**IMPLEMENTATION REPORT FOR**

**A100 Testing and Tracking System**

Spring 2015

Fairfield University, School of Engineering

***Prepared by:***

***Divya Kamath***

***Syed Sohail***

***Sagar Kannamangala Singri Gowda***

***Subbarayudu Bodavula***

***Luke Auger***

***Nicholas Cummings***

|  |  |  |
| --- | --- | --- |
|  |  |  |
| S.No | Topic | Page |
| 1 | Implementation Language | 4 |
| 2.1 | Coding Conventions for implementing A100 | 5 |
| 2.2 | Purpose | 5 |
| 2.3 | File Names | 5 |
| 2.3.1 | Coding Conventions for php and MySQL | 5 |
| 2.3.2 | Code Structure | 6 |
| 2.3.3 | Control Structure | 6 |
| 2.3.4 | Ternary Operator | 6 |
| 2.3.5 | Naming | 6 |
| 2.3.6 | Variables | 6 |
| 2.3.7 | Comments And Documentation | 7 |
| 3.0 | Some examples of code with comments | 8 |

1. ***Implementation Language***

PHP is a widely-used general-purpose scripting language that is especially suited for Web development and can be embedded into HTML. The project A100 Testing and Tracking system is done by using procedural PHP and MYSQL for the back end and Html, java scripts, CSS for the front end. All the database tables were created in MYSQL.

For using the webserver we can use either WAMP or LAMP. These are bundle installers so that all the required packages to run a server can be installed into our system. In the system we have used WAMP environment to configure the server in our system.

All server side coding for A100 testing and tracking system was done in php. The database system used is MySQL which is an open source RDBMS. It is very light and highly functional. Also with php and MySQL being used together very frequently a lot of online support was available.

Java script is used to write the client side coding.

***2. Coding Conventions for A100 Testing and tracking System***

**2.1 Purpose**

To give a description of the coding standards that are used in the application.

**2.2 File Names**

HTML documents use the suffix *.html*

PHP documents use the suffix .php

XML files use the suffix .xml

MySQL files use the suffix .sql

**2.3 Coding Conventions for PHP with MySQL**

The below mentioned are the coding conventions used within the files written in

php.

**2.3.1Code Structure:**

Assignment Expressions

Assignment can be used to fetch the results from the database. For this effective way to use a while () clause which will legitimate for iterations.

$res = $ref\_db->query(‘SELECT \* FROM players where tid=””.$code.””’)

While ($row = $dbr->fetch Objects($res)) {

Showrow($row);

}

Spaces

When writing spaces on put space on either side of binary operators.

$a = $b + $c;

And when writing parenthesis on the inside give spaces except where the parenthesis are empty.

$var = getF( $var1 );

$var = getB();

**2.3.2 Control Structures**

In php we have control structure which are similar to other program languages they are if, while, for, foreach and switch as well as catch keyword should follow by a space.

If ( isF() )

{

$a = ‘a100’;

}

When using type casting we can directly write

(int)$f;

When writing a comment one space between the # or // character need to be used

// comment

**2.3.3 Ternary operator**

The ternary operator can be most profitable if the expressions are short

$ln = isset ( $this->nm[‘name’]) ? $this->nm[‘name’]: false;

**2.3.4 Naming**

Use lowerCamelCase when naming functions or variables. For example:

private function getData( $Player, $Team )

Use UpperCamelCase when naming classes: class ImportantClass. Use uppercase with underscores for global and class constants: DB\_MASTER.

**2.3.5 Variables**

It is common to work with an instance of the Database class; we have a naming convention for these which helps keep track of the nature of the server to which we are connected. This is of particular importance in replicated environments, such as Wikimedia and other large wikis; in development environments there is usually no difference between the two types, which can conceal subtle errors.

$dbw – a Database object for writing (a master connection)

$dbr – a Database object for non-concurrency-sensitive reading

**2.3.6 Comments and Documentation**

Use /\*\* to begin the comments, instead of the Qt-style formatting /\*!.

It is essential that code need to be well documented. The format for parameters is:

@param type $paramName description of this parameter

Multiple types can be listed by separating with a pipe character:

@param datatype1|datatype2 $paramName description of this parameter

class F {

/\*\*

\* @var array $bar: Description here

\* @example array( 'f' => B, 'quux' => B, .. )

\*/

protected $bar;

/\*\*

\* Description here, following by documentation of the parameters.

\*

\* @since 1.42

\*

\* @param FooContext $context context for decoding Foos

\* @param array|string $options Optionally pass extra options. Either a string or an array of strings.

\* @return Foo|null: New instance of Foo or null of quuxification failed.

**Some Examples**:-

Code for login page

<?php

$title = "Login"; // Every page has a title //

include "home-header.php"; //this command includes home-header.php file//

if ((isset($\_POST)) and (!empty($\_POST)))

{

$username = filter\_input(INPUT\_POST, 'username');

$password = filter\_input(INPUT\_POST, 'password');

$verify = mysqli\_query($link,"SELECT \* FROM `users` WHERE (`username` OR `email`)= '" . $username . "' AND `password` = '" . $password . "'");

//the user id and the password are stored in the session where session id Is created dynamically//

if (mysqli\_num\_rows($verify) == '1')

{

$d = mysqli\_fetch\_array($verify);

$\_SESSION['user\_id'] = $d['id'];

$\_SESSION['username'] = $d['username'];

if ($d['role'] == 1) {

$\_SESSION['user'] = true;

header("location:profile.php");

} else if ($d['role'] == 0) {

$\_SESSION['login'] = true;

header("location:scores.php");

}

}

Else // If the given user name and password are incorrect it shows the alert message//

{

$message = "<div class='alert alert-danger'>Incorrect Login credentials.</div>";

header("refresh:2;url = login.php");

}

}

?>

//this is the html code for the front end//

<div class="col-sm-5 center-block center"> <?= (!empty($message)) ? $message : "" ?></div>

<div class="h1 clearfix"></div>

<div class="account-wall"><img class="profile-img" src="images/LoginImage.png" alt="No Image Found">

<form class="form-signin" method="post">

<input type="text" class="form-control" placeholder="Email or Username" name="username" required

autofocus>

<input type="password" class="form-control" placeholder="Password" name="password" required>

<button class="btn btn-lg btn-primary btn-block" type="submit" name="login"><i

class="glyphicon glyphicon-log-in"></i> Sign In

</button>

<div class="text-right">New User ? <a href="register.php">Click here.</a> </div>

</form>

</div>

<div class="h1 clearfix"></div>

<?php include "footer.php" ?>

Code for selecting the skill

include "config.php";

if(!empty($\_GET['skill'])) //get method is used to get the selected skill//

{

$skill = filter\_input(INPUT\_GET,'skill');

//Query to fetch the skill using id//

$check\_skill = mysqli\_query($link,"SELECT `has\_test` from `courses` WHERE `id`='".$skill."'");

$s = mysqli\_fetch\_row($check\_skill);

echo $s[0];

}

if(isset($\_POST)and(!empty($\_POST))) // post method is used to show the skill//

{

$user\_id = filter\_input(INPUT\_POST,'user\_id');

$skill\_id = filter\_input(INPUT\_POST,'skill\_id');

$test\_id = filter\_input(INPUT\_POST,'test\_id');

$data = mysqli\_query($link,"SELECT `answer` from `answers` WHERE `answered\_by`='".$user\_id."' and `course\_id`='".$skill\_id."' and `test\_id`='".$test\_id."'");

if(mysqli\_num\_rows($data)=='1')

{

$d= mysqli\_fetch\_row($data);

echo "<textarea readonly class='from-control' name='answer' rows='10'>".$d['0']."</textarea>";

}

Else

{

echo "0";

}