* **Project Scope:**

This software product the dance class management to improve their services for all  
the students of the dance class. This also reduce the manual work of the persons in admin   
penal and the bundle of registers that were search when to find the information of a   
previous student, because through this system you can store the data of the students   
of the dance class.  
Through this you can check the personal profile of all the current students   
within few minutes the data base of the system will help you to check a particular one.   
The system will help you to check the attendance of every student and the student’s   
dance class dues. The students of the dance class will be recognized from the ID number allocated batch wise.

* **WORKING OF PROJECT**
* **Front End**

1. We have designed our “Project” using many programing languages.

2. For starters we have designed our front end using “html , javascript , css”.

3. Html is a scripting language that is used to design static web pages, here it is used to design forms eg-‘Registration form’, ’Exam form',’Dance program form’,Attendance Record for students and teaching staff’s.

4. Javascript also a scripting language but much more powerful than simple html, it has some features of java, we can do oop programing also, here it is used to add animation to make our web site fun to browse

eg-‘Form Validation’

6. What we are doing here is accepting all data the Manging Staff or student has filled, This data is taken in html forms and directed to our middle ware php and then to Mongodb.

* **Middle ware**

1. We have used ‘php’ to take data that has been entered by the user in html form and direct it to our database.

2. PHP Hypertext pre-processor php is used because we cannot do a html and database connectivity directly. So it can be said as php is the bridge or interface between front end and back end.

3. No operation is done directly on database i.e we do not actually fire queries n terminal on database.

4. Whatever data is being entered on front end will be entered to data base by php-database connectivity code, this data will be placed accordingly in database i.e Database queries will be fired through this php connectivity code.

* **Back End**

1. Back end means database end, we are using mongodb as our database it is a nosql database. Database is used as a suitable means of storing data.

2. Nosql means not only sql, it provides a mechanism for storage and retrieval of data that is modelled in means other than tabular format

Used in relational database

3. Widely used because simplicity of design, horizontal scaling, finer control over availability.

4. In mongodb there is no concept of tables as in mysql, here we have collections to store and manage our data.

5.The storage/retrieval of data in/from our database is done through ‘php’ that is our middle ware, we can see the changes made to our database by queries that are fired through ‘php’ by firing appropriate queries.

* **Requirements**
* **Hardware requirements**

No special hardware requirements, any basic configuration will do

(We have used)

1. Intel I3 processor

2. 2GB Ram

3. 165.5 GB hard disk

* **Software Requirements**

1. Operating system-Windows 8.

2. Apache server

3. Mongodb

4. PHP(version 5.2)

5. Browser

\*Note-: Browser should be updated to support some of the functionality

All of the above mentioned software’s are freely available and are open source.

* **Normalization:**
* Normalization is a *process* in which we systematically examine relations for *anomalies* and, when detected, remove those anomalies by splitting up the relation into two new, related, relations.
* Normalization is an important part of the database development process: Often during normalization, the database designers get their first real look into how the data are going to interact in the database.
* Finding problems with the database structure at this stage is strongly preferred to finding problems further along in the development process because at this point it is fairly easy to cycle back to the conceptual model (Entity Relationship model) and make changes.
* Normalization can also be thought of as a trade-off between data redundancy and performance.
* **First Normal Form(1nf):**

A relation is said to be in 1NF if it contains no non-atomic values and each row can provide a unique combination of values.

* **Second Normal Form (2NF)**

A relation is said to be in 2NF f if it is already in 1NF and each and every attribute fully depends on the primary key of the relation. Speaking inversely, if a table has some attributes which is not dependant on the primary key of that table, then it is not in 2NF.

### Third Normal Form (3NF)

A relation is said to be in 3NF, if it is already in 2NF and there exists no **transitive dependency** in that relation. Speaking inversely, if a table contains transitive dependency, then it is not in 3NF, and the table must be split to bring it into 3NF.

What is a transitive dependency?

Within a relation if we see

A → B [B depends on A]   
And   
B → C [C depends on B]   
Then we may derive   
A → C[C depends on A]

* **source code:**
* **Student registration form:**

<html><head>

<title>validation</title>

<script language="javascript">

function verify() {

if (document.form1.Name.value == "")

{

alert("Please give the name");

document.form1.Name.focus();

return false;

}

if (document.form1.Address.value == "")

{

alert("Please give address");

document.form1.Address.focus();

return false;

}

return( true );

}

</script>

</head>

<!-- Creating Form -->

<body bgcolor="white"><center><font size=5>

<form method="POST" action="successstud.php" name="form1">

<b><h3>REGISTRATION FORM</h3></b>

<table border=1 align=center bgcolor="Lavender" cellpadding=20px><br>

<td colspan=6 align=center><b>STUDENT INFORMATION</td>

<form name ="registration Form" action="successstud.php " method="POST"><tr>

<td>Name:</td><td> <input type = "text" name = "Name" size=30 maxlength=30></td></tr>

<tr><td>Phone\_No :</td> <td><input type = "text" name = "Ph\_no" size=12 maxlength=12></td></tr>

<tr><td>Address :</td><td><textarea name="Address" id="Address" size=30 maxlength=30></textarea> </td></tr>

<tr><td>email\_id:</td><td> <input type = "text" name = "eid" size=30 maxlength=30></td></tr>

<tr><td>Date:</td><td><textarea name="date" rows="2" cols="20"></textarea></td></tr>

<tr><td> Dance\_Type:<td><select name="dance\_type">

<option value="-1" selected>select..</option>

<option value="Bharatnatyam">Bharatnatyam</option>

<option value="Hip-Hop">Hip-Hop</option> </select></td> </td></tr>

<tr><td colspan =2 align="center">

<button type="submit">submit</button>

<button type="Reset">Reset</button></td></tr></table><br /><br />

<center><h3><u><b><a href="http://localhost/stud.html">BACK</a></u></b></h3></center></form></body></html>

* **PHP code:**

<?php

$m=new MongoClient();

$db=$m->mydatabase;

$collection=$db->sinfo;

$name=$\_POST['Name'];

$Phno=$\_POST['Ph\_no'];

$add=$\_POST['Address'];

$email=$\_POST['eid'];

$DOB=$\_POST['date'];

$Dtype=$\_POST['dance\_type'];

$disp = $collection->find();

foreach($disp as $doc)

{

if($email===$doc['email']){

$f=1;

}

else{

$f=0;

}

}

if($f===1){

echo"Email already exists";

}

else{

$dc=array("name"=>$name,"phno"=>$Phno,"add"=>$add,"email"=>$email,"dob"=>$DOB,"dtype"=>$Dtype);

$inse=$collection->insert($dc); echo "<center>";

echo "<h1>REGISTERED SUCCESSFULLY!!!!</h1><br/>";

echo "document inserted successfully";

echo "<br /><br />";

echo "<h2><u><b><a href='http://localhost/stud\_info.html'>BACK</a></u></b></h2>";

echo "</center>"; } ?>

* **html code for managing staff login:**

<html>

<body align="center"><br><br><br><br>

<font size="+4">MANAGING STAFF LOGIN</font><br><br>

<form method="POST" action="password.php" name="form1">

<table align="center"><tr><td>

<form action="password.php" method="POST">

Username: <input type="text" name="Username"><br>

password: <input type="password" name="password"><br><br>

<button type="submit"> Submit</button>

<button type="Reset">Reset</button><button type="Previous" ><a href="eg.html">previous</a></button></form></html></body>

* **PHP code:**

<?php

$m=new MongoClient();

$db=$m->mydatabase;

$collection=$db->minfo;

$user=$\_POST['Username'];

$pswd=$\_POST['password'];

$disp = $collection->find();

foreach($disp as $doc)

{

if($pswd===$doc['pswd']){

$f=1;

}

else{

$f=0;

}

}

if($f===1){

echo '<br><br><br><br><h1><b><center><a href="managing.html">Click here </a></center></b></h1>';

}

else{

echo "<br><br><br><br><h1><b><center>wrong username or password!!!!</center></b></h1>"; }?>

* **HTML code for attendance:**

<html><head>

<title>table</title>

<body bgcolor="white">

<table align="center" border="1" bgcolor="Lavender"style="width:75%" cellpadding="15px">

<form name ="registration Form" action = "satnd.php" method="POST">

<caption><font color="Black"><h2><b><u>STUDENT'S ATTENDANCE</b></caption>

<tr><td><u>stud\_id</td>

<td><u>Batch\_no</td>

<td><u>Date</td>

<td><u>Status</td></tr>

<tr>

<td><textarea rows="2" cols="20" name="11"></textarea></td>

<td><textarea rows="2" cols="20" name="12"></textarea></td>

<td><textarea rows="2" cols="20" name="13"></textarea></td>

<td><textarea rows="2" cols="20" name="14"></textarea></td></tr>

<tr>

<td><textarea rows="2" cols="20" name="21"></textarea></td>

<td><textarea rows="2" cols="20" name="22"></textarea></td>

<td><textarea rows="2" cols="20" name="23"></textarea></td>

<td><textarea rows="2" cols="20" name="24"></textarea></td></tr>

<tr>

<td><textarea rows="2" cols="20" name="31"></textarea></td>

<td><textarea rows="2" cols="20" name="32"></textarea></td>

<td><textarea rows="2" cols="20" name="33"></textarea></td>

<td><textarea rows="2" cols="20" name="34"></textarea></td></tr>

<tr>

<td><textarea rows="2" cols="20" name="41"></textarea></td>

<td><textarea rows="2" cols="20" name="42"></textarea></td>

<td><textarea rows="2" cols="20" name="43"></textarea></td>

<td><textarea rows="2" cols="20" name="44"></textarea></td></tr></table>

<table align="center" cellpadding="20px" cellspacing="15px">

<tr><td><input type="submit" name="submit" value="submit"></td></tr></table></form></head>

</body></html>

* **PHP code:**

<html>

<body align="center"><br><br><br><br>

<h1> ATTENDANCE CHECKED!!!!</h1>

<?php

$m=new MongoClient();

echo "connected";

$db=$m->mydatabase;

echo"mydb database selected";

$collection=$db->sattendance;

$stud\_id1=$\_POST['11'];

$Batch\_no1=$\_POST['12'];

$Date1=$\_POST['13'];

$Status1=$\_POST['14'];

$dc=array("id1"=>$stud\_id1,"batch1"=>$Batch\_no1,"date1"=>$Date1,"status1"=>$Status1);

$inse=$collection->insert($dc);

$stud\_id2=$\_POST['21'];

$Batch\_no2=$\_POST['22'];

$Date2=$\_POST['23'];

$Status2=$\_POST['24'];

$dc1=array("id2"=>$stud\_id2,"batch2"=>$Batch\_no2,"date2"=>$Date2,"status2"=>$Status2);

$inse=$collection->insert($dc1);

$stud\_id3=$\_POST['31'];

$Batch\_no3=$\_POST['32'];

$Date3=$\_POST['33'];

$Status3=$\_POST['34'];

$dc2=array("id3"=>$stud\_id3,"batch3"=>$Batch\_no3,"date3"=>$Date3,"status3"=>$Status3);

$inse=$collection->insert($dc2);

$stud\_id4=$\_POST['41'];

$Batch\_no4=$\_POST['42'];

$Status4=$\_POST['44'];

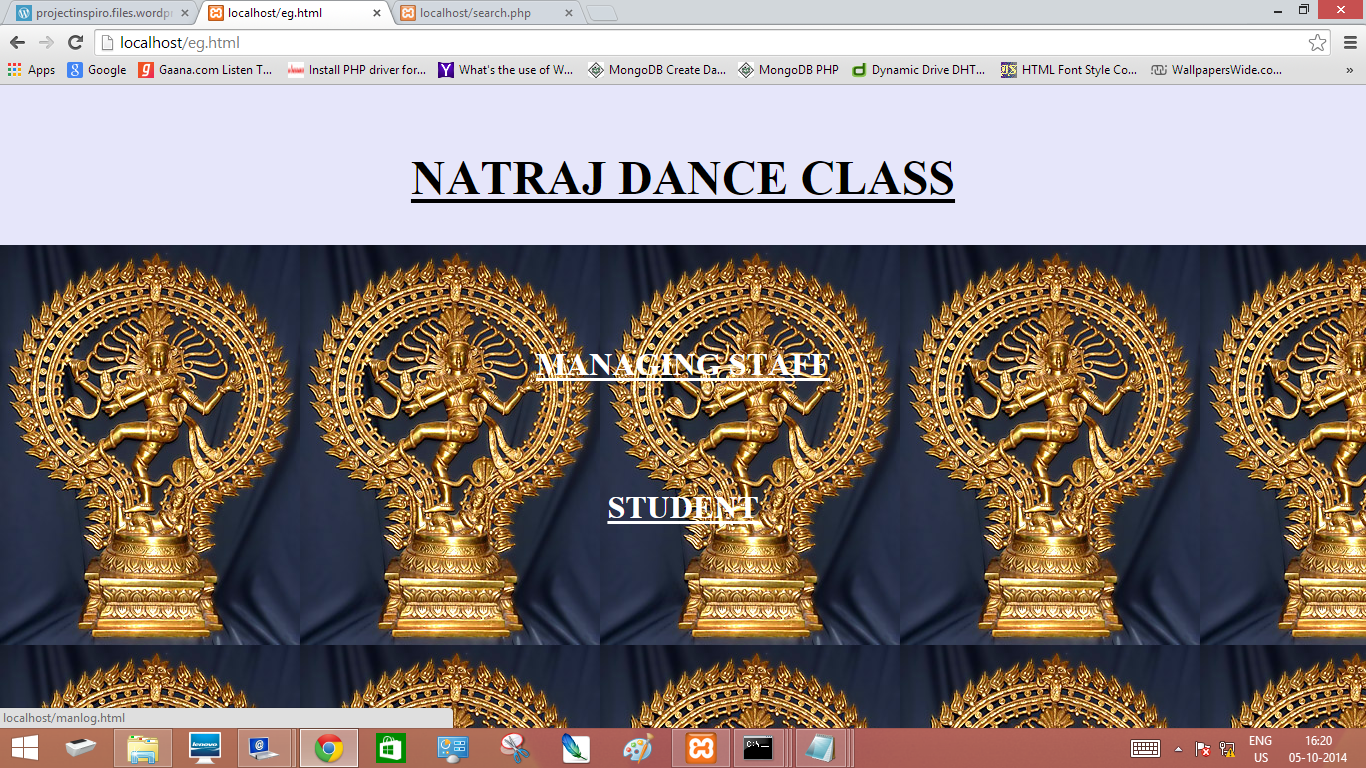
$Date4=$\_POST['43'];

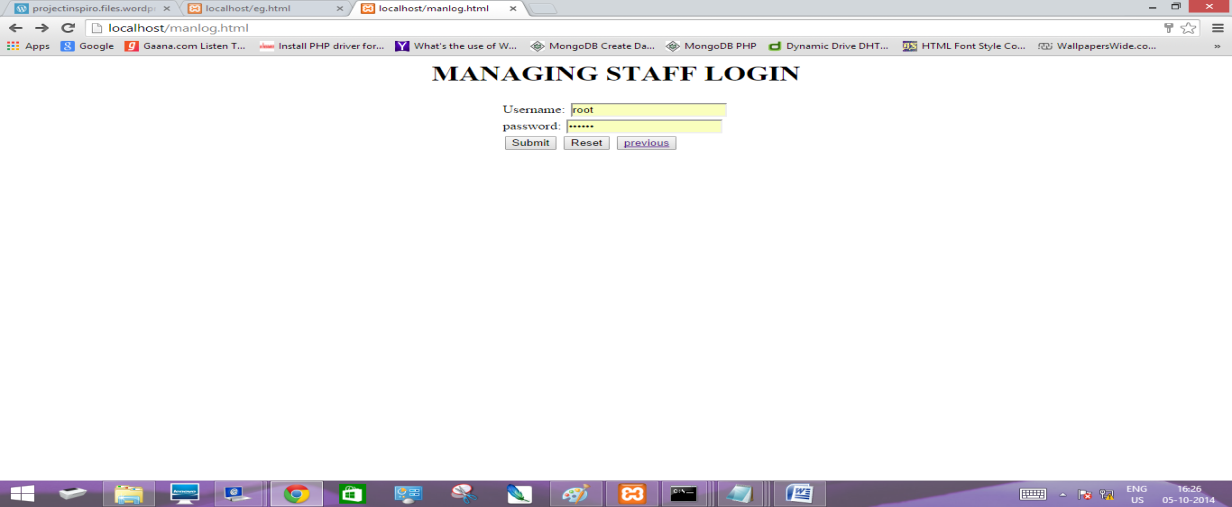
$dc3=array("id4"=>$stud\_id4,"batch4"=>$Batch\_no4,"date4"=>$Date4,"status4"=>$Status4);

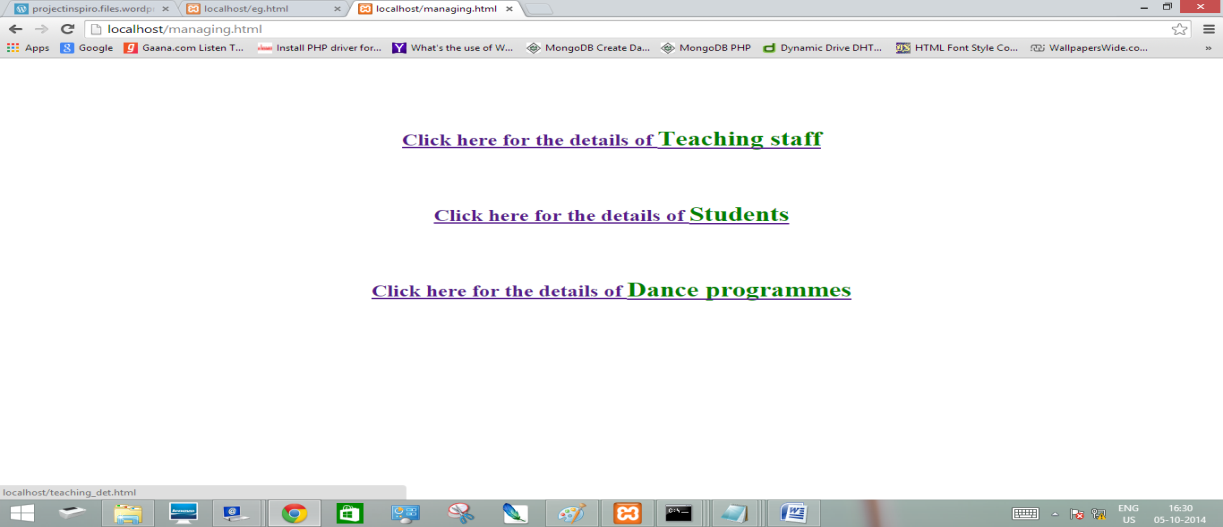
$inse=$collection->insert($dc3);

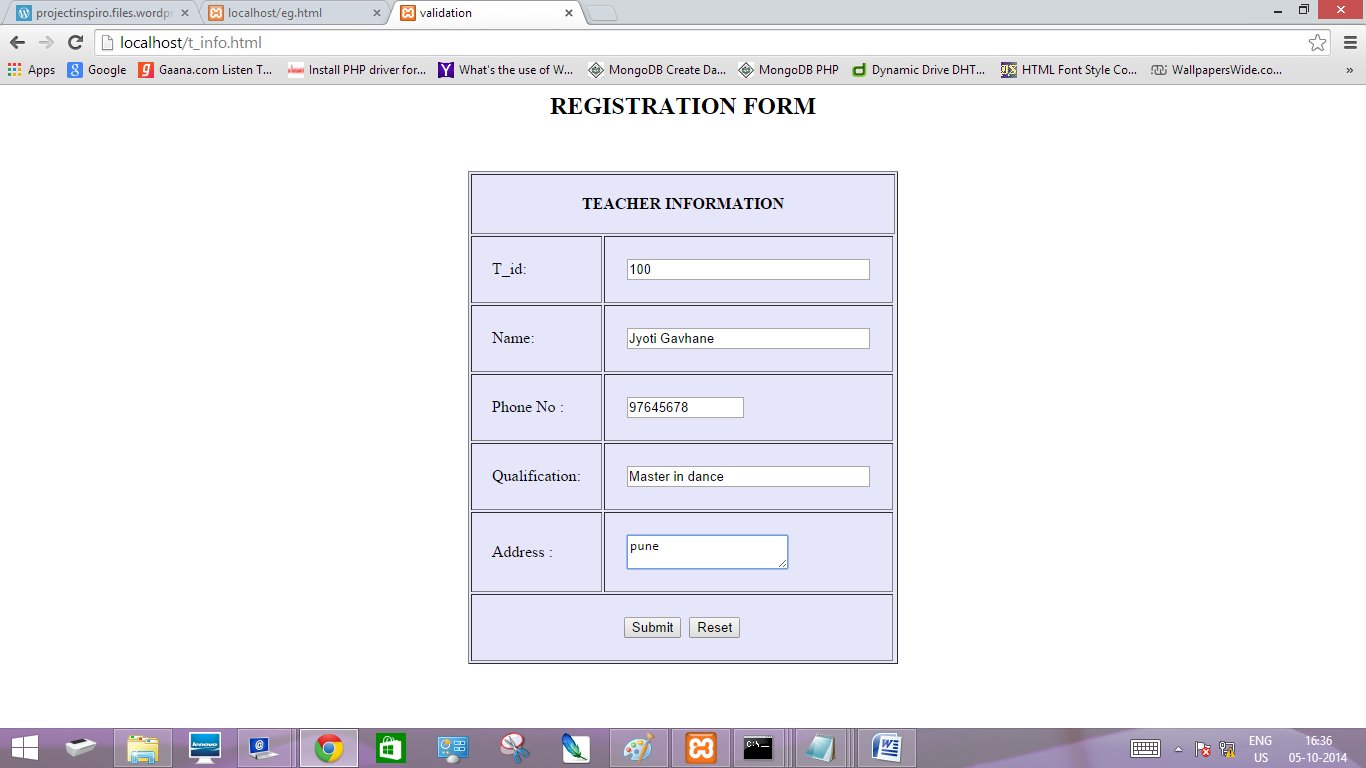
echo"document inserted successfully"; ?></body></html>

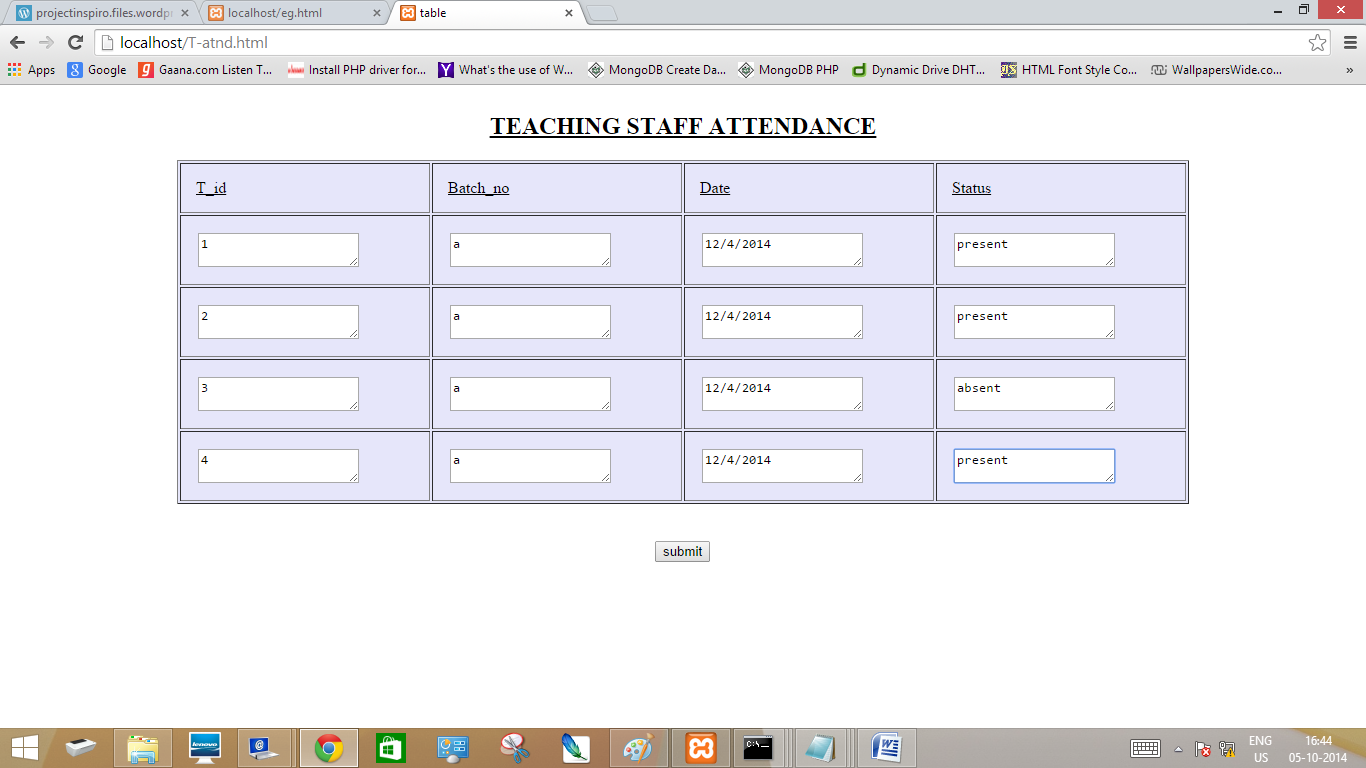
* **graphical user interface:**











* **Testing :**
* **TEST PROCEDURES:**

The project needs to be fully tested as it needs to be integrated to an internal existing system. Also the data that it processes and its accuracy are equally important. Since the system works on the network it was thoroughly tested internally and can only then be implemented incrementally to the other modules. The methods adopted for testing are as follows:

1. **Unit Testing:**

The entire code that is used in the project has been tested thoroughly. The code is tested by subjecting it with various inputs and in various amounts, with various databases. The output is then tested against project requirements. As stated above they were first tested locally.

1. **Integration testing:**

It is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested components and build a program structure that has been dictated by design. The system involves data transfer through TCP/IP; this made it necessary to test whether the input to the output of one module properly enters as input of the other module. For this purpose integration testing has to be done.

1. **System Testing:**

In system testing the primary objective was to fully exercise the computer based system.

1. **Recovery Testing:**

In recovery testing the system was forced to fall in a variety of ways and verified that recovery is properly performed. If the recovery was automatic (performed by system itself), reinitialization, data recovery and restart were evaluated for correctness. If the recovery requires human intervention, the mean-time –to repair(MTTR) was evaluated to determine that it is in acceptable limits

1. **Security Testing:**

As the system has to be operated on LAN, security testing determined to verify that protection mechanisms built in to the system protect it from improper penetration.

1. **Black Box Testing:**

This is accomplished by making the system work on the internal LAN for about 2-3 weeks. This testing is the deciding one and also the most exhaustive as it is not recommended to test the system on live servers directly as it may cause any incompetence, which would interrupt the existing work setup.

* **Future work:**

I.We here have created a very small version of a very big project, in upcoming future we intend to make this a full scale fully functioning Website

II. We have provided ability that any numbers of students can take admission using registration form.

III. The analysis assumes conservative and uniform spread of values, this may not hold and more sophisticated tools for estimation should be used.

IV. Managing staff can delete the student record if the student wants to cancel his admission, via email .

V. Greater information about the user can be stored and the user may update his data status via email .

* **CONCLUSION:**

After all the rigorous sittings and hours of hard labour by all the team members, the projectis finally completed and has really shaped in a satisfying way. The application stands out inmost of its performance requirements and also has a very friendly and self-explaining userinterface.

The application is very practical and simple to run and use. In today’s time when maintaining backup is so much important that everyone needs to save and keep record of their data, this application can prove to be very handy to the user in terms of keeping track of teaching staff information ,students’ data, attendance. It also satisfies security concerns asit is based on username, password and one time verification code.We hope the application ends up giving a nice experience to every user and has no fatalerrors and any severe drawback.

* **References:**

I.“Author Guidelines”, *tutorialspoint.com/mongodb/mongodb\_php.htm*

II.“Database System Concepts Fifth Edition*”,* [*Avi Silberschatz*](http://www.cs.yale.edu/homes/avi)*,* [*Henry F. Korth*](http://www.lehigh.edu/%7Ehfk2/hfk2.html)*,* [*S. Sudarshan*](http://www.cse.iitb.ac.in/%7Esudarsha)