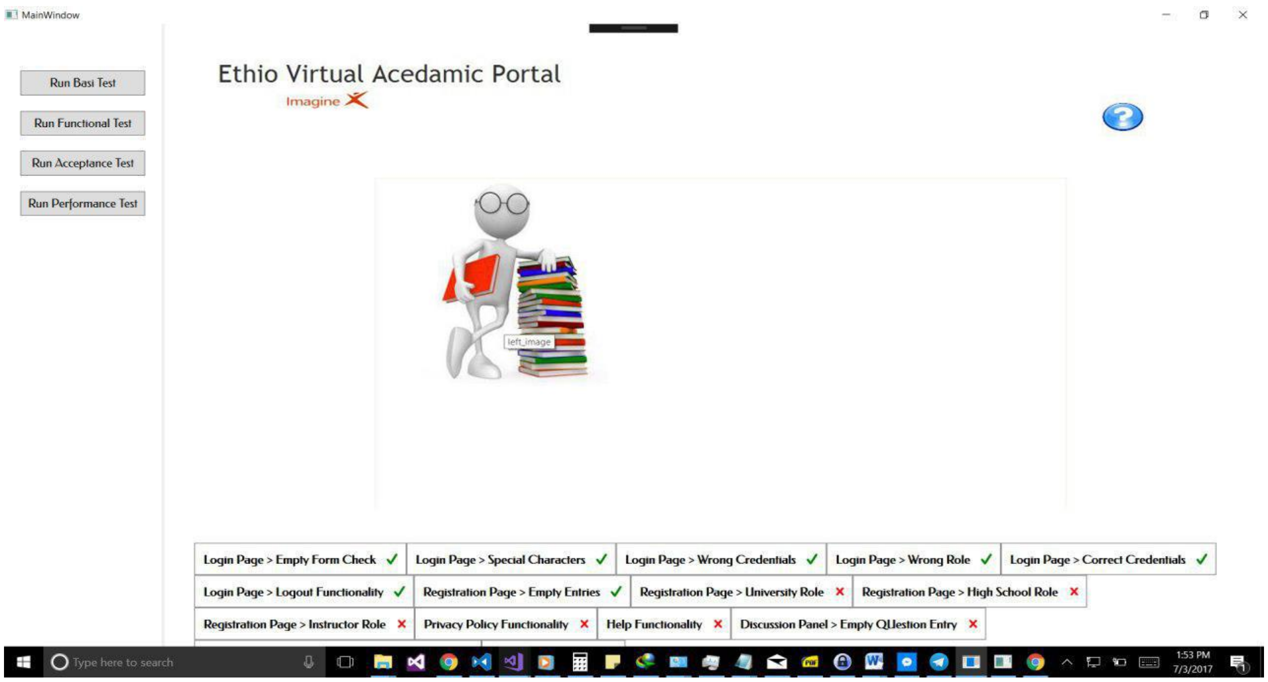
This test was undertaken by ignoring the internal mechanisms of the system or component and by solely focusing on the outputs generated in response to selected inputs and execution conditions.

The goal of Functional testing is to verify whether our product meets the intended functional specifications mentioned in our development documentation.

Functional Test Scenarios:

- ✓ Test all the mandatory fields should be validated.
- ✓ Test the system should not display the error message for optional fields.
- ✓ Test all input fields for empty entry
- ✓ Test all input fields for special characters.
- ✓ Test the functionality of the buttons available
- ✓ Test the Privacy Policy & Help is clearly defined and should be available for users.

Test if any functionality fails the user gets redirected to the custom error page.



Login Page > Empty Form Check ✓ | Login Page > Special Characters ✓ | Login Page > Wrong Credentials ✓ | Login Page > Wrong Role ✓ | Login Page > Correct Credentials ✓

Login Page > Logout Functionality ✓ | Registration Page > Empty Entries ✓ | Registration Page > University Role ✗ | Registration Page > High School Role ✗

Registration Page > Instructor Role ✗ | Privacy Policy Functionality ✗ | Help Functionality ✗ | Discussion Panel > Empty QUestion Entry ✗

FIGURE 12: FUNCTIONAL TESTING

4.4.3 Regression Test

Regression testing is defined as a type of software testing to confirm that a recent program or code change has not adversely affected existing features.

Regression testing is nothing but full or partial selection of already executed test cases which are re-executed to ensure existing functionalities work fine.

This testing is done to make sure that new code changes should not have side effects on the existing functionalities. It ensures that old code still works once the new code changes are done.

4.4.4 Performance Test

Performance testing is a type of testing to ensure software applications will perform well under their expected workload.

Features and Functionality supported by a software system is not the only concern. A software application's performance like its response time, reliability, resource usage and scalability do matter. The goal of performance testing is not to find bugs but to eliminate performance bottlenecks.

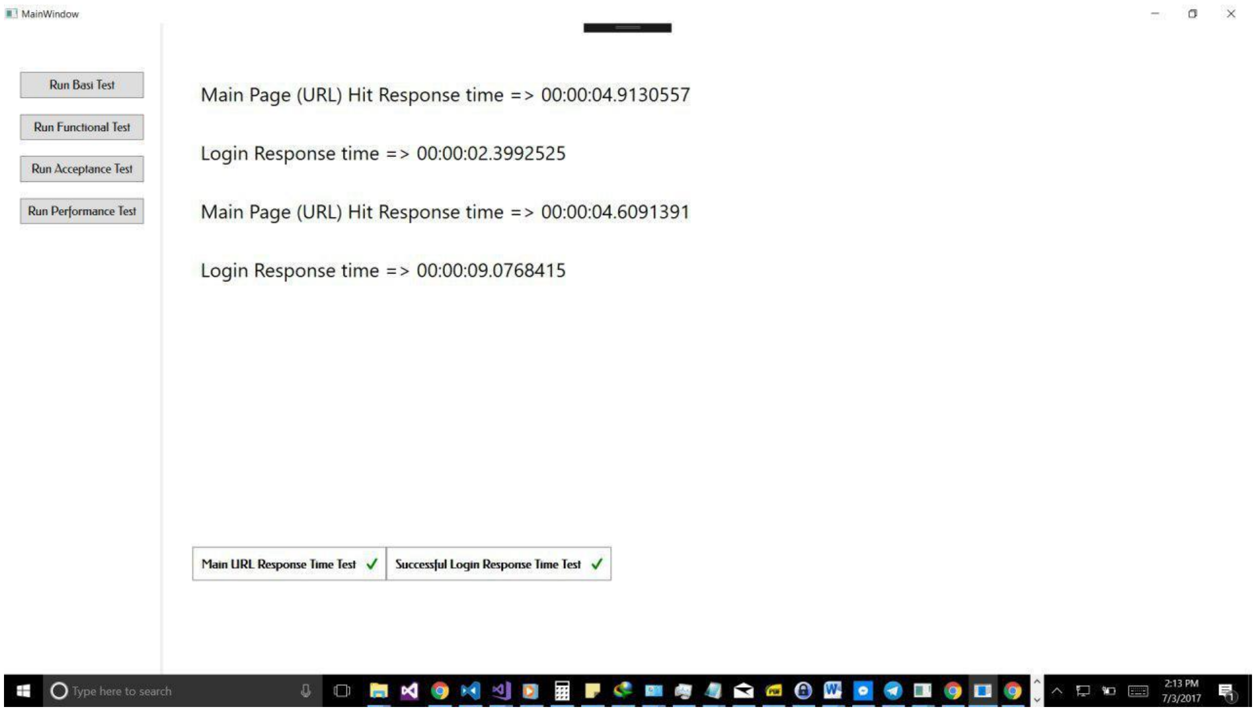
The focus of Performance testing is checking a software program's

- Speed - Determines whether the application responds quickly
- Scalability - Determines maximum user load the software application can handle.
- Stability - Determines if the application is stable under varying loads

Performance testing is popularly called as "Perf Testing" and is a subset of performance engineering.

And we have tried to implement this testing on some page hit speed. Some of the Hit Response Time tested pages are:

- Main Page (URL) Hit Response Time
- Login Response Time
- Main Page (URL) Hit Response Time
- Login Response Time



```
Main Page (URL) Hit Response time => 00:00:04.9130557

Login Response time => 00:00:02.3992525

Main Page (URL) Hit Response time => 00:00:04.6091391

Login Response time => 00:00:09.0768415
```

FIGURE 13: PERFORMANCE TESTING

4.4.4 Acceptance Test

The purpose behind user acceptance testing is to conform that system is developed according to the specified user requirements and is ready for operational use. Acceptance testing is carried out at two levels - Alpha and Beta Testing. User acceptance testing (UAT) will be done at the Client.

Bibliography

Web based

1. http://www.fit.vutbr.cz/study/courses/ITS/public/ieee829.html access date 2/7/2017
2. http://www.guru99.com/complete-web-application-testing-checklist.html

Books

1. Text Book - Software Testing and Quality Assurance -Theory and Practice
2. Software Quality Engineering Testing Quality Assurance and Quantifiable Improvement Wiley