

## TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NO.
	ABSTRACT	iv
	LIST OF TABLES	2
	LIST OF FIGURES	3
1	INTRODUCTION	4
	1.1 Project Summary	4
	1.2 Purpose	5
	1.3 Scope	6
	1.4 Platform	6
2	TECHNOLOGY OVERVIEW	7
	2.1 PHP	7
	2.2 MySQL	7
	2.3 Apache	8
	2.4 XAMPP	8
3	SYSTEM ARCHITECTURE DESIGN	11
	3.1 System Design	11
	3.2 Data Design	11
	3.3 Functional Design	12
4	SYSTEM DIAGRAM	13
	4.1 Flow-chart Diagram	13
	4.2 Context Diagram	13
	4.3 Data-Flow Diagram	14
	4.3.1 Level-1	14
	4.3.2 Level 2	15
	4.4 E-R Diagram	16
	4.5 Use-Case	17
5	SOURCE CODE	18
	5.1 Index.php	18
	5.2 admin.php	18
	5.3 lecture.php	21
	5.4 logout.php	24

6	SCREENSHOTS	25
	6.1 Main Page	25
	6.2 Student Page	26
	6.3 Admin/ Lecture Page	27
	6.4 Contact Page	28
7	CONCLUSION	29
8	REFERENCES	30

## LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
2.1	Installing XAMPP SERVER	09
2.2	Creating database in XAMPP SERVER	09
2.3	Inserting database file in XAMPP SERVER	10
3.1	System Architecture Diagram	11
3.2	Data Design	11
3.3	Functional Design	12
4.1	System Flow Chart Diagram	13
4.2	Context Diagram	14
4.3	Level-1 DFD diagram for login	14
4.4	level-2 DFD diagram for marks	15
4.5	level-2 DFD diagram for Admin	15
4.6	DFD diagram for modifying subject	16
4.7	E-R diagram for student and course	16
4.8	Use- Case diagram for system	17
4.9	Use-Case diagram for teacher login	17
6.1	Home Page	25
6.2	Student Search Page	26
6.3	Admin/ Lecturer Login Page	27
6.4	Contact page	28

# INTRODUCTION

## 1.1 PROJECT SUMMARY

A Student Information System (SMS) for School/ College Management Information System (SIS) is a System that manages the records of student regarding admission and examination.

A Student Information System (SIS) is designed to help collages for management of School/ College student. Viewing student data, managing admission, managing seats, semester, faculty, examination, subject management, result and related issues are made simple and easy.

Student management function involves-

- Manage new admission and enrolment
- Searching student record
- Roll number generation
- student Basic Information
- Manage faculty
- Manage semester and year
- Result management
- Subject management

In SMS, every teacher has a Login ID and Password. Also all the users have different permission rights to access the applications. These rights are Dynamic and can be changed.

There are two main roles in the system. Admin and Teacher. Admin has complete access to the whole system while Teacher is the role that is responsible for updating the student field related to it.

The Admin role can be as follow:

- Introduce new category, course etc.
- Manage faculties
- Manage subjects
- Management of semester
- Generation of student roll number
- 

The teacher role can:

- Search student
- Result etc.

Now when the user with the particular role Logs on he can see only those pages which are allowed to them.

## **1.2 PURPOSE**

The project is about to handle all the information of the student regarding admission and examination. Also it manages resources which were managed and handled by manpower previously. The main purpose of the project is to integrate distinct sections of the organization into consistent manner so that complex functions can be handled smoothly by any technical or non-technical persons.

The project aims at the following matters:

- Automation of admission and enrolment as per board, quota, category and available seats.
- Assistance in decision-making.
- To manage information of student, faculty and courses.
- Consistently update information of all the students.
- Reports- To gather all the related information about any application of the HRMS.

All the above-mentioned matters are to be incorporated in the application along with some additional requirements.

The main purpose of the Admin Module is to introduce new things and configure important aspects. For e.g. only admin is authorized to introduce subject, category etc. and only admin is allowed to configure exam. So the master screens for all these are visible to only admin role. This is done by the Admin Module. It also can create the users and Physical and Logical Locations. Thus the main purpose of the Admin Module is to managing the dynamic working of the system.

### **1.3 SCOPE**

The scope of the project includes the following:

- Any college can use this system as it is not client centric.
- All admission and examination related work for the student can be done using this system.
- Deliver Electronic Workplace
- Application Support & Maintenance after deployment to production
- The Admin Module can be reused for projects as well which have many users with different rights. Hence it is reusable.

### **1.4 PLATFORM**

**Operating Systems:** Microsoft Windows

**Technologies Used:**

- Front End: HTML, CSS, JavaScript
- Web designing language: PHP
- RDBMS(Back end): MySQL

**Software Requirements:**

- PHP 5.0
- APACHE HTTP Server
- Microsoft Windows or Linux

# TECHNOLOGY OVERVIEW

The technology selected for implementing Student Information Management System is PHP/ MYSQL, Apache is used as the HTTP server. The development was done in a 'windows' environment using Notepad++.

## 2.1 PHP

PHP is a general-purpose scripting language that is especially suited to server-side web development where PHP generally runs on a web server. PHP code is embedded into the HTML source document. Any PHP code in a requested file is executed by the PHP runtime, usually to create dynamic web page content. It can also be used for command-line scripting and client-side GUI applications. PHP can be deployed on many web servers and operating systems, and can be used with many relational database management systems (RDBMS). It is available free of charge, and the PHP Group provides the complete source code for users to build, customize and extend for their own use.

## 2.2 MySQL

MySQL is a relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases. MySQL is a popular choice of database for use in web applications and is an open source product. The process of setting up a MySQL database varies from host to host, however we will end up with a database name, a user name and a password. Before using our database, we must create a table. A table is a section of the database for storing related information. In a table we will set up the different fields which will be used in that table. Creating a table in phpMyAdmin is simple, we just type the name, select the number of fields and click the 'go' button. We will then be taken to a setup screen where you must create the fields for the database. Another way of creating databases and tables in phpMyAdmin is by executing simple SQL statements. We have used this method in order to create our database and tables.

## **2.3 Apache**

The Apache HTTP Server is a web server software notable for playing a key role in the initial growth of the World Wide Web. In 2009 it became the first web server software to surpass the 100 million web site milestone. Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation. Since April 1996 Apache has been the most popular HTTP server software in use. As of November 2010 Apache served over 59.36% of all websites and over 66.56% of the first one million busiest websites.

## **2.4 XAMPP**

XAMPP is a small and light Apache distribution containing the most common web development technologies in a single package. Its contents, small size, and portability make it the ideal tool for students developing and testing applications in PHP and MySQL. XAMPP is available as a free download in two specific packages: full and lite. While the full package download provides a wide array of development tools, XAMPP Lite contains the necessary technologies that meet the Ontario Skills Competition standards. The light version is a small package containing Apache HTTP Server, PHP, MySQL, phpMyAdmin, OpenSSL, and SQLite.

### **Obtaining and Installing XAMPP**

On the website, navigate and find the Windows version of XAMPP and download the self-extracting ZIP archive. After downloading the archive, run and extract its contents into the root path of a hard disk or USB drive. For example, the extract path for a local Windows installation would simply be C:\. Next we will test if the server is running correctly by opening an internet browser and typing `http://localhost/` into the address bar. If configured correctly, we will be presented with a screen similar to that of the one below. In order to stop all Apache processes we do not close the running terminal application, but instead



run another batch file in the XAMPP lite directory called apache\_stop.bat.



Fig (2.1) Installing XAMPP SERVER

## Creating a Database and Inserting Data

Now that we have run and tested Apache and PHP, the next step is running MySQL and creating a database and table which will hold information to be used by our website. The XAMPP package contains an application called phpMyAdmin which allows developers to administer and maintain MySQL databases. We will be using phpMyAdmin to create a database and table, and enter test data. Before testing phpMyAdmin, make sure that both Apache and MySQL are running by opening their respective batch files: apache\_start.bat and mysql\_start.bat. Along with Apache and MySQL running in the background, we type `http://localhost/phpMyAdmin/` into the web browser. If successful we will be presented with a phpMyAdmin start page similar to the one shown below.

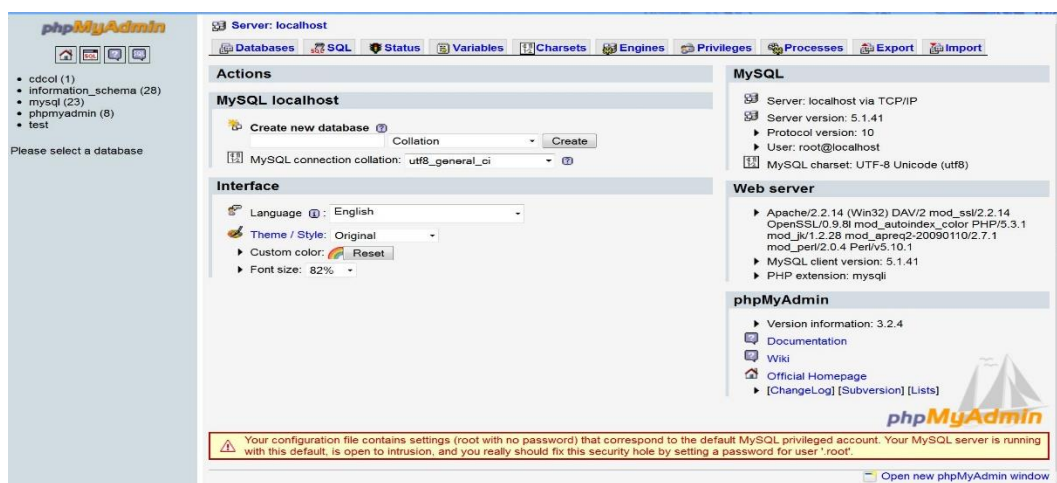
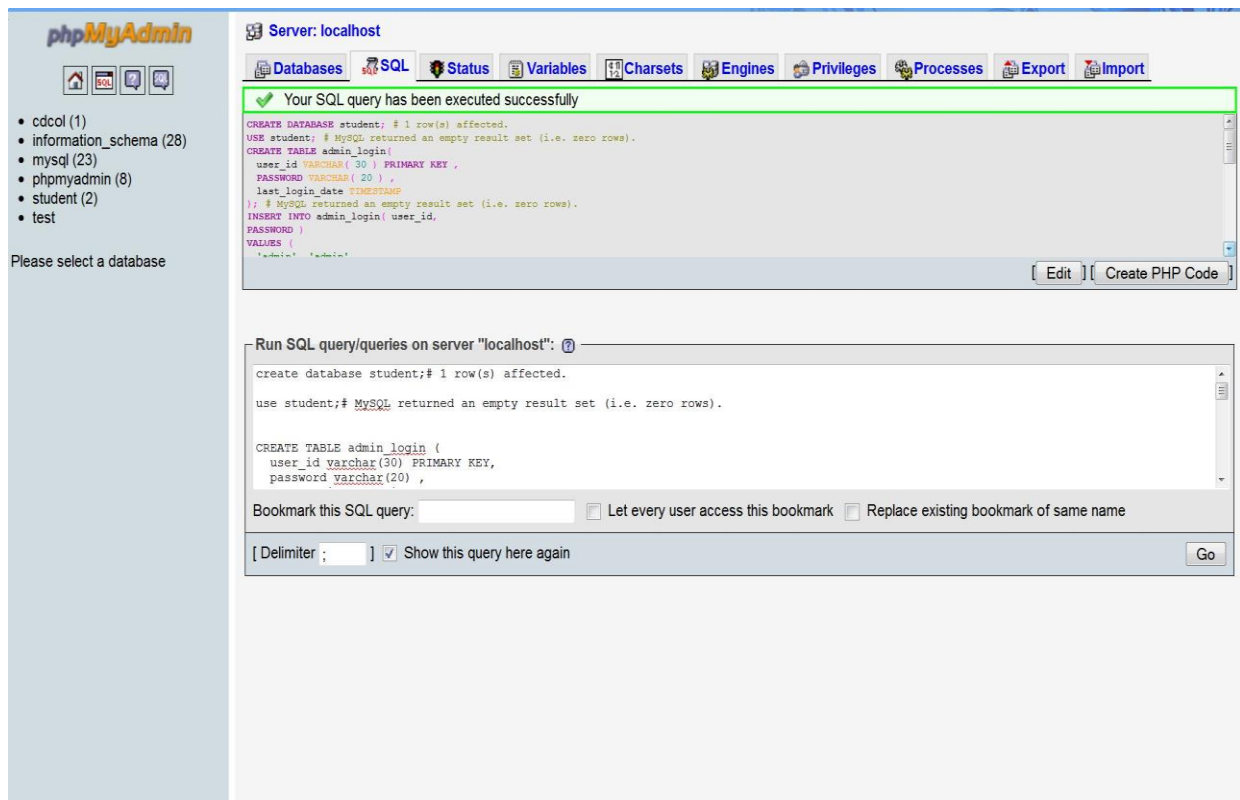


Fig (2.2) Creating database in XAMPP SERVER

The first step with phpMyAdmin running is creating a new database. We create a new database by directly executing SQL statements as shown below. The successful execution of the SQL query creates a database 'student' with two tables in it. The table are admin login and student information. We also inserted values in the admin table. The screenshot below shows the successful execution of the query thus creation of a database named student.

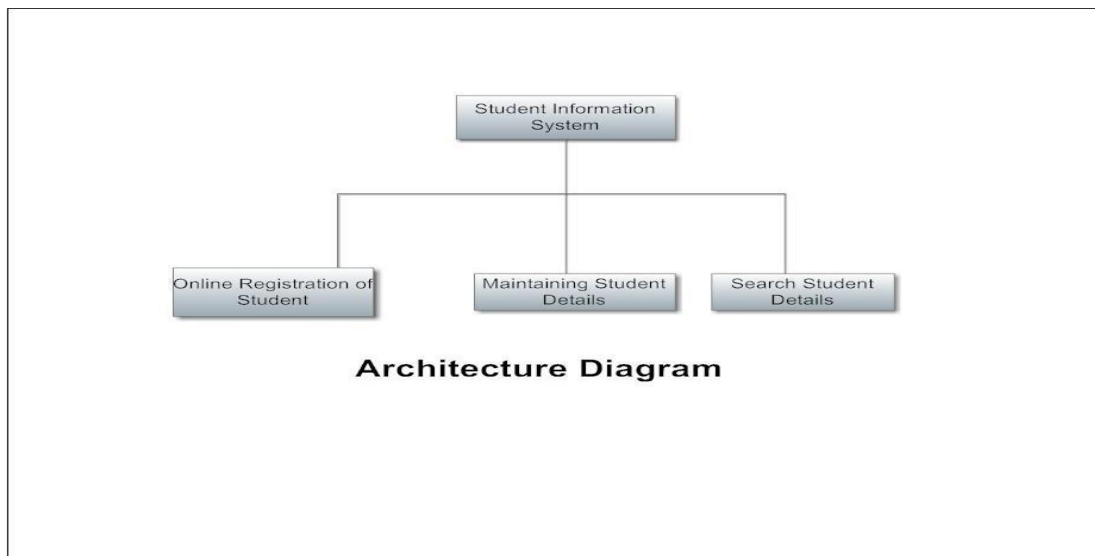


**Fig (2.3) Inserting database file in XAMPP SERVER**

# SYSTEM ARCHITECTURE DESIGN

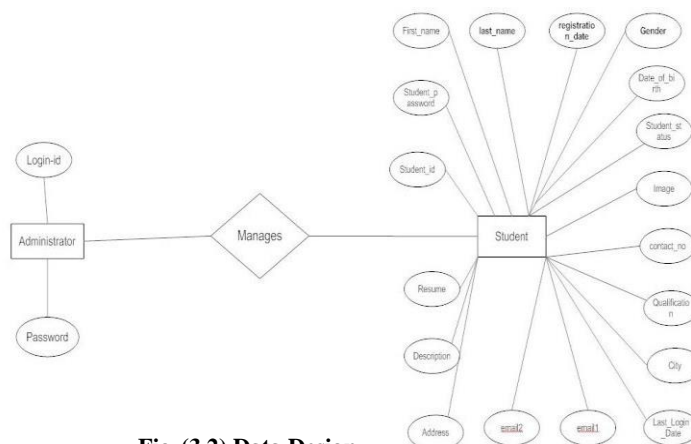
## 3.1 SYSTEM DESIGN

The SIS is a system which contain major part which include student Detail. The user selects one of the available options as an input to the system. According to the input by the user the system acts and the rest of the functions are performed accordingly. The administrator can operate on any student details, but the normal student or users can only access their details of all the functionalities.



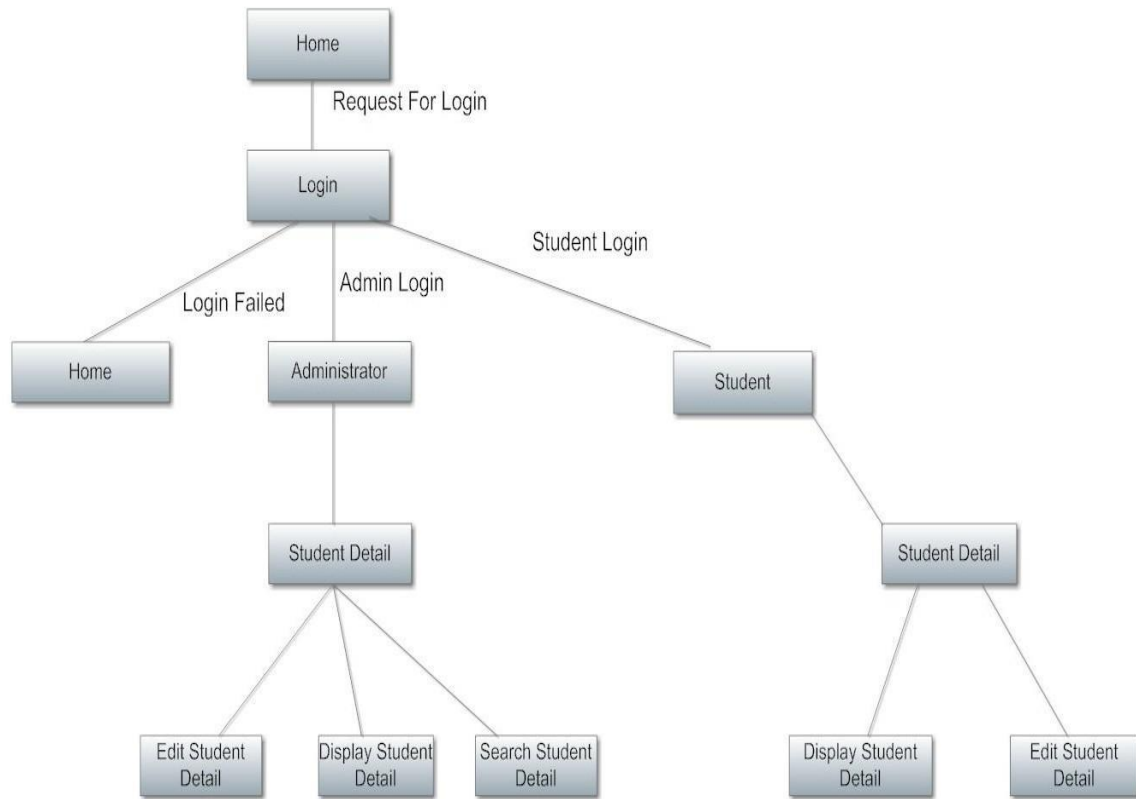
**Fig (3.1) System Architecture Diagram**

## 3.2 DATA DESIGN



**Fig (3.2) Data Design**

### 3.3 FUNCTIONAL DESIGN



#### DECISION TREE

**Fig (3.3) Functional Design**

# SYSTEM DIAGRAM

## 4.1 FLOW-CHART DIAGRAM

Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields. This flow chart diagram shows how the information system interact with admin as well as teacher.

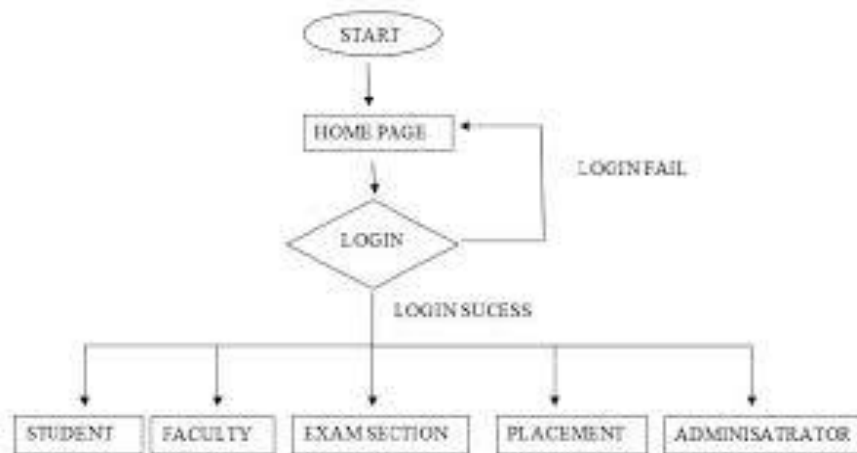


Fig (4.1) System Flow Chart Diagram

## 4.2 CONTEXT DIAGRAM

The context diagram is the most abstract data flow representation of a system. It represents the entire system as a single bubble and. The various external entities with which the system interacts and the data flows occurring between the system and the external entities are also represented. The name context diagram is well justified because it represents the context in which the system is to exist i.e. the external entities (users) that would interact with the system and specific data items they would be receiving from the system.

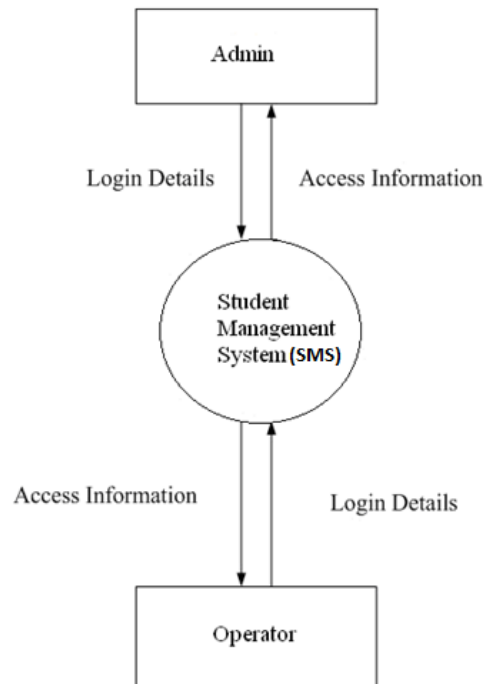


Fig (4.2) Context Diagram

### 4.3 DATA-FLOW DIAGRAM

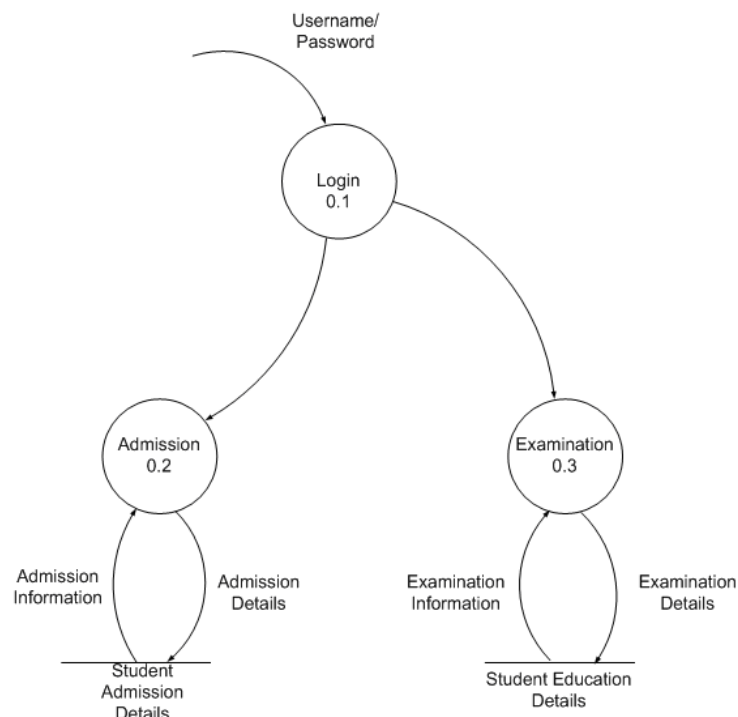


Fig (4.3) Level-1 DFD diagram for login

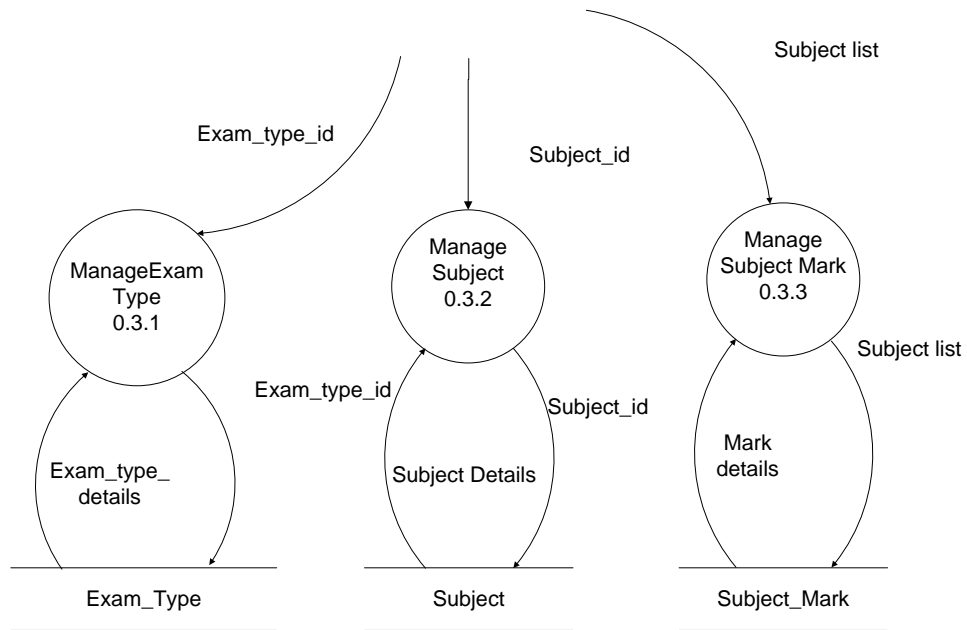


Fig (4.4) level-2 DFD diagram for marks

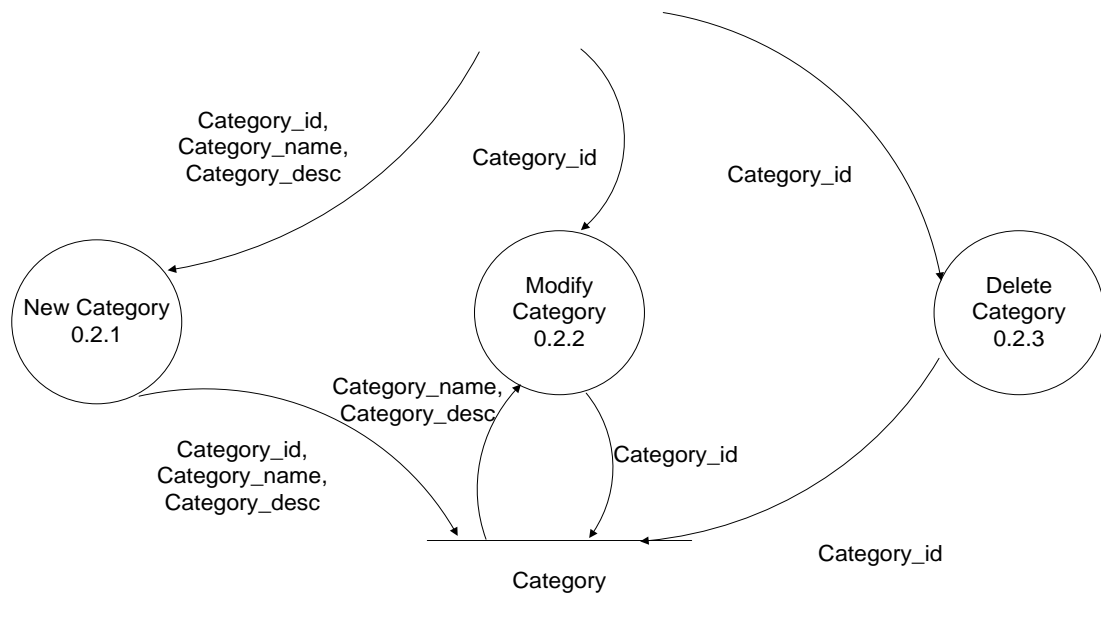


Fig (4.5) level-2 DFD diagram for Admin

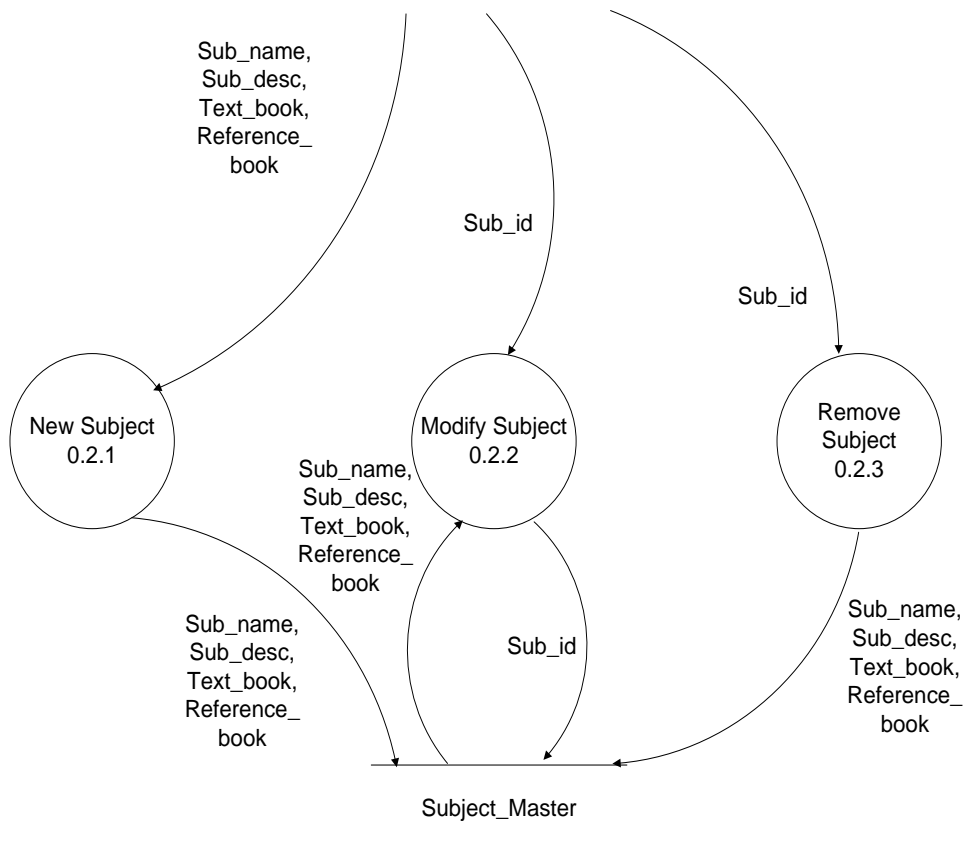


Fig (4.6) DFD diagram for modifying subject

## 4.4 E-R DIAGRAM

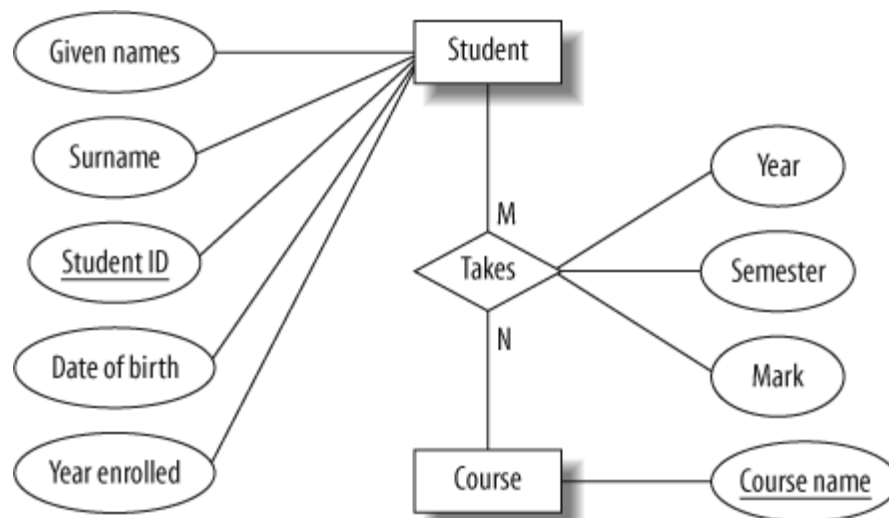


Fig (4.7) E-R diagram for student and course



## 4.5 USE-CASE

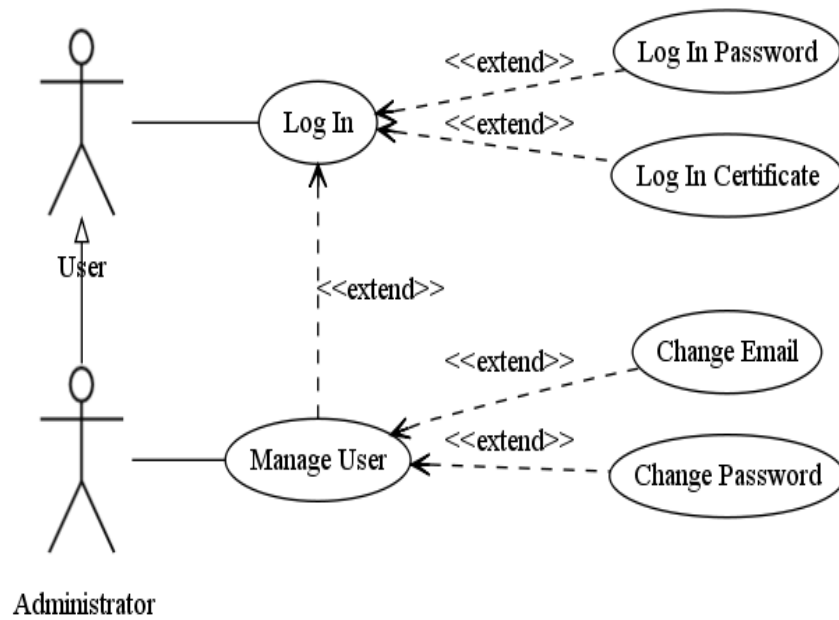


Fig (4.8) Use- Case diagram for system

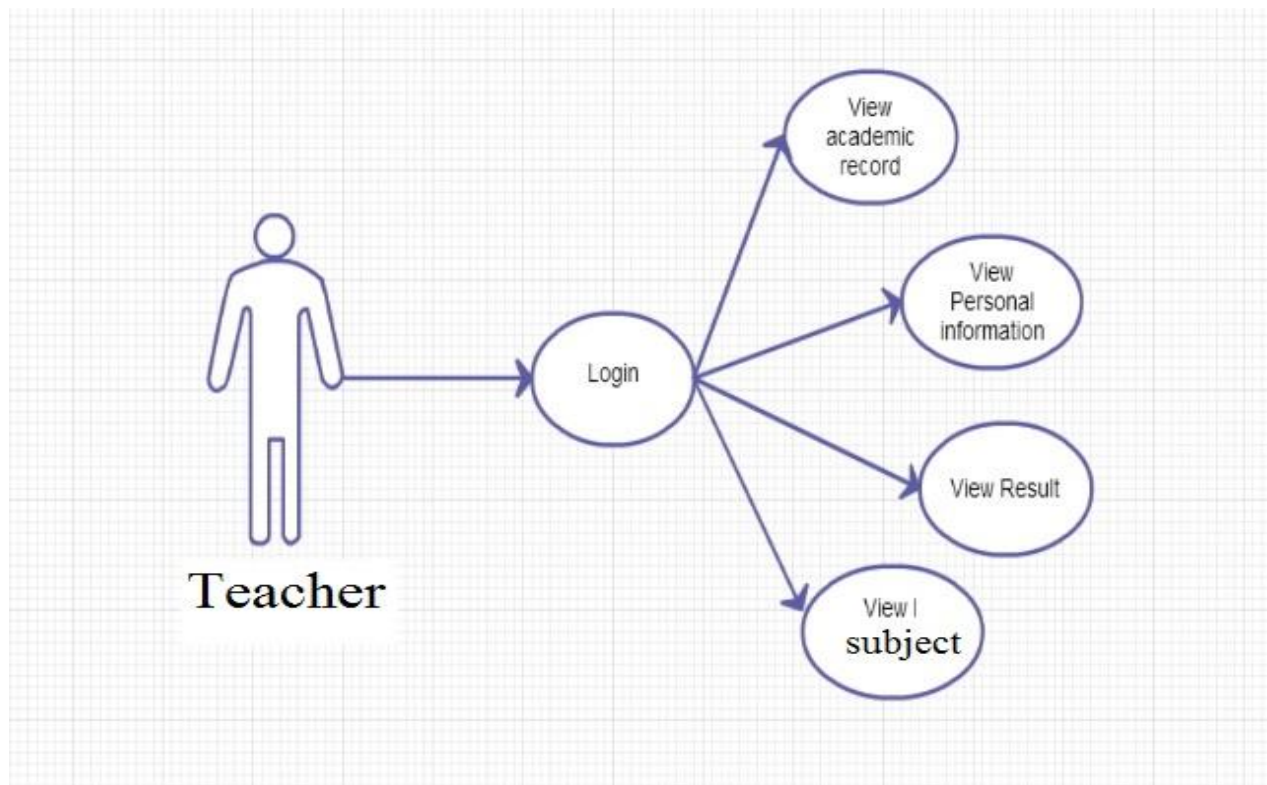


Fig (4.9) Use-Case diagram for teacher login

## SOURCE CODE

### 5.1 INDEX.PHP

```
<?php include("header.php");?>
<section id="page"><font color="#FFFFFF"></font>
<header id="pageheader" class="homeheader">
<h1 class="sitedescription"><h2>&nbsp;</h2></h1>
</header>
<article class="post">
<header class="postheader">
<h1><a href="#">Student Information System</a></h1><h3>&nbsp;</h3>
<h3><i>SRM UNIVERSITY</i></h3>
<a href="#"></a></p>
</header>
<p><b>Student Information Systems are the primary systems for managing information about
students in Schools/ Colleges.
    The Student Information System is a student-level data collection system that allows the
    Department to collect and analyze more accurate and comprehensive information.
    Student information systems provide capabilities for entering student records, tracking
    student attendance, and managing many other student-related data needs in a college or
    university.
</b></p>
</article>
<article class="post">
<header class="postheader">
<h1>Student Information System Supports:</h1><p class="postinfo">&nbsp;</p>
</header>
<p><b>•    Handling the Student details. </b></p>
<p><b>•    Maintaining the Student Marks Details. </b></p>
<p><b>•    Handling Student Attendance Records. </b></p>
<p><b>•    Maintaining records of absences and attendance.</b></p>
</article>
<div class="clear"></div>
<div class="clear"></div>
</section>
</div>
<?php include("footer.php"); ?>
```

### 5.2 ADMIN.PHP

```
<?php
session_start();
//echo $_SESSION["uid"];
if(isset($_SESSION["userid"]))
{
```

```

        if($_SESSION["type"]=="admin")
        {
            header("Location: dashboard.php");
        }
        else
        {
            header("Location: lectureaccount.php");
        }
    }
    include("header.php");
    include("conection.php");
    if(isset($_POST["uid"]) && isset($_POST["pwd"]) )
    {
        //      echo "sdfsd".  $_POST[uid];
        $result = mysql_query("SELECT * FROM administrator WHERE adminid='$_POST[uid]'");
        while($row = mysql_fetch_array($result))
        {
            $pwdmd5 = $row["password"];
        }

        if(md5($_POST["pwd"])== $pwdmd5)
        {
            $_SESSION["userid"] = $_POST["uid"];
            $_SESSION["type"]="admin";
            header("Location: dashboard.php");
        }
        else
        {
            $log = "Login failed.. Please try again..";
        }
    }
    if(isset($_POST["luid"]) && isset($_POST["lpwd"]))
    {
        $result = mysql_query("SELECT * FROM lectures WHERE lecid='$_POST[luid]'");
        while($row = mysql_fetch_array($result))
        {
            $pwdm= $row["password"];
            $_SESSION["lecname"] = $row["lecname"];
            $_SESSION["coid"] = $row["courseid"];
        }
        //echo"pwd". md5($_POST["lpwd"]);
        if(md5($_POST["lpwd"])==@$pwdm)
        {
            $_SESSION["userid"] = $_POST["luid"];
            $_SESSION["type"]=="lecturer";
            header("Location: lectureaccount.php");
        }
    }

```

```

}
Else
    $log12 = "Login failed.. Please try again..";
}
?>
<section id="page">
<header id="pageheader" class="normalheader">
<h2 class="sitedescription">
</h2>
</header>
<section id="contents">
<article class="post">
    <header class="postheader">
        <h2><u>Admin Login</u></h2>
        <h2><?php echo @$log;?></h2>
    </header>
    <section class="entry">
        <form action="admin.php" method="post" class="form">
            <p class="textfield">
                <label for="author">
                    <small>Admin Login ID (required)</small>
                </label>
                <input name="uid" id="uid" value="" size="22" tabindex="1" type="text">
            </p>
            <p class="textfield">
                <label for="email">
                    <small>Password (required)</small>
                </label>
                <input name="pwd" id="pwd" value="" size="22" tabindex="2" type="password">
            </p>
            <p>
                <input name="submit" id="submit" tabindex="5" type="image" src="images/submit.png">
                <input name="comment_post_ID" value="1" type="hidden">
            </p>
        </div class="clear"></div>
    </form>
    <form action="admin.php" method="post" class="form">
    <div class="clear">
    <hr />
    <header class="postheader">
        <h2><u>Lectures Login</u></h2>
        <h2><?php echo @$log12;?></h2>
    </header>
    <section class="entry">
        <p class="textfield">

```

```

        <label for="author2"> <small><br />
        Lecture Login ID (required)</small> </label>
        <input name="luid" id="luid" value="" size="22" tabindex="3" type="text" />
    </p>
    <p class="textfield">
        <label for="email2"> <small>Password (required)</small> </label>
        <input name="lpwd" id="lpwd" size="22" tabindex="4" type="password" /> </p><p>
        <input name="submit2" id="submit2" tabindex="5" type="image"
src="images/submit.png" />
        <input name="comment_post_ID2" value="1" type="hidden" />
    </p>
    <div class="clear"></div>
</form>
<div class="clear"></div>
</section>
</div>
</section>
</article>
</section>
<?php
include("adminmenu.php");
include("footer.php"); ?>

```

### 5.3 LECTURE.PHP

```

<?php
include("validation.php");
include("conection.php");
$totcourse = mysql_query("SELECT * FROM lectures");
$result1 = mysql_query("SELECT * FROM course");
$totid = mysql_num_rows($totcourse)+1;
if (@isset($_POST[button]))
{
    @$pwde = md5($_POST[password]);
    $sql="INSERT INTO lectures (lecid, password, courseid, lecname, gender, address ,contactno)
VALUES
('$_POST[lecid]','$pwde', '$_POST[course]', '$_POST[lecname]',
'$_POST[gender]','$_POST[address]','$_POST[contactno]')";
    if (!mysql_query($sql,$con))
    {
        die('Error: ' . mysql_error());
    }
    else
    {
        echo "Record inserted Successfully...";
    }
}
}

```

```

if(isset($_POST["button2"]))
{
    $pwde = md5($_POST[password]);
    mysql_query("UPDATE lectures SET lecname='$_POST[lecname]',
gender='$_POST[gender]', address='$_POST[address]', courseid='$_POST[coursekey]'
    contactno='$_POST[contactno]' WHERE lecid = '$_POST[lecid]'");
    echo "Record updated successfully...";
}
if(@$_GET["view"] == "lectures")
{
    $result = mysql_query("SELECT * FROM lectures where lecid='$_GET[slid]'");
    while($row1 = mysql_fetch_array($result))
    {
        $totid = $row1["lecid"];
        $lecname = $row1["lecname"];
        $password = $row1["password"];
        $courseid = $row1["courseid"];
        $gender = $row1["gender"];
        $address = $row1["address"];
        $contactno = $row1["contactno"];
    }
}
else
{
    $result = mysql_query("SELECT * FROM lectures");
}
?>
<form name="form1" method="post" action="" id="formID">
    <p>&nbsp;</p>
    <p>
        <label for="lecid">Lecturer ID&nbsp;&nbsp;&nbsp;</label>
        <input type="text" name="lecid" id="lecid" class="validate[required] text-input" readonly
value="<?php echo $totid; ?>">
    </p>
    <p>
        <label for="lecname">Lecturer Name</label>
        <input type="text" name="lecname" id="lecname"
class="validate[required,custom[onlyLetterSp]] text-input" value="<?php echo @$_lecname;
?>">
    </p>
    <p>
        <label for="password">Password</label>
        <input type="password" name="password" id="password" class="validate[required] text-
input">
    </p>
</p>

```

```

<label for="textfield4">Confirm Password</label>
<input type="password" name="textfield4" id="textfield4"
class="validate[required,equals[password]] text-input">
</p>
<p>
<label for="textfield2">Course </label>
<select name="course" value="<?php echo $courseid; ?>">
<option value="">Course Details</option>
<?php
while($row1 = mysql_fetch_array($result1))
{
    if($courseid == $row1[courseid])
    {
        $selvar = "selected";
    }
    echo "<option value='$row1[courseid]' ". $selvar . ">$row1[coursekey]</option>";
    $selvar = "";
}
?>
</select>
</p>
<p>Gender
<input type="radio" name="gender" id="radio" value="Male" <?php
if(@$gender == "Male")
{
    echo "checked";
}
?> class="validate[required] radio" />
<label for="radio">Male</label>
<input type="radio" name="gender" id="radio2" value="Female" <?php if(@$gender ==
"Female")
{
    echo "checked";
} ?> class="validate[required] radio" />
<label for="radio2">Female</label>
</p>
<p>
<label for="address">Address</label>
<textarea name="address" id="address" class="validate[required]" cols="25"
rows="5"><?php echo @$address; ?></textarea>
</p>
<p>
<label for="contactno">Contact No</label>
<input type="text" name="contactno" id="contactno"
class="validate[required,custom[phone]] text-input" value="<?php echo @$contactno; ?>">
</p>

```

```

<p>
  <input type="submit" name="button" id="button" value="Submit">
  <input type="submit" name="button2" id="button2" value="Update" />
  <input type="submit" name="button2" id="button2" value="Reset">
<form id="myform">
  <input type="button" value="Close" name="B1" onClick="parent.emailwindow.hide()" /></p>
</form>
</p>
<p>&nbsp;</p>
</form>
<a href='lecturereview.php'><< Back </a>

```

## 5.4 LOGOUT.PHP

```

<?php
session_start();
include("header.php");
session_destroy();
?>
<section id="page">
<header id="pageheader" class="homeheader">
</header>
<section id="contents">
<article class="post">
<header class="postheader">
<h2><a href="#">Logged Out Successfully</a>...</h2>
<p class="postinfo">&nbsp;</p>
</header>
<p>
  <article class="post">
    <footer class="postfooter"></footer>
  </article>
</p>
</article>
<div class="blog-nav"></div>
</section>
<div class="clear"></div>

<div class="clear"></div>
</section>
</div>
<?php include("footer.php"); ?>

```



## SCREENSHOTS

### 6.1 MAIN PAGE



Fig (6.1) Home- Page

## 6.2 STUDENT PAGE



The image shows a web page for SRM University with a wooden frame border. At the top, the text "SRM University" is on the left, and four buttons labeled "HOME", "STUDENTS", "ADMIN", and "CONTACT-US" are on the right. Below the navigation bar is a large photograph of a university building. Under the photo, there are two search sections. The first section, titled "Student Registration No.", has a text input field for "Roll No" and a "Submit" button. The second section, titled "Search by Name and Class", has a text input field for "Name", a dropdown menu for "Course" with "B Tech" selected, and a "Submit" button.

**SRM University**   HOME   STUDENTS   ADMIN   CONTACT-US



*Student Registration No.*

Roll No

*Search by Name and Class*

Name

Course

Fig (6.2) Student Search Page

## 6.3 ADMIN/ LECTURE PAGE



The image shows a web page for SRM University with a wooden-themed header. The header contains the university name and four navigation buttons: HOME, STUDENTS, ADMIN, and CONTACT-US. Below the header is a banner image of a university building. The main content area has two login sections. The first section, titled 'Admin Login', has a label 'Admin Login ID (required)' above a text input field containing '123', a label 'Password (required)' above a password input field with masked characters, and a 'SUBMIT' button. The second section, titled 'Lectures Login', has a label 'Lecture Login ID (required)' above a text input field containing '1', a label 'Password (required)' above a password input field with masked characters, and a 'SUBMIT' button.

**SRM University**   HOME   STUDENTS   ADMIN   CONTACT-US

*Admin Login*

Admin Login ID (required)  
123

Password (required)  
.....

SUBMIT

---

*Lectures Login*

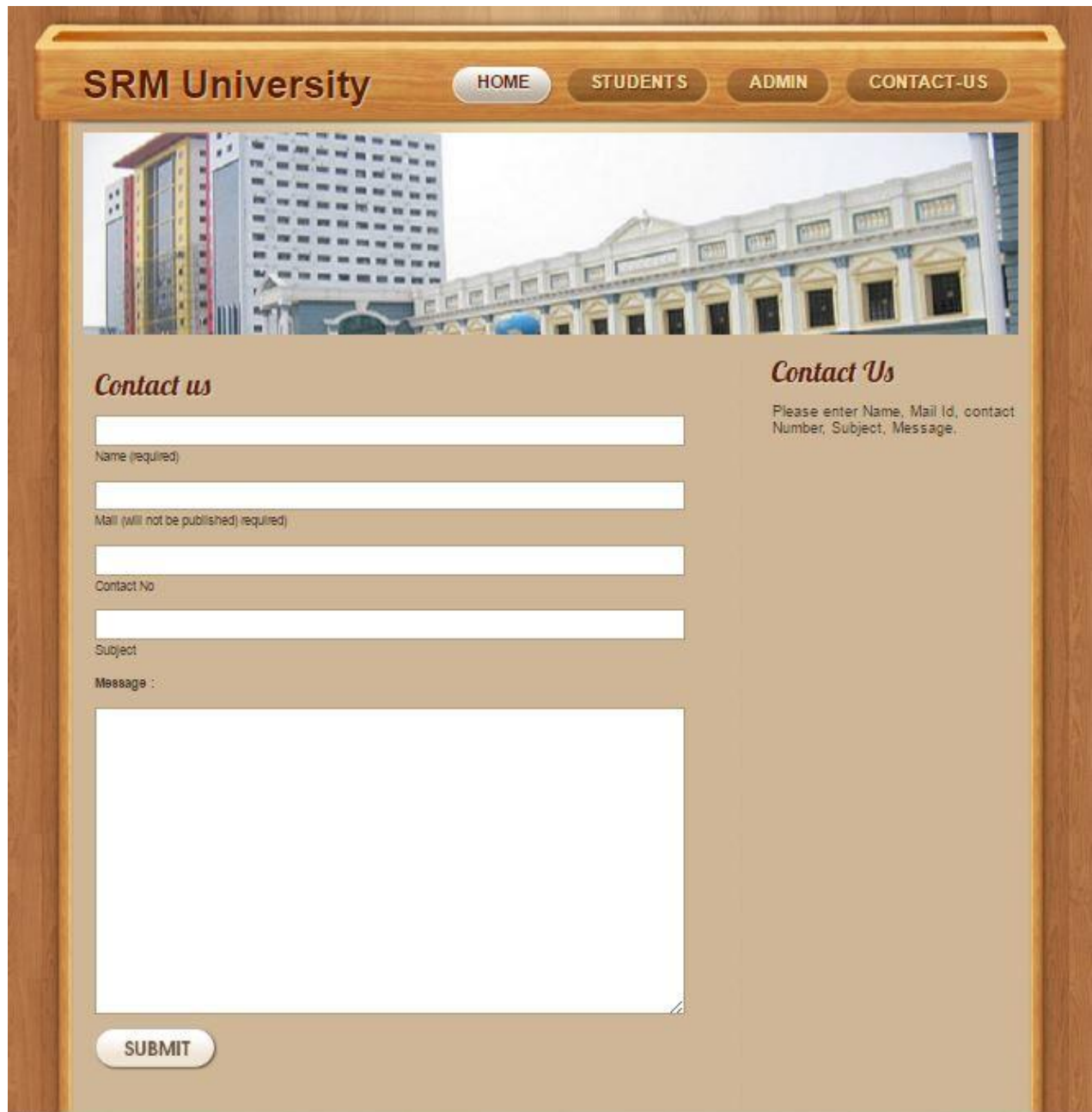
Lecture Login ID (required)  
1

Password (required)  
.....

SUBMIT

Fig (6.3) Amin/ Lecturer login Page

## 6.4 CONTACT PAGE



The image shows a web page for SRM University with a contact form. The page has a wooden-themed header with the university name and navigation buttons. Below the header is a banner image of a university building. The contact form is titled 'Contact us' and includes fields for Name, Mail, Contact No, Subject, and a large Message area, followed by a SUBMIT button. To the right of the form, there is a heading 'Contact Us' and a note asking users to enter their details.

**SRM University**   [HOME](#)   [STUDENTS](#)   [ADMIN](#)   [CONTACT-US](#)

*Contact us*

Name (required)

Mail (will not be published) required

Contact No

Subject

Message :

*Contact Us*

Please enter Name, Mail Id, contact Number, Subject, Message.

Fig (6.4) Contact Page

## **CONCLUSION**

- SIS will be helpful to perform paperless work and manage all data.
- This provides easy, accurate, unambiguous and faster data access.
- This Student Information System project will serve as a useful approach to database dialog box to update, add, delete for the authorized person. It reduces the time taken by the user to update, add, delete, view & search the information of student.
- Thus, the project is the user friendly approach.

## REFERENCES

- PHP book by Vasvani (TMH publications)
- Beginning PHP5 by WROX
- [www.google.com](http://www.google.com)
- [www.wikipedia.com](http://www.wikipedia.com)
- [www.w3schools.com](http://www.w3schools.com)
- [www.codeguru.com](http://www.codeguru.com)
- JavaScript Bible by Danny Goodman