­A Project Report on

#### **Course Information Monitoring System**

Submitted in partial fulfillment of requirement

for the award of the degree

##### **MASTER of COMPUTER APPLICATIONS**

Of

###### Visvesvaraya Technological University

By

###### **Abhijith B. S. – 1PI13MCA01**

###### **Joshua Fernandes- 1PI13MCA31**



PES INSTITUTE OF TECHNOLOGY

100 Ft Ring Road, B.S.K 3rd Stage, Bangalore-85

**2015**

**PES INSTITUTE OF TECHNOLOGY**

**Department of MCA**

**100 Ft Ring Road, BSK 3rd Stage**

**Bangalore 85**

**2015**



**C E R T I F I C A T E**

This is to certify that the project entitled **“Course Information Monitoring System”** is a bonafide work carried out by **Abhijith B.S. 1PI13MCA01 & Joshua Fernandes 1PI13MCA31** submitted in partial fulfillment of the requirement of Fourth semester course work of MCA during the academic session Jan-May 2015.

**Guide Dr. Neelam Bawane**

Assistant Professor, Professor & Head

Department of MCA Department of MCA PES Institute of Technology PES Institute of Technology Bangalore Bangalore

**Examiners**

**1.**

**2**

**DECLARATION**

We **Abhijith B. S.** bearing University seat number **1PI13MCA01** and **Joshua Fernandes** bearing University Seat Number **1PI13MCA31** of 4th Semester, PESIT, hereby declare that the project entitled **“Course Information Monitoring System”** has been carried out by us under the supervision of Project Guide **Lolika Padmanabhan,** Assistant Professor, Department Of MCA, PES Institute Of Technology, Bangalore is to be submitted in partial requirements for the award of the Degree Of Master Of Computer Applications by **Visvesvaraya Technological University** during the academic year 2015. This report has been not submitted to any other organization/university for any award or degree certificate.

**Date:**

**Place: Bangalore Abhijith B. S.**

**Date:**

**Place: Bangalore Joshua Fernandes**

**ACKNOWLEDGEMENT**

We take this opportunity to express my profound sense of sincere and deep gratitude to all those who have contributed to the knowledge and experience that I have gained during my project work.

To begin with, I express my heartfelt gratitude to **Dr. K.S. Sridhar**, Principal, PES Institute of Technology, Bangalore for providing me an opportunity to take up this project.

I would also like to express my heartfelt thanks and regards to **Dr. Neelam Bawane**, Head of the Department of MCA, PES Institute of Technology, Bangalore, for her overwhelming support in my project.

I extend my profound gratitude to the project guide**, Lolika Padmanabhan**, Assistant Professor, Department of MCA, PES Institute of Technology, Bangalore for her valuable guidance, co-operation and suggestions and also his encouragement to complete this project.

Finally, I would like to thank my friends for all their support and encouragement during my project work.

**ABSTRACT**

The Course Information Management System(CIMS) will provide the head of Department, teachers and students an online application that will allow them to track the workflow of the syllabus. It will provide teachers and students with information on how much of the course has been completed and how much is left. It will be driven by three interfaces; one for the head of the department, who will be able to keep of track of every subject and teacher and how much progression has been made. A second interface for teachers to update and keep track of how much of the syllabus is complete, and a third interface for students to check how much is completed based on which semester they are in.

Table of Contents

[1. Introduction 1](#_Toc419806368)

[1.1 Purpose 1](#_Toc419806369)

[1.2 Product Perspective 1](#_Toc419806370)

[1.3 Product Functions 1](#_Toc419806371)

[**1.4** User Classes and Characteristics 1](#_Toc419806372)

[2. REQUIREMENTS ANALYSIS 2](#_Toc419806373)

[2.1 Operating Environment 2](#_Toc419806374)

[2.2 Design and Implementation Constraints 2](#_Toc419806375)

[2.3 User Documentation 2](#_Toc419806376)

[2.4 Assumptions and Dependencies 2](#_Toc419806377)

[3. SOFTWARE REQUIREMENTS SPECIFICATION 2](#_Toc419806378)

[3.1 Users 2](#_Toc419806379)

[3.2 Functional Requirements 3](#_Toc419806380)

[3.3 Non- Functional Requirements 4](#_Toc419806385)

[3.3.1 Performance Requirements 4](#_Toc419806386)

[3.3.2 Safety Requirements 4](#_Toc419806387)

[3.3.3 Security Requirements 4](#_Toc419806388)

[3.3.4 Software Quality Attributes 4](#_Toc419806389)

[3.4 Software and Hardware Requirement 5](#_Toc419806390)

[*3.4.1* Hardware Interface 5](#_Toc419806391)

[3.4.2 Software Interfaces 5](#_Toc419806392)

[3.4.3 Communications Interfaces 5](#_Toc419806393)

[4. ANALYSIS AND DESIGN 6](#_Toc419806394)

[4.1 Use Case 6](#_Toc419806395)

[4.2 ER Diagram 8](#_Toc419806396)

[4.3 DFD Diagram 9](#_Toc419806397)

[4.4 Data Dictionary 9](#_Toc419806398)

[5. IMPELEMENTATION 11](#_Toc419806399)

[5.1 Code 11](#_Toc419806400)

[5.2 Screen Shots 16](#_Toc419806401)

[6. SOFTWARE TESTING 20](#_Toc419806402)

[6.1 Testing Objectives: 20](#_Toc419806403)

[6.2 Testing Principles: 20](#_Toc419806404)

[6.3 Unit testing: 20](#_Toc419806405)

[6.4 Integration Testing: 20](#_Toc419806406)

[6.5 System testing: 20](#_Toc419806407)

[6.6 Test Cases: 21](#_Toc419806408)

[6.6.1 Login Page 21](#_Toc419806409)

[6.6.2 Add Faculty Page 21](#_Toc419806410)

[6.6.3 Add Subject Page 22](#_Toc419806411)

[6.6.4 Faculty Information 22](#_Toc419806412)

[6.6.5 Notification 22](#_Toc419806413)

[6.6.6 Add Chapters 23](#_Toc419806414)

[6.6.7 Course-info update 23](#_Toc419806415)

[7. CONCLUSION 24](#_Toc419806416)

[8. FUTURE ENHANCEMENTS 25](#_Toc419806417)

[9. BIBLIOGRAPHY 25](#_Toc419806418)

Table of Figures

[Figure 1 CMIS Use Case 6](file:///C:\Users\Joshua\Desktop\CMS\Report\REPORT.docx#_Toc419805905)

[Figure 2 Faculty Use Case 6](file:///C:\Users\Joshua\Desktop\CMS\Report\REPORT.docx#_Toc419805906)

[Figure 3 Student Use Case 7](file:///C:\Users\Joshua\Desktop\CMS\Report\REPORT.docx#_Toc419805907)

[Figure 4 HOD Use Case 7](file:///C:\Users\Joshua\Desktop\CMS\Report\REPORT.docx#_Toc419805908)

[Figure 5DFD 8](file:///C:\Users\Joshua\Desktop\CMS\Report\REPORT.docx#_Toc419805909)

[Figure 6 ER Diagram 9](#_Toc419805910)

[Figure 7 Login Screen Shot 16](#_Toc419805911)

[Figure 8 HOD Screen Shot 16](#_Toc419805912)

[Figure 9 Faculty Page Screen Shot 17](file:///C:\Users\Joshua\Desktop\CMS\Report\REPORT.docx#_Toc419805913)

[Figure 10 HOD faculty report Screen Shot 17](#_Toc419805914)

[Figure 11 Faculty Screen Shot 17](#_Toc419805915)

[Figure 12 Faculty Update Screen Sho 18](#_Toc419805916)

[Figure 13 Faculty Notification Screen Shot 18](#_Toc419805917)

[Figure 14 Student Screen Shot 19](#_Toc419805918)

[Figure 15 Student Subject Overall Screen Shot 19](#_Toc419805919)

# Introduction

## Purpose

The Course Management System(CMS) is a system which integrates itself with the educational department, automating the task of monitoring the progress carried out by the faculties and staff and helps the head of the department monitor the progress and help him/her make decisions and evaluate the staff performance.

## Product Perspective

The product to be developed is a first of its kind, envisioned and required by the department to help monitor the faculties’ progress which was not possible manually and was unfeasible to do manually.

This product is a self-contained and can be extended further to provided new services to the department.

This SRS describes the requirements of the system as a whole and each of the services provided by the product.

It also lists the interfaces to be implemented and their inter-connections.

## Product Functions

It will be driven by three interfaces;

1. One for the head of the department, who will be able to keep of track of every subject and teacher and how much progression has been made.
2. A second interface for teachers to update and keep track of how much of the syllabus is complete.
3. Third interface for students to check how much is completed based on which semester they are in.
4. It provides a facility for the faculties to publish notifications for the students, like the assignments and their deadlines.

## User Classes and Characteristics

**HOD-** this type of user will be able to track the progress of the faculties and run queries and generate report against the DB

**Faculty-** This type of user will be able to add/update/edit the chapters and units completed and publish notifications for the students.

**Students-** This type of user will be able to see the state of the completion of the Course topics.

# REQUIREMENTS ANALYSIS

## Operating Environment

Apache Server, Linux OS, MySQL DB

## Design and Implementation Constraints

1. GUI is only in English.
2. Login and password is used for the identification of users.
3. Only registered Faculties and HOD will be authorized to use the services to add and edit.
4. Any student will be able to view only.
5. Limited to HTTP/HTTPS.
6. This system works on single server.

## User Documentation

The Documentation of the project and the manual will be provided with the software along with the deployment.

## Assumptions and Dependencies

The project assumes that the faculties will maintain up-to-date information about the course progress that they make.

The project depends on the Apache server and the server up time to make the resource available 99.9%

# SOFTWARE REQUIREMENTS SPECIFICATION

## Users

* + 1. **Head Of the Department**

Allow the hod to select faculty member that will display overall progress of the teacher with respect to every subject. In the same interface will allow them to select a semester along with the subject to keep track of specific subject. Will also include a button to refresh the database for the next semester. An additional interface for the hod will provide a view of the test portion for each semester with respect to the test number.

* + 1. **Faculty**

Will present the faculty member with its overall report in the first page, along with the ability to either add a new subject, edit a subject, update current date’s progress, set test portions or select particular semester/subject.

The Add new subject will allow the faculty the ability to add a subject based on semester and subject name and provide texts boxes with chapter name and number of hours.

The edit a subject will allow the faculty to edit either the number of hours or the chapter description along with feature to add more chapters or delete existing ones.

The update current date’s progress will allow them to update what chapter was completed today and how many hours was taken for which semester/subject.

Set test portions will allow them to select chapter from a list of completed portions to include in a particular test.

Selecting a particular semester/subject will provide them a view of their current progress; by generating reports of how much time has passed and how much progress has been made.

* + 1. **Student**

Students will be able to enter their semester and track all the subjects progress and view Test portions for the subject if available.

## Functional Requirements

### Ability for the HOD to monitor and generate reports.

The HOD will be able to keep of track of every subject and teacher and how much progression has been made. And he/she will be able to generate reports against the database which helps her in decision making and in evaluation of faculties.

### The Faculty should be able to update progress and manipulate Course information of the subject she handles

The Faculty should be able to add Chapters and units under them and manipulate them which are reflected in the DB

### **The system should provide an interface for students to check their course progress corresponding to their Semester.**

The Student enters their semester and section and should be provided with an overall view of their semester, followed by subject wise view.

### The system provides a facility for the faculties to publish notifications for the students, like the assignments and their deadlines.

The Faculty should be able to generate notification reminders on their interface which should reflect on the students interface.

## Non- Functional Requirements

### Performance Requirements

(1) Static numerical requirements:

        (a)  The number of terminals to be supported: 20+

        (b)  The number of simultaneous users to be supported: 20+

        (c)  Number of files and records to be handled: 10

        (d)  Sizes of tables and files: 100mb

(2)  The numbers of transactions and tasks will be 5 per second and 95% of the transactions shall be processed in less than 1 s on a normal workload conditions and the numbers of transactions and tasks will be 20 per second and 95% of the transactions shall be processed in less than 5 s on a peak workload conditions

### Safety Requirements

Data will be transmitted with the use HTTP protocols and use inbuilt browser security measures.

### Security Requirements

Users will be authenticated based on provided login details. Faculty members will not be able to track other faculty member’s progress or view reports. Highest clearance will be provided to the hod.

### Software Quality Attributes

**Availability**

Is available anywhere in campus when connected to the intranet. Will display a 404 page in case of loss of connection. No data will be lost and no commits will be made until complete transaction is made to the server.

**Usability**

The system is easy to handle and navigates in the most expected way with no delays. The system program reacts accordingly and transverses quickly between its states.

## Software and Hardware Requirement

### Hardware Interface

|  |  |  |  |
| --- | --- | --- | --- |
| **Client Side** | | | |
|  | Processor | RAM | Disk Space |
| **Internet Explorer – 6 or above** | All Intel or AMD - 1 GHZ | 256 MB | 100 MB |

|  |  |  |  |
| --- | --- | --- | --- |
| **ServerSide** | | | |
|  | Processor | RAM | Disk Space |
| **Apache Server** | All Intel or AMD - 2 GHZ | 2GB | 1 GB |
| **MySql** | 512 MB | 500MB(Excluding Data Size) |

### Software Interfaces

**Client on Internet**

Web Browser, Operating System (any)

**Client on Intranet**

Web Browser, Operating System (any)

**Web Server**

Apache (WAMP, XAMP), Operating System (any)

**Data Base Server**

MySQL, Operating System (any)

**Development End**

LAMP (Linux, Apache, HTML, XML, AJAX, PHP), MySQL, OS (Windows),

### Communications Interfaces

1. It uses the any Modern Web browser capable of running Java-script on client side.

2. It makes use of HTTP Protocol.

3. The system services should be accessible throughout the Local Area Network which is available to each of the faculties on their computer systems.

# ANALYSIS AND DESIGN

## Use Case

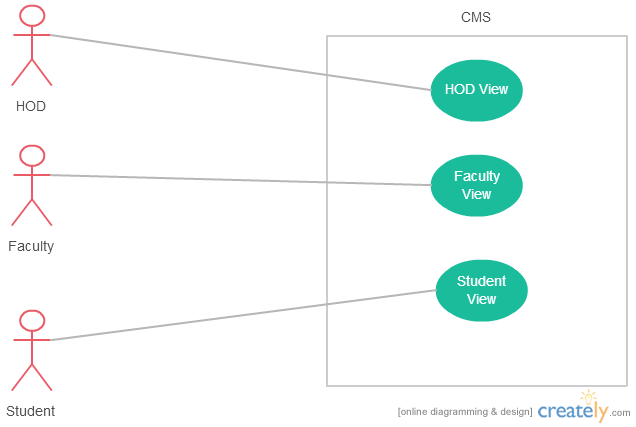


Figure CMIS Use Case

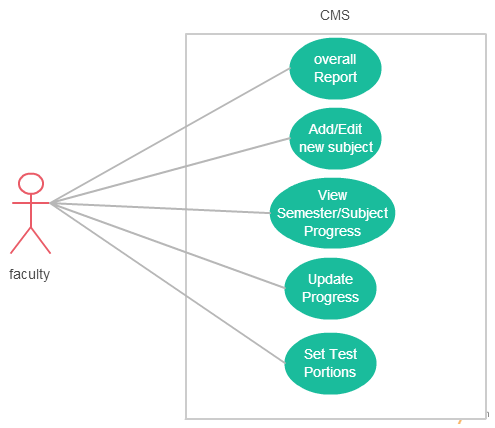


Figure Faculty Use Case

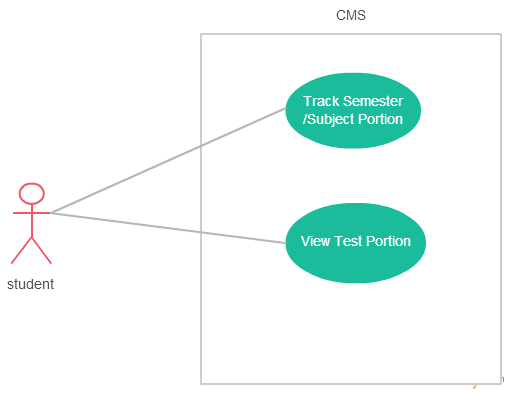


Figure Student Use Case

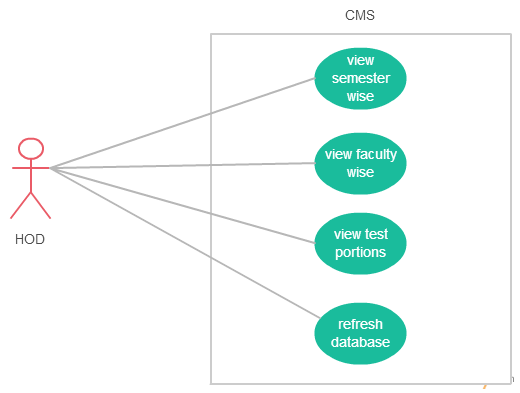


Figure HOD Use Case

## ER Diagram

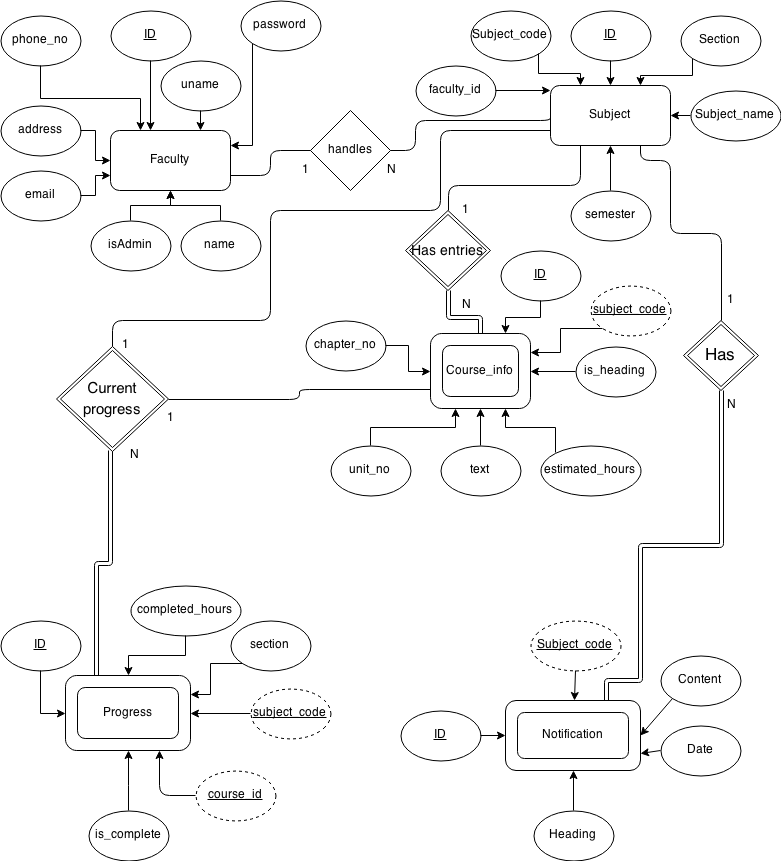


Figure ER Diagram

## DFD Diagram

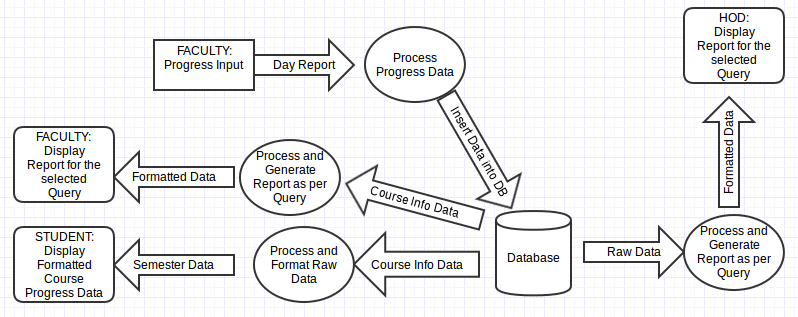


Figure DFD Diagram

## Data Dictionary

**Course\_info**

|  |  |  |
| --- | --- | --- |
| **Name** | **Data type** | **Comment** |
| Id | int(11) | Primary Key |
| chap\_no | int(2) |  |
| unit\_no | int(2) |  |
| Text | varchar(256) |  |
| est\_hrs | int(3) |  |
| is\_heading | int(1) | Bianary |
| sub\_code | varchar(20) |  |

Table course\_info data dictionary

**Faculty**

|  |  |  |
| --- | --- | --- |
| **Name** | **Data type** | **Comment** |
| Id | int(11) | Primary Key |
| Uname | varchar(30) | Unique |
| Pword | varchar(18) |  |
| Name | varchar(30) |  |
| isAdmin | int(2) |  |
| email | varchar(50) |  |
| address | varchar(200) |  |
| phone\_no | varchar(15) |  |

Table Faculty data dictionary

**Notification**

|  |  |  |
| --- | --- | --- |
| **Name** | **Datatype** | **Comment** |
| id | int(5) | Primary Key |
| heading | varchar(40) |  |
| content | varchar(500) |  |
| date | date |  |
| subject\_code | varchar(15) |  |

Table notification data dictionary

**Progress**

|  |  |  |
| --- | --- | --- |
| **Name** | **Datatype** | **Comment** |
| id | int(5) | Primary Key |
| completed\_hrs | int(3) |  |
| is\_complete | int(3) | Binary |
| subject\_code | varchar(20) |  |
| section | varchar(3) |  |
| course\_id | int(5) |  |

Table Progress Data Dictionary

**Subject**

|  |  |  |
| --- | --- | --- |
| **Name** | **Datatype** | **Comment** |
| id | int(5) | Primary Key |
| subject\_code | varchar(15) | Unique |
| section | varchar(4) |  |
| subject\_name | varchar(40) |  |
| faculty\_id | int(5) |  |
| semester | int(3) |  |

Table Subject Data Dictionary

# IMPELEMENTATION

## Code

**Connection.php**

<?php

$servername = "localhost";

$username = "root";

$password = "";

$dbname = "cms";

$conn = new mysqli($servername, $username, $password, $dbname);

?>

**Login.php**

session\_start();

//the below code block is required as it controls which user can access the pages,please don't remove it

if(isset($\_SESSION['unames'])) //every page checks if logged in ,and if not then go to login page , we are already in login page so no else condition

{

if($\_SESSION['isAdmin']==1) // if admin, then go to admin page

{

header('Location: admin.php');

}

if($\_SESSION['isAdmin']==0) //if faculty go to faculty page

{

header('Location: faculty.php');

}

}

$pluname="Enter faculty ID";

$plpassword="Enter Password";

if(isset($\_POST['logs'])){

include 'connection.php';

//conn object is initialized in connection.php

if ($conn->connect\_error) { //Check connection

die("Connection failed: " . $conn->connect\_error);

}

$pluname=$\_POST['uname'];

$plpassword="";

$sql = "SELECT name,pword,isAdmin FROM faculty where uname='".$\_POST['uname']."' ";

$result = $conn->query($sql);

if ($result->num\_rows > 0) {

$row = $result->fetch\_assoc();

//echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";

if($\_POST['pword'] == $row["pword"])

{

$\_SESSION["unames"] = $\_POST['uname'];

$\_SESSION["name"]=$row["name"];

if($row["isAdmin"]==1)

{

$\_SESSION["isAdmin"] = 1;

header('Location: admin.php');

//http\_redirect("admin.php", array("uname" => $\_POST['uname']), false, HTTP\_REDIRECT\_PERM);

}

else{

$\_SESSION["isAdmin"] = 0;

header('Location: faculty.php');

}

}

else{

$pluname='/" value="'.$\_POST['uname'].'"';

$plpassword="Invalid Password";

}

} else {

$pluname="Invalid Username";

}

$conn->close();

}

?>

**Inserting Data**

//Code to insert the notification if it was submitted by user

if(isset($\_POST['submit']) && $\_POST['submit']=="Post Notification")

{

if(!empty($\_POST['title']) && !empty($\_POST['description']) && !empty($\_POST['subject\_code'])){

//got notification request, process it

$sql="INSERT INTO notification(heading,content,date,subject\_code) VALUES('".$\_POST['title']."','".$\_POST['description']."',CURDATE(),'".$\_POST['subject\_code']."');";

$result=$conn->query($sql);

if(!$result) {echo "<h2>Error...Unable to post Notification...</h2>";}

}

}

//Code to delete the notification requested by user

if(isset($\_POST['delete']) && $\_POST['delete']=="Delete")

{

if(!empty($\_POST['id'])){

//delete the notification,process it

$sql="DELETE FROM notification WHERE id=".$\_POST['id'].";";

if(!($conn->query($sql))) {echo "<h2>Error...Unable to delete Notification...</h2>";}

}

}

**View Data**

<?php

echo '<form action="" method="GET">';

echo '<input type=hidden name="subject\_id" value="'.$subjectid.'">'; //setting hidden data of subject id and section and code

echo '<input type=hidden name="subject\_code" value="'.$subjectcode1.'">';

echo '<input type=hidden name="section\_sel" value="'.$subjectsec1.'">';

//fetching course information data

$sql="select c.id,c.chap\_no,c.unit\_no,c.`text`,c.est\_hrs,c.is\_heading,p.completed\_hrs,p.is\_complete from course\_info c left join subject s on c.sub\_code = s.subject\_code

left join progress p on p.course\_id=c.id and p.section=s.section

where s.subject\_code='".$subjectcode1."'

and s.section='".$subjectsec1."' ORDER BY c.chap\_no,c.unit\_no";

$result = $conn->query($sql);

while($row = $result->fetch\_assoc())

{

if($row["is\_heading"]==1)

echo '<tr class="odd">';

else

echo '<tr class="even">';

echo '<td>'.$row["chap\_no"].'</td>';

echo '<td>'.$row["unit\_no"].'</td>';

echo '<td>'.$row["text"].'</td>';

if($row["is\_heading"]==0){

echo '<td>'.$row["est\_hrs"].'</td>';

echo '<td><input type=text name="'.$row["id"].'" value="'.$row["completed\_hrs"].'" size=2 /></td>';

}

else

echo "<td></td><td></td>";

if($row["is\_heading"]==1)

echo '<td></td>';

else

{

echo '<td><button name="to\_add\_id" value="'.$row["id"].'" type="submit">Add hour</button></td>';

//echo '<td><input type=submit value="Add Hour" name="add'.$row["id"].'" </td>';

echo '<td><button name="to\_update" value="'.$row["id"].'" type="submit">Update hour</button></td>';

}

echo '</tr>';

}

echo '</form>';

?>

**Logout**

<?php

session\_start(); // start current session

session\_destroy(); //destroy all session data

header('Location: index.php');

?>

**Charts**

<?php

include\_once('./connection.php');

//fetching main information

if ($conn->connect\_error) { //Check connection

die("Connection failed: " . $conn->connect\_error);

}

function printPie($element)

{

$subject\_code=$\_GET["subject\_code"];

$section=$\_GET["section\_sel"];

global $conn;

$query="select sum(c.est\_hrs) as estimated,sum(p.completed\_hrs) as completed

from course\_info c left join subject s on c.sub\_code = s.subject\_code

left join progress p on p.course\_id=c.id and p.section=s.section

where s.subject\_code='".$subject\_code."'

and s.section='".$section."'

and c.is\_heading = 0;";

$result = $conn->query($query);

$row = $result->fetch\_assoc();

$estimated=$row["estimated"];

$completed=$row["completed"];

$remaining=intval($estimated)-intval($completed);

$r=array("Remaining",$remaining,"#b7b3a3","#6b6964");

$c=array("Completed",$completed,"#20b31c","#11720d");

$values=array($r,$c);

echo " <script> var pieData = [ ";

foreach($values as $k=>$v)

{

echo "{value: '".$v[1]."',label: '".$v[0]."',color: '".$v[2]."',highlight: '".$v[3]."'},";

}

echo "];";

echo " window.onload = function(){

var context = document.getElementById('".$element."').getContext('2d');

var skillsChart = new Chart(context).Pie(pieData);

var title = document.getElementById('PieTitle');

title.innerHTML=\"Estimated hours: ".$estimated." Completed: ".$completed." Remaining: ".$remaining."\";

}

</script>";

}

?>

## Screen Shots

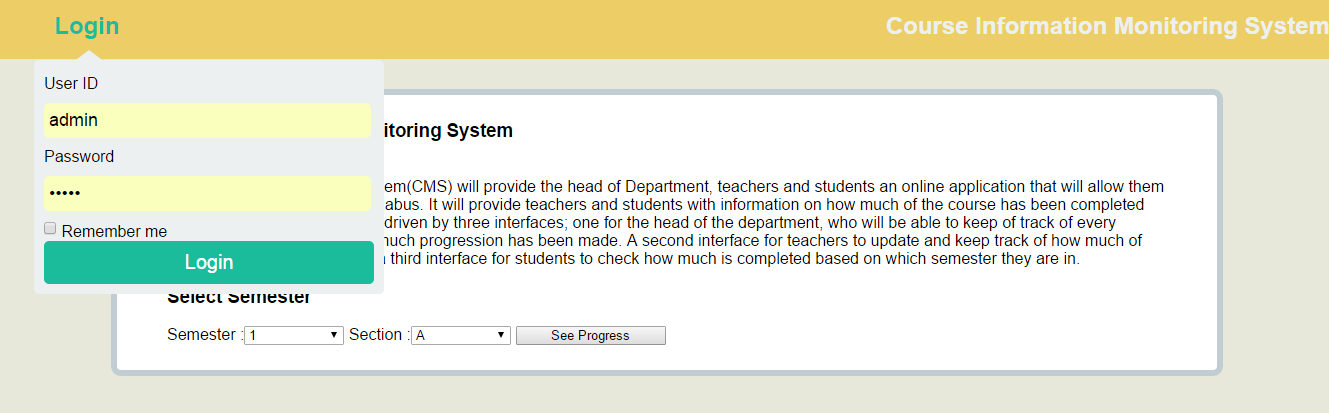


Figure Login Screen Shot

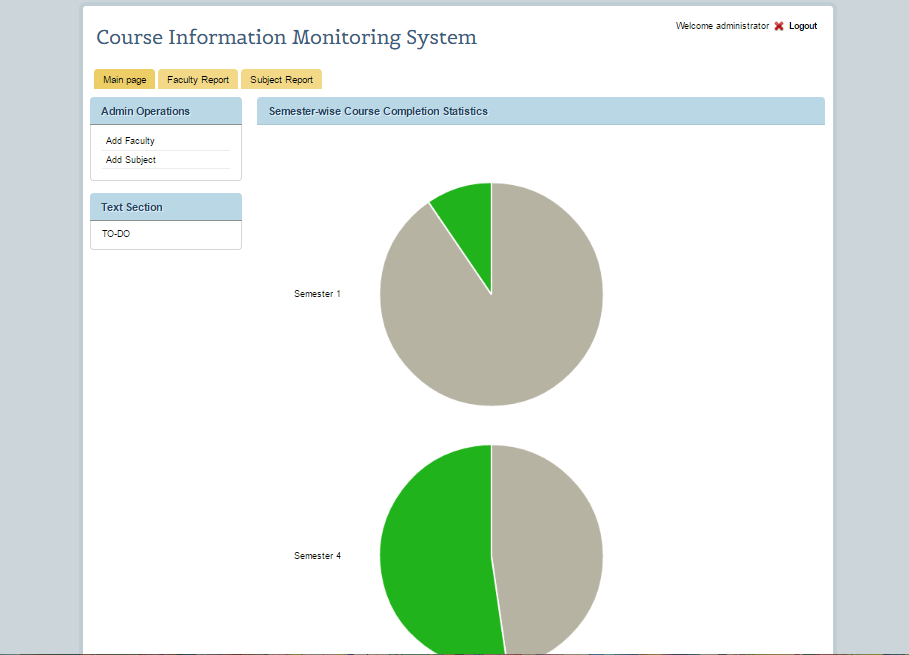
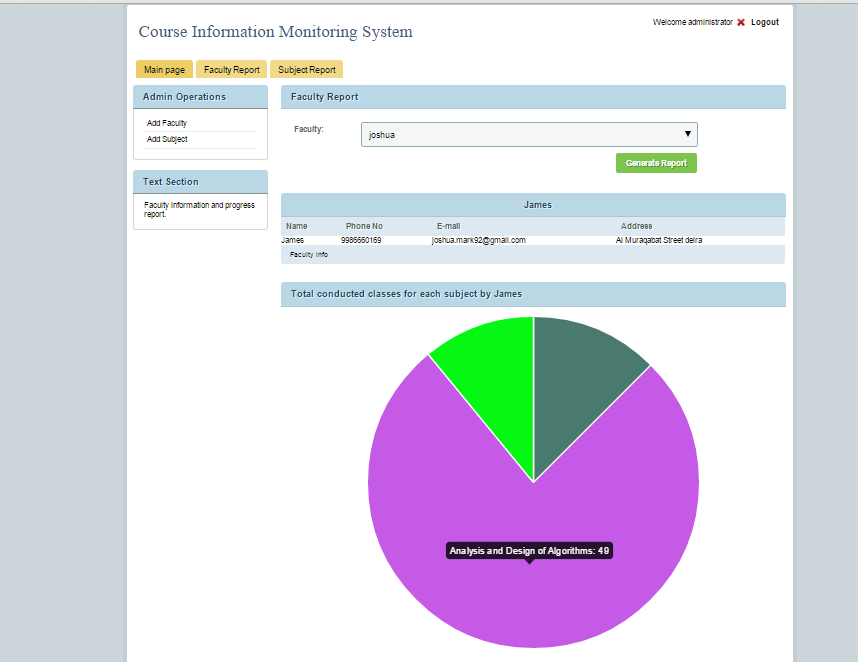


Figure HOD Screen Shot



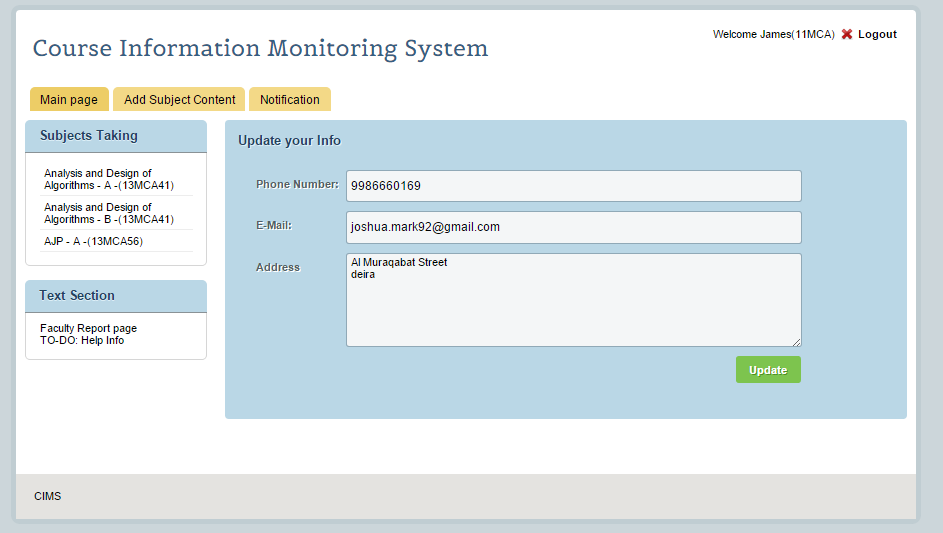
Figure HOD faculty report Screen Shot

Figure Faculty Page Screen Shot

Figure Faculty Screen Shot

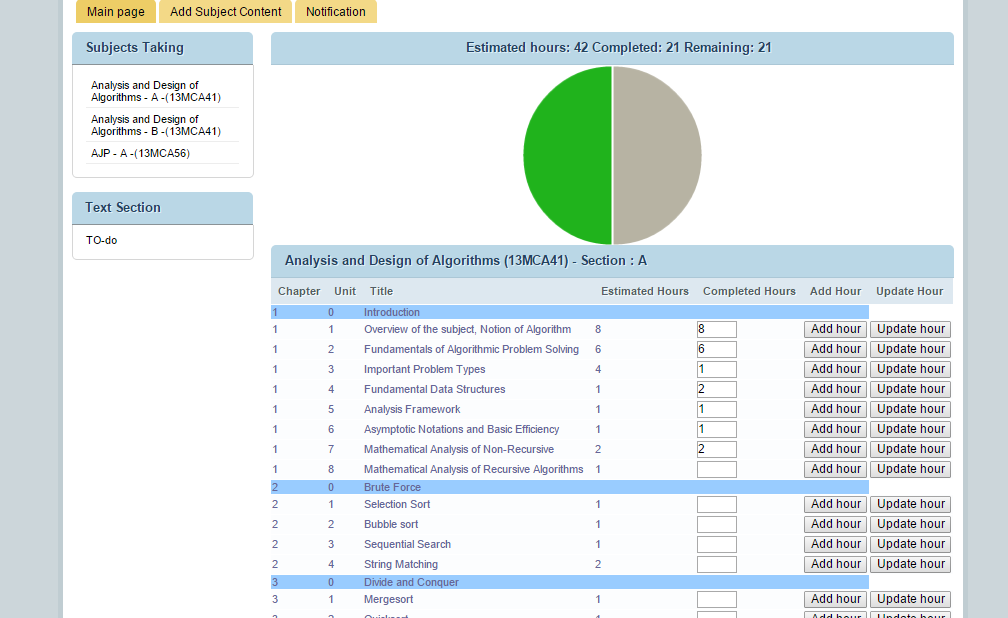


Figure Faculty Update Screen Sho

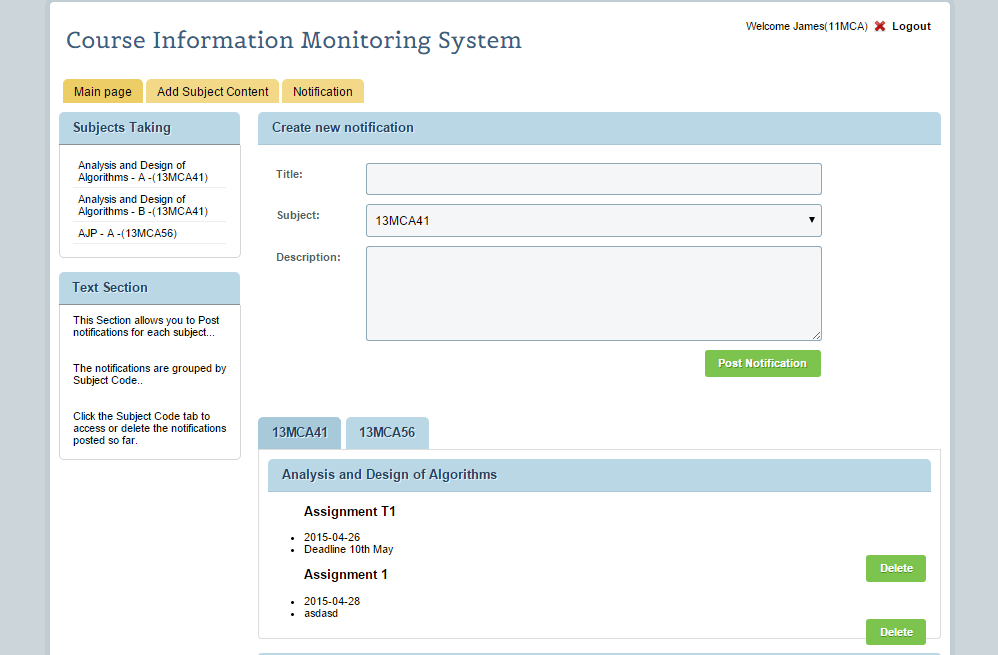


Figure Faculty Notification Screen Shot

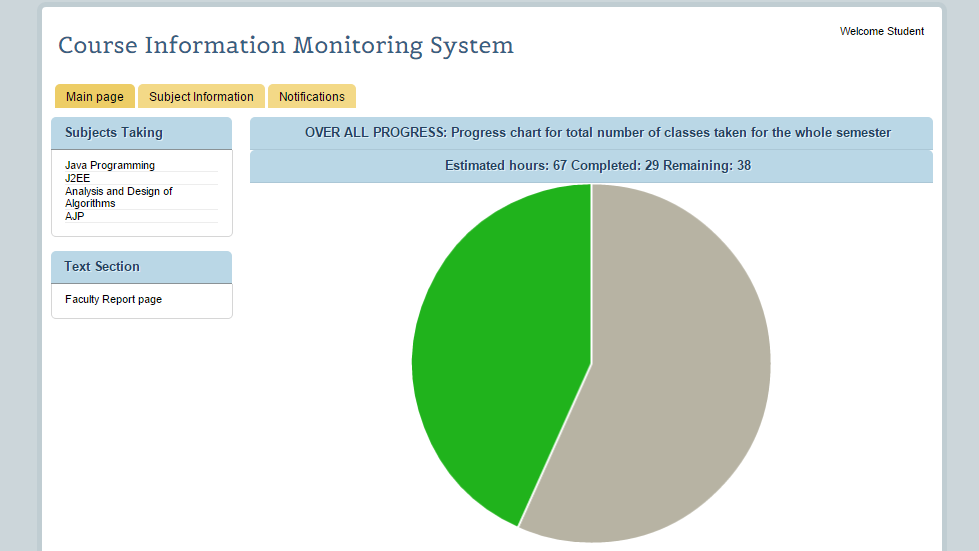


Figure Student Screen Shot

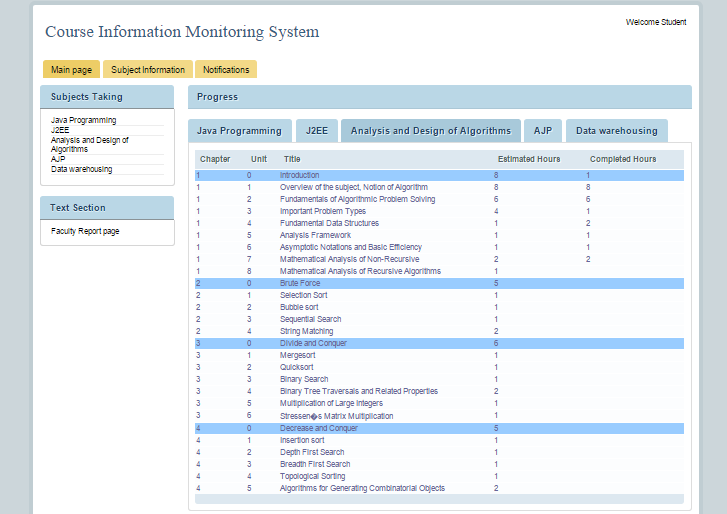


Figure Student Subject Overall Screen Shot

# SOFTWARE TESTING

## Testing Objectives:

Testing is a process of executing a program with the intent of finding an error A good test case is one that has a probability of finding an as yet undiscovered. A successful test is one that uncovers an undiscovered error.

## Testing Principles:

* All tests should be traceable to end user requirements
* Tests should be planned long before testing begins
* Testing should begin on a small scale and progress towards testing in large
* Exhaustive testing is not possible

## Unit testing:

Unit testing is essential for the verification of the code produced during the coding phase and hence the goal is to test the internal logic of the modules. Using the detailed design description as a guide, important paths are tested to uncover errors with in the boundary of the modules. These tests were carried out during the programming stage itself.

## Integration Testing:

Integration testing focuses on unit tested modules and build the program structure that is dictated by the design phase.

## System testing:

System testing tests the integration of each module in the system. It also tests to find discrepancies between the system and its original objective, current specification and system documentation. The primary concern is the compatibility of individual modules. Entire system is working properly or not will be tested here, and specified path ODBC connection will correct or not, and giving output or not are tested here these verifications and validations are done by giving input values to the system and by comparing with expected output. Top-down testing implementing here.

## Test Cases:

Test cases are derived to ensure that all statements in the program have been executed at least once during testing and that all logical conditions have been executed. Using White-Box testing methods, the software engineer can drive test cases that guarantee that logical decisions on their true and false sides. Exercise all logical decisions on their true and false sides.

### Login Page

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.NO.** | **Test** | **User Input** | **Expected Result** | **Actual Result** | **Status** |
| 1. | Check login | Valid username and password | Login to user’s homepage | If admin, go to admin.php, if faculty, go to faculty.php | Pass |
| 2 | Valid username and invalid password | Print “Incorrect Password” | Set’s placeholder of password an invalid | Pass |
| 3 | Invalid username | Print “Invalid username” | Sets placeholder of username as invalid | Pass |

### Add Faculty Page

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.NO.** | **Test** | **User Input** | **Expected Result** | **Actual Result** | **Status** |
| 1. | Check faculty form input | Null name | Print “Enter a name” | Java alert “Enter a name” | Pass |
| 2 | Invalid name(special characters) | Prints “Invalid name” | Java alert “Enter a Valid name” | Pass |
| 3 | Null username | Print “Enter a user name” | Java alert “Enter a user name” | Pass |
| 4 | Null Password | Print “Enter a Password” | Java alert “Enter a Password” | Pass |
| 5 | Password length less than 8 | Print “Password is too short” | Java alert “Please Enter a password with 8 Characters” | Pass |
| 6 | Null repeat Password | Print “Enter the repeat Password” | Java alert “Enter the repeat Password” | Pass |
| 7 | Repeat password does not match | Print “Passwords do not match” | Java Alert ”Passwords do not match” | Pass |

### Add Subject Page

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.NO.** | **Test** | **User Input** | **Expected Result** | **Actual Result** | **Status** |
| 1. | Check Subject form input | Null Subject code | Print “Input subject code” | Java alert “Enter a subject code” | Pass |
| 2 | Subject code with special characters | Print “Invalid subject code” | Java alert “Enter valid subject code” | Pass |
| 3 | Null Subject name | Print “Input subject name” | Java alert “Enter a subject name” | Pass |

### Faculty Information

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.NO.** | **Test** | **User Input** | **Expected Result** | **Actual Result** | **Status** |
| 1. | Check Faculty Information form input | Null Phone number | Print “Input Phone number” | Java alert “Enter your phone number” | Pass |
| 2 | Invalid phone number  (characters) | Print “Invalid phone number” | Java alert “Enter valid phone number” | Pass |
| 3 | Null e-mail | Print “Input e-mail” | Java alert “Enter your e-mail” | Pass |
| 4 | Invalid e-mail | Print “Input valid e-mail” | Java alert “Enter valid e-mail” | Pass |
| 5 | Null address | Print “Input your address” | Java alert “Enter your address” | Pass |

### Notification

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.NO.** | **Test** | **User Input** | **Expected Result** | **Actual Result** | **Status** |
| 1. | Check Notification creation form | Null Title | Print “Input Title” | Java alert “Enter a title” | Pass |
| 2 | Null Description | Print “Input Description” | Java alert “Enter a description” | Pass |

### Add Chapters

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.NO.** | **Test** | **User Input** | **Expected Result** | **Actual Result** | **Status** |
| 1 | Check Chapter creation form | Null Chapter Title | Print “Input Chapter Title” | Java alert “Enter a chapter title” | Pass |
| 2 | Null Unit name | Print “Input Unit name” | Java alert “Enter the unit name” | Pass |
| 3 | Non-integer estimated hours | Print “Invalid unit number” | Java alert “Invalid estimated hours” | Pass |

### Course-info update

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.NO.** | **Test** | **User Input** | **Expected Result** | **Actual Result** | **Status** |
| 1 | Check completed hours form input | Null value | Print “Input a integer completed hours” | Java alert “Enter integer for completed hours” | Pass |
| 2 | Non-integer value | Print “Input a integer completed hours” | Java alert “Enter integer for completed hours” | Pass |

# CONCLUSION

The foundation idea of the project was to provide a convenient and unpretentious platform for faculties and the Head of the Department to synchronize amongst them self, using an online interface to share real time information about the course progress made by the faculty. This project also helps faculty to organize their strategy for the semester, in order to complete the course’s topics in time.

This project uses data mining and data visualization techniques to provide a graphical informational tool for faculties and the Head of the Department to coordinate amongst them self’s.

This project “Course Information Management System” exploits the features provided by the various web technologies like, PHP v5, MySQL v5, Apache server v2 in the backend and HTML v5, Ajax and JavaScript in the front end.

This particular project deals with the problems on managing the progress of course under a department and avoids the problems which occur when carried manually. Identification of the drawbacks of the existing system leads to the designing of computerized system that will be compatible to the existing system with the system which is more user friendly, GUI oriented and provides real time data analysis from the database.

# FUTURE ENHANCEMENTS

This project provides a base platform and an idea which can be taken forward to involve more modules to automate and digitize the tasks which are normally carried out manually.

One such module would allow the faculty to select the subjects already completed and post them as the notification which would be included in the upcoming test’s portions rather than posting it on bulletin boards manually.

Another feature would be to create a module which would alert the faculty if their estimated/completed hours exceed their allotted hours and prompt the faculty to make a request to the head of the department to allocate extra hours for her particular subject.

A feature could be added to the student module to give their feedback about the faculty after the end of the semester.

In regard of data mining and visualization, more reports and visualization technique like bar, line graphs could be generated.

# BIBLIOGRAPHY

1. www.w3schools.com

2. www.php.net

3. en.wikipedia.org/

4. www.hotscripts.com/category/php/

5. www.apache.org/

6. www.mysql.com/

7. www.stackoverflow.com

8. www.charts-js.com – chart.js library

9. Database Design – Navathe