**CHAPTER 1**

1. **INTRODUCTION**
   1. **Problem statement**

The problem of the system is that there was no web applications available for existing system, and that existing system are only accessing the database source and doesn’t provide access for external data source Excel book. And another problem is that it was not more interactive with user. In the existing system, the report is built only using these excel and it was difficult for the admin to build a report based on a set of queries. And there was no direct connection to build a report from excel and database. These problems lead to lot of human errors and to overcome these problems, this web application is developed from scratch.

**CHAPTER 2**

**SYSTEM ANALYSIS**

* 1. **Existing System**

In existing system only database can be accessed. The existing report system will allow data to come from tables, views, or stores procedures within the report builder. It doesn’t provide access for external data source excel book.

* 1. **Proposed System**

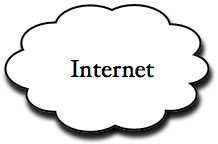
To overcome the existing system, Report Builder Application can provide the functionalities to insert a new student record, store all the student records in the user interface form and provide the data reports using Excel inputs and reports can be retrieved in various formats such as doc file, pdf file and Excel file. By this Application all the information stored in the database will be in a uniform manner.

* 1. **System Architecture**

System Architecture of the application starts with the super admin and staff who interacts with the server through the frontend. This server connects with the database.

Super-admin Browser/client system



REPORT BUILDER APPLICATION V.1.0

Excel

Sever DB

*FIG 2.3.1: System Architecture*

* 1. **Subsystem Description**

Subsystem description describes how the components is further divided into subcomponents and relationships and interaction between the subcomponents.

Subsystem of this tool are described below

* Manage students
* Reports
* Load Excel file
* Manage staff

**1. MANAGE STUDENTS:**

In this module super admin has full privilege of adding new student to all the departments, modifying student information, further updates or deletion can be done only by the super admin. Staffs can also view the student record of their respective department and based on the privileging rights given by the super admin to the staff. Staff can modify or delete the student record from the database.

**2. REPORTS:**

Reports are customized document. There are predefined categories based on that the information needed, the data’s are retrieved from the database and that can be generated as a report and also based on the user requirement. The categories can be customized and the data are retrieved as per the requirement from the database and the reports are generated the different types of charts are generated from the report and the user can select type of the chart according to their wish. The generated report can be downloaded in the various formats like pdf, excel and csv. Super admin can able to view all the department records whereas staffs can view only their privileged departments. Super admin can able to generate the report from all the departments, staff can generate the report only from their department.

**3. LOAD EXCEL FILE:**

This module can be accessed by both super admin and staffs. Here reports are generated from the external excel file. Once the user login into the system can only able access this module. Even the data from the excel file can be customized as per the requirement and the report are generated based on the data required.

**4. MANAGE STAFF:**

This module fully controlled by super admin. To manage all the staff login process and giving access privilege is done super admin using in this module. Access privilege in the sense each staff may belong to one department there is no need of accessing other department process or information. So cut down those privilege super admin will give the access privilege to only the department they belong to. So the staff cannot access other department. This process is done by the super admin. Super admin may able to give access privilege to one or more department if there is need of accessing. Super admin can also change there or modify the access privilege whenever is a need.

* 1. **Functional requirements**
     1. **Data description**

The data that have been used in this system are described over here.

* + - 1. **Data objects**

Data object that compile to system are:

|  |  |
| --- | --- |
| **DATA OBJECT** | **ADMISSION\_DETAILS** |
| **FIELD** | **DESCRIPTION** |
| ADMISSION\_NO | Admission number |
| ADMISSION\_DATE | Date of admission |
| ADMISSION\_QUOTA | Admission quota |

|  |  |
| --- | --- |
| **DATA OBJECT**  **FIELD** | **STUDENT PERSONAL DETAILS**  **DESCRIPTION** |
| ADMISSION\_NO | Admission number for student |
| STU\_ROLLNO | Student unique roll no |
| STU\_FIRSTNAME | Student first name |
| STU\_LASTNAME | Student last name |
| STU\_GENDER | Student gender |
| STU\_DOB | Student date of birth |
| STU\_RELIGION | Student religion |
| STU\_COMMUNITY | Student community |
| STU\_MOTHER MAIDEN NAME | Student mother maiden name |
| STU\_MOTHER NAME | Student mother name |
| STU\_FATHER NAME | Student father name |
| STU\_PARENT\_INCOME | Student parent’s income |
| STU\_NATIONALITY | Student nationality |
| STU\_BLOOD\_GROUP | Student blood group |
| STU\_MOTHER\_TONGUE | Student mother tongue |
| STU\_LANGUAGE\_KNOWN1 | Student language known 1 |
| STU\_LANGUAGE\_KNOWN2 | Student language known 2 |

|  |  |
| --- | --- |
| **DATA OBJECT**  **FIELD** | **STUDENT CONTACT DETAILS**  **DESCRIPTION** |
| STU\_ROLLNO | Student unique roll no |
| STU\_EMAIL | Student personal email id |
| STU\_PARENT\_EMAIL | Student parent email id |
| STU\_MOBILE | Student personal mobile number |
| STU\_PARENT\_MOBILE | Student parents’ mobile number |
| STU\_ALTER\_MOBILE | Student alternative mobile |
| PRESENT\_HOUSENO | Student present address house number |
| PRESENT\_STREET | Student present address street |
| PRESENT\_AREA | Student present address area or landmark |
| PRESENT\_CITY | Student present address city |
| PRESENT\_DISTRICT | Student present address district |
| PRESENT\_STATE | Student present address state |
| PRESENT\_COUNTRY | Student present address country |
| PRESENT\_PINCODE | Student present address pin code |
| PERMANENT\_HOUSENO | Student permanent address house no |
| PERMANENT\_STREET | Student permanent address street |
| PERMANENT\_AREA | Student permanent address area or landmark |
| PERMANENT\_CITY | Student permanent address city |
| PERMANENT\_DISTRICT | Student permanent address district |
| PERMANENT\_STATE | Student permanent address state |
| PERMANENT\_COUNTRY | Student permanent address country |
| PERMANENT\_PINCODE | Student permanent address pin code |

|  |  |
| --- | --- |
| **DATA OBJECT**  **FIELD** | **STUDENT CURRENT COURSE DETAILS**  **DESCRIPTION** |
| STU\_ROLLNO | Student unique identity roll no |
| STU\_REGNO | Student unique university register number |
| STU\_DEGREE | Student currently pursing degree |
| STU\_COURSE | Student currently pursing course |
| STU\_BRANCH | Student currently pursing branch |
| STU\_SECTION | Student section |
| STU\_BATCH | Student batch year |
| STU\_COURSE\_TYPE | Student course type |
| STU\_JOIN\_MODE | Student join mode for regular or lateral entry |

|  |  |
| --- | --- |
| **DATA OBJECT**  **FIELD** | **STUDENT PREVIOUS ACADEMIC DETAILS**  **DESCRIPTION** |
| STU\_ROLLNO | Student unique identity roll no |
| PREV\_DEGREE | Student previous studied degree |
| PREV\_COURSE | Student previous studied course |
| PREV\_BRANCH | Student previous studied branch |
| YEAR\_OF\_PASSING | Year of passing previous degree |
| COURSE\_TYPE | Previous course type |
| INSTITUTE NAME | Previous institute name |
| BOARD OF EDUCATION/UNIVERSITY NAME | Previous degree board of education |
| CGPA/MARK OBTAINED | Previous degree cgpa or mark obtained |
| TOTAL CGPA/MARK | Total cgpa or mark for previous degree |
| PERCENTAGE | Percentage of previous degree |

|  |  |
| --- | --- |
| **DATA OBJECT**  **FIELD** | **STUDENT BLACK MARK DETAILS**  **DESCRIPTION** |
| REMARK\_ID | Remark unique id |
| STU\_ROLLNO | Student unique roll no |
| REMARK\_DATE | Remark given date |
| REMARK\_STAFF | Staff name who given remark for student |
| REMARK\_REASON | Remark reason |
| MET\_STAFF | Student met staff yes or no |
| DATE\_OF\_MET | Remark student date of met to staff |

|  |  |
| --- | --- |
| **DATA OBJECT**  **FIELD** | **DEGREE DETAILS**  **DESCRIPTION** |
| DEGREE\_ID | Degree unique id |
| DEGREE\_NAME | Degree name |

|  |  |
| --- | --- |
| **DATA OBJECT**  **FIELD** | **COURSE DETAILS**  **DESCRIPTION** |
| COURSE\_ID | Course unique id |
| DEGREE\_ID | Degree unique id |
| COURSE\_NAME | Course name |

|  |  |
| --- | --- |
| **DATA OBJECT**  **FIELD** | **BRANCH DETAILS**  **DESCRIPTION** |
| BRANCH\_ID | Branch unique id |
| COURSE\_ID | Course unique id |
| BRANCH\_NAME | Branch name |

|  |  |
| --- | --- |
| **DATA OBJECT**  **FIELD** | **STAFFS DETAILS**  **DESCRIPTION** |
| STAFF\_ID | Staff unique identity number |
| STAFF\_NAME | Staff name |
| STAFF\_GENDER | Staff gender |
| STAFF\_DOB | Staff date of birth |
| PRESENT ADDRESS | Present address for staff |
| PERMANAENT ADDRESS | Permanent address for staff |
| STAFF\_EMAIL | Staff unique personal email |
| STAFF\_MOBILE | Staff contact mobile number |
| STAFF\_DESIGNATION | Staff designation HOD or staffs |
| STAFF\_DEPT | Staff department |
| STAFF\_DOJ | Date of joins in staff |
| STAFF\_QUALIFICATION | Qualification for staff |

|  |  |
| --- | --- |
| **DATA OBJECT**  **FIELD** | **LOGIN\_INFO DETAILS**  **DESCRIPTION** |
| STAFF\_ID | Staff unique identity id |
| USERNAME | Staff username for login |
| PASSWORD | Staff password for login |

|  |  |
| --- | --- |
| **DATA OBJECT**  **FIELD** | **ACCESS RIGHTS\_INFO DETAILS**  **DESCRIPTION** |
| ID | Unique permission id |
| STAFF\_ID | Unique identity id for staff |
| PERMISSION | Access permissions for staff |
| MODULE | Access permission belongs to modules |

* 1. **NON – FUNCTIONAL REQUIREMENTS**

Non-functional requirements define the needs in terms of performance, reliability, security, maintainability, and portability

**Performance requirements**

Performance requirements define acceptable response times for the system functionality, this system can transfer register information from application to database in 5 seconds.

**Reliability**

This application can be used for any number of systems and provide good efficiency.

**Security**

This system provides high security by authorizing staffs using their user id

**Maintainability**

This application is being developed in PHP. So, it is easy to maintain.

**Portability**

This software is easily transferred to another environment, including install ability.

**2.6.1 Hardware Specification**

**Client side** (minimum required)

* Processor : Pentium 4
* RAM : 1 GB
* Disk space : 1 GB

**Server side** (minimum required)

* Processor : Intel Xeon
* RAM : 1 GB
* Disk space : 100 GB

**2.6.2 Software Specification**

**Client side** (minimum required)

* Front End : HTML
* Back End : MYSQL
* Scripting Language : JavaScript and jQuery
* Framework : BOOTSTRAP
* Operating System : Windows XP SP3

**Server side** (minimum required)

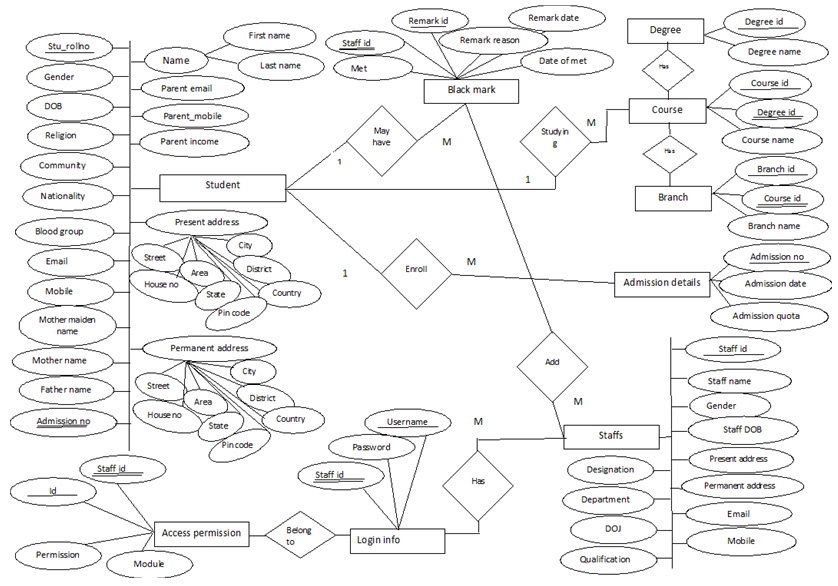
* Business Logic : PHP
* Server : Apache Tomcat 6.0
* Operating System : LINUX

**CHAPTER 3**

**SYSTEM DESIGN**

**3.1 DATA MODEL**

3.1.1 E-R DIAGRAM

*****Fig**3.1.1 E-R Diagram*

**3.1.2 TABLE DESIGN:**

**3.1.2.1 ADMISSION\_DETAILS:**

|  |  |  |
| --- | --- | --- |
| FIELDS | DATATYPE | CONSTRAINTS |
| admission\_no | INTEGER | PRIMARY KEY |
| admission\_date | DATE | NOTNULL |
| admission\_quota | VARCHAR(25) | NOTNULL |

**3.1.2.2 STUDENT\_PERSONAL\_DETAILS:**

|  |  |  |
| --- | --- | --- |
| FIELDS | DATATYPE | CONSTRAINT |
| Admission\_no | INTEGER | FOREIGN KEY (admission\_details) |
| stu\_rollno | INTEGER(11) | PRIMARY KEY |
| stu\_firstname | VARCHAR(40) | NOTNULL |
| stu\_lastname | VARCHAR(40) | NOTNULL |
| stu\_gender | VARCHAR(8) | NOTNULL |
| stu\_dob | DATE | NOTNULL |
| stu\_religion | VARCHAR(40) | NOTNULL |
| stu\_community | VARCHAR(40) | NOTNULL |
| stu\_mother’s\_maiden\_name | VARCHAR(40) | NULL |
| stu\_mother’s\_name | VARCHAR(40) | NOTNULL |
| stu\_father’s\_name | VARCHAR(40) | NOTNULL |
| stu\_parent\_income | DOUBLE (10,2) | NOTNULL |
| stu\_nationality | VARCHAR(40) | NOTNULL |
| stu\_blood\_group | VARCHAR(5) | NOTNULL |
| stu\_mother\_tongue | VARCHAR(40) | NOTNULL |
| stu\_language\_known1 | VARCHAR(40) | NULL |
| stu\_language\_known2 | VARCHAR(40) | NULL |

**3.1.2.3 STUDENT\_CONTACT\_DETAILS:**

|  |  |  |
| --- | --- | --- |
| FIELDS | DATATYPE | CONSTRAINT |
| stu\_rollno | INTEGER | FOREIGNKEY(stu\_personal\_details)  PRIMARY KEY |
| stu\_email | VARCHAR(50) | UNIQUE |
| stu\_parent\_email | VARCHAR(50) | NULL |
| stu\_mobile | VARCHAR(50) | UNIQUE |
| stu\_alter\_mobile | VARCHAR(50) | NOTNULL |
| stu\_parents\_mobile | VARCHAR(50) | NOTNULL |
| stu\_pre\_houseno | VARCHAR (15) | NOTNULL |
| stu\_pre\_street | VARCHAR (50) | NOTNULL |
| stu\_pre\_area | VARCHAR(100) | NOTNULL |
| stu\_pre\_city | VARCHAR(100) | NOTNULL |
| stu\_pre\_district | VARCHAR(100) | NOTNULL |
| stu\_pre\_state | VARCHAR(100) | NOTNULL |
| stu\_pre\_country | VARCHAR(100) | NOTNULL |
| stu\_pre\_pincode | MEDIUMINT(9) | NOTNULL |
| stu\_per\_houseno | VARCHAR (15) | NOTNULL |
| Stu\_per\_street | VARCHAR(50) | NOTNULL |
| stu\_per\_area | VARCHAR(100) | NOTNULL |
| stu\_per\_city | VARCHAR(100) | NOTNULL |
| stu\_per\_district | VARCHAR(100) | NOTNULL |
| stu\_per\_state | VARCHAR(100) | NOTNULL |
| stu\_per\_country | VARCHAR(100) | NOTNULL |
| stu\_per\_pincode | MEDIUMINT(9) | NOTNULL |

**3.1.2.4 STUDENT\_COURSE\_DETAILS:**

|  |  |  |
| --- | --- | --- |
| FIELDS | DATATYPE | CONSTRAINTS |
| stu\_rollno | INTEGER | PRIMARY KEY/FOREIGN KEY(stu\_personal\_details) |
| stu\_univ\_regno | BIGINT | UNIQUE |
| stu\_degree | INTEGER | NOTNULL |
| stu\_course | INTEGER | NOTNULL |
| stu\_branch | INTEGER | NOTNULL |
| stu\_section | VARCHAR(4) | NULL |
| stu\_batch | VARCHAR(12) | NOTNULL |
| stu\_course\_type | VARCHAR(30) | NOTNULL |
| stu\_joined | VARCHAR(30) | NOTNULL |

**3.1.2.5 STUDENT PREVIOUS ACADEMIC DETAILS:**

|  |  |  |
| --- | --- | --- |
| FIELD | DATATYPE | CONSTRAINTS |
| id | INTEGER | PRIMARY KEY |
| Stu\_rollno | INTEGER | FOREIGN KEY (stu\_personal\_details) |
| prev\_degree | VARCHAR(20) | NOTNULL |
| prev\_course | VARCHAR(50) | NOTNULL |
| prev\_branch | VARCHAR(50) | NOTNULL |
| yr\_of\_passing | YEAR | NOTNULL |
| course\_type | VARCHAR(30) | NOTNULL |
| institute\_name | VARCHAR(150) | NOTNULL |
| board\_of education/university name | VARCHAR(100) | NOTNULL |
| cgpa/mark obtained | FLOAT | NOTNULL |
| total mark/cgpa | INTEGER | NOTNULL |
| percentage | FLOAT | NOTNULL |

**3.1.2.6 BLACK\_MARK DETAILS:**

|  |  |  |
| --- | --- | --- |
| FIELD | DATATYPE | CONSTRAINTS |
| remark\_id | INTEGER | PRIMARY KEY |
| stu\_rollno | INTEGER | FOREGIN KEY (stu\_personal\_details) |
| remark\_date | DATE | NOTNULL |
| remark\_staff | INTEGER | FOREIGN KEY(staff\_details) |
| remark\_reason | TEXT | NOTNULL |
| met\_staff | VARCHAR(5) | NOTNULL |
| date\_of\_met | DATE | NOTNULL |

**3.1.2.7 DEGREE:**

|  |  |  |
| --- | --- | --- |
| FIELDS | DATATYPE | CONSTRAINTS |
| degree\_id | SMALLINT | PRIMARY KEY |
| degree\_name | VARCHAR(200) | NOTNULL |

**3.1.2.8 COURSE:**

|  |  |  |
| --- | --- | --- |
| FIELDS | DATATYPE | CONSTRAINTS |
| course\_id | SMALLINT | PRIMARY KEY |
| degree\_id | SMALLINT | FOREIGN KEY(degree) |
| course\_name | VARCHAR(100) | NOTNULL |
| course\_duration | SMALLINT | NOTNULL |

**3.1.2.9 BRANCH:**

|  |  |  |
| --- | --- | --- |
| FIELD | DATATYPE | CONSTRAINTS |
| branch\_id | SMALLINT | PRIMARY KEY |
| course\_id | SMALLINT | FOREIGN KEY(courses) |
| branch\_name | VARCHAR(50) | NOTNULL |

**3.1.2.10 STAFF\_DETAILS:**

|  |  |  |
| --- | --- | --- |
| FIELD | DATATYPE | CONSTRAINTS |
| Staff\_id | INTEGER | PRIMARY KEY |
| Staff\_name | VARCHAR(30) | NOTNULL |
| Staff\_gender | VARCHAR(10) | NOTNULL |
| Staff\_dob | DATE | NOTNULL |
| Staff\_present\_address | TEXT | NOTNULL |
| Staff\_permanent\_address | TEXT | NOTNULL |
| Staff\_email | VARCHAR(50) | NOTNULL |
| Staff\_mobile | VARCHAR(10) | NOTNULL |
| Staff\_date\_of\_join | DATE | NOTNULL |
| Staff\_qualification | VARCHAR(50) | NOTNULL |

**3.1.2.11 STAFF\_NON\_TEACHING:**

|  |  |  |
| --- | --- | --- |
| FIELD | DATATYPE | CONSTRAINTS |
| Id | INTEGER | PRIMARY KEY |
| Staff\_id | INTEGER | FOREIGNKEY(staff details) |
| department | VARCHAR(100) | NOTNULL |
| designation | VARCHAR(100) | NOTNULL |
| Join date | DATE | NOTNULL |

**3.1.2.12 STAFF\_TEACHING:**

|  |  |  |
| --- | --- | --- |
| FIELD | DATATYPE | CONSTRAINTS |
| Id | INTEGER | PRIMARY KEY |
| Staff\_id | INTEGER | FOREIGNKEY(staff details) |
| branch | INTEGER | FOREIGN KEY(branch) |
| designation | VARCHAR(100) | NOTNULL |
| join date | DATE | NOTNULL |

**3.1.2.13 LOGIN\_INFO:**

|  |  |  |
| --- | --- | --- |
| FIELD | DATATYPE | CONSTRAINTS |
| Staff\_id | INTEGER | FOREIGN KEY(staff\_details) |
| username | VARCHAR(50) | PRIMARY KEY |
| password | VARCHAR(32) | NOTNULL |

**3.1.2.14 ACCESS\_RIGHTS INFO:**

|  |  |  |
| --- | --- | --- |
| FIELD | DATATYPE | CONSTRAINTS |
| Id | INTEGER | PRIMARY KEY |
| username | VARCHAR(50) | FOREGIN KEY (username) |
| permission | VARCHAR(50) | NOTNULL |
| Module | VARCHAR(50) | NOTNULL |
| Dept\_permission | VARCHAR(50) | NOTNULL |

**3.2 PROCESS MODEL**

**3.2.1 USE CASE DIAGRAM**

**3.2.1.1SUPER-ADMIN/STAFF**

****

FIG 3.2.1.1:*SUPER-ADMIN/STAFF*

**3.2.2 DATA FLOW DIAGRAM**

**3.2.2.1. Level 0:**

Store and retrieve login

Login

Confirmation

Super-admin

Add, delete, and edit student record

Confirmation store student data

Generate reports

Retrieve record

Confirmation report data

*FIG 3.2.2.1 Level 0 Data Flow Diagram*

**3.2.2.2. Level 1:**

Confirmation Add student

Super-Admin

ModifyDB Student data

Confirmation update data

Delete

Confirmation

*FIG 3.2.2.2 Level 1 Data Flow Diagram*

**3.2.2.3. Level 2:**

Generate report

Request to search data

Data

Super -admin

Send corresponding record

*FIG 3.2.2.3 Level 2 Data Flow Diagram*

**3.2.3 SEQUENCE DIAGRAM**

**3.2.3.1 MANAGE STUDENTS**



*FIG 3.2.3.1: Manage Students*

**3.2.3.2 CREATE STAFF LOGIN**



*FIG 3.2.3.2: Create Staff Login*

**3.2.3.3 REPORT GENERATION FROM DB**



*FIG 3.2.3.3: Report Generation from DB*

**3.2.3.4 REPORT GENERATION FROM EXCEL**



*FIG 3.2.3.4: Report Generation from Excel*

**3.2.4 COLLABORATION DIAGRAM**

**3.2.4.1 MANAGE STUDENTS**



*FIG 3.2.4.1: Manage Students*

**3.2.4.2 CREATE STAFF LOGIN**



*FIG 3.2.4.2: Create Staff Login*

**3.2.4.3 REPORT GENERATION FROM DB DATA**



*FIG 3.2.4.3: Report Generation from DB Data*

**3.2.4.4 REPORT GENERATION FROM EXCEL DATA**



*FIG 3.2.4.4: Report Generation from Excel Data*

**3.2.5 ACTIVITY DIAGRAM**



*FIG 3.2.5.1: Activity Diagram*

**CHAPTER 4**

**SYSTEM IMPLEMENTATION**

**4.1 SAMPLE CODE**

**Index.php**

<! DOCTYPE html>

<Html>

<Head>

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<title>AdminLTE 2 | Dashboard</title>

<!-- Tell the browser to be responsive to screen width -->

<meta content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no" name="viewport">

<!-- Bootstrap 3.3.5 -->

<link rel="stylesheet" href="bootstrap/css/bootstrap.min.css">

</head>

<body class="hold-transition skin-blue-light sidebar-mini">

<div class="wrapper">

<?php

include("header.php");

include("sidebar.php");

?>

<!-- Content Wrapper. Contains page content -->

<div class="content-wrapper">

<!-- Content Header (Page header) -->

<section class="content-header">

<h1>

Dashboard

<small>Control panel</small>

</h1>

<ol class="breadcrumb">

<li><a href="#"><i class="fa fa-dashboard"></i> Home</a></li>

<li class="active">Dashboard</li>

</ol>

</section>

<?php

include("db\_con.php");

$query = "SELECT COUNT(\*) FROM admission\_details;";

$run\_query = mysqli\_query($con, $query);

$total\_students = mysqli\_fetch\_array($run\_query);

$query = "SELECT COUNT(\*) FROM current\_course WHERE stu\_degree = 3";

$run\_query = mysqli\_query($con, $query);

$ug\_students = mysqli\_fetch\_array($run\_query);

$query = "SELECT COUNT(\*) FROM current\_course WHERE stu\_degree = 2";

$run\_query = mysqli\_query($con, $query);

$pg\_students = mysqli\_fetch\_array($run\_query);

?>

<section class="content">

<!-- Small boxes (Total-student STUDENTS) -->

<div class="row">

<div class="col-lg-3 col-xs-6">

<!-- small box -->

<div class="small-box bg-aqua">

<div class="inner">

<h3><?php echo $total\_students[0]; ?></h3>

<p>Total Students</p>

</div>

<div class="icon">

<i class="fa fa-user"></i>

</div>

<a href="#" class="small-box-footer">More info <i class="fa fa-arrow-circle-right"></i></a>

</div>

</div><!-- ./col -->

<div class="col-lg-3 col-xs-6">

<!-- small box (UG STUDENTS)-->

<div class="small-box bg-green">

<div class="inner">

<h3><?php echo $ug\_students[0]; ?><sup style="font-size: 20px"></sup></h3>

<p>UG STUDENTS</p>

</div>

<div class="icon">

<i class="fa fa-user"></i>

</div>

<a href="#" class="small-box-footer">More info <i class="fa fa-arrow-circle-right"></i></a>

</div>

</div><!-- ./col -->

<div class="col-lg-3 col-xs-6">

<!-- small box (PG STUDENTS)-->

<div class="small-box bg-green">

<div class="inner">

<h3><?php echo $pg\_students[0] ?><sup style="font-size: 20px"></sup></h3>

<p>PG STUDENTS</p>

</div>

<div class="icon">

<i class="fa fa-user"></i>

</div>

<a href="#" class="small-box-footer">More info <i class="fa fa-arrow-circle-right"></i></a>

</div>

</div><!-- ./col -->

<div class="col-lg-3 col-xs-6">

<!-- small box (UG STUDENTS)-->

<div class="small-box bg-yellow">

<div class="inner">

<h3>100 <sup style="font-size: 20px"></sup></h3>

<p>STAFFS</p>

</div>

<div class="icon">

<!--<i class="ion ion-stats-bars"></i>-->

</div>

<a href="#" class="small-box-footer">More info <i class="fa fa-arrow-circle-right"></i></a>

</div>

</div><!-- ./col -->

</div><!--./row -->

</section><!-- ./section-content -->

<?php

include("sidepane.php");

?>

</div><!-- /.content-wrapper -->

<?php

include("footer.php");

?>

</div><!-- ./wrapper -->

</body>

</html>

**Add.php:**

<?php

include("../../db\_con.php");

?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<title>Admin - Managestudents</title>

<!-- Custom css For Form Styling -->

<link rel="stylesheet" type="text/css" href="<?php echo admin\_index; ?>dist/css/style.css" />

<!-- Select2 -->

<link rel="stylesheet" href="<?php echo admin\_index; ?>plugins/select2/select2.min.css">

</head>

<body class="hold-transition skin-blue-light sidebar-mini">

<div class="loader"></div><!-- div from loader -->

<div class="wrapper">

<?php

//Include the header.php

include("../../header.php");

//Include the sidebar.php

include("../../sidebar.php");

?>

<div class="content-wrapper">

<section class="content-header">

<h1>Manage Students

<small>Add Student Record</small>

</h1>

<ol class="breadcrumb">

<li><a href="#"></a>Home</li>

<li><a href="#">Manage Students</a></li>

<li>Add Student Record</li>

</ol>

</section><!--./section header -->

<section class="content">

<div class="row">

<div class="panel panel-info">

<div class="panel-heading">Add Student Record</div>

<div class="panel-body" style="font-size:12px;">

<form class="" id="student\_form" name="student" method="post" action="">

<!-- Student Personal Details -->

<div class="frm" id="sf1">

<fieldset>

<legend>Step 1 of 4 <span>(Admission/Personal Details)</span></legend>

<div class="from-group">

<label class="col-lg-2 control-label" for="admission\_no">Admission No <span class="text-danger">\*</span></label>

<div class="col-lg-6">

<input type="text" name="admission\_no" class="form-control input-sm" id="admission\_no" autocomplete="off" />

</div><!-- ./lg6-->

</div><!-- ./form group for admissionno-->

<div class="clearfix" style="height: 10px;clear: both;"></div>

<!-- Admission date -->

<div class="form-group" style="margin-bottom:0px;">

<label class="col-lg-2 control-label" for="admission\_date">Date Of Admission<span class="text-danger">\*</span></label>

<div class="col-lg-6 input-group" style="margin-left:18%;">

<div class="input-group-addon">

<i class="fa fa-calendar"></i>

</div><!-- ./Input group addon -->

<input type="text" class="form-control input-sm" name="admission\_date" style="width:94%;" data-inputmask="'alias':'dd/mm/yyyy'" data-mask />

</div><!-- ./col-lg-6 and input-group-->

<label for="admission\_date" generated="true" class="error" style="margin-left:18%;"></label>

</div><!-- ./form-group for admission\_date -->

<!-- Admission qouta -->

<div class="form-group">

<label class="col-lg-2 control-label" for="admission\_qouta">Admission-Quota <span class="text-danger">\*</span></label>

<div class="col-lg-6 col-sm-5">

<select class="form-control select2 input-sm" name="admission\_quota" data-placeholder="Select a admission\_quota">

<option value="default">--Select--</option>

<option>Councelling</option>

<option>Management</option>

</select>

<label for="religion" generated="true" class="error"></label>

</div>

</div><!-- ./form-group for religion -->

<div class="clearfix" style="height: 10px;clear: both;"></div>

<div class="from-group">

<label class="col-lg-2 control-label" for="rollno">Rollno <span class="text-danger">\*</span></label>

<div class="col-lg-6">

<input type="text" name="rollno" class="form-control input-sm" id="rollno" autocomplete="off" />

</div><!-- ./lg6-->

</div><!-- ./form group for rollno-->

<div class="clear fix" style="height: 10px; clear: both ;"></div>

<div class="form-group">

<label class="col-lg-2 control-label" for="fname">FirstName <span class="text-danger">\*</span></label>

<div class="col-lg-6">

<input type="text" class="form-control input-sm" name="fname" id="fname" autocomplete="off" />

</div><!-- ./lg-6-->

</div><!-- ./form-group for fname -->

<div class="clearfix" style="height: 10px;clear: both;"></div>

<div class="form-group">

<label class="col-lg-2 control-label" for="lname">LastName <span class="text-danger">\*</span></label>

<div class="col-lg-6">

<input type="text" class="form-control input-sm" name="lname" id="lname" autocomplete="off" />

</div><!-- ./lg-6 -->

</div><!-- ./form-group for lname -->

<div class="clearfix" style="height: 10px;clear: both;"></div>

<div class="form-group">

<label class="col-lg-2 control-label" for="gender">Gender <span class="text-danger">\*</span></label>

<div class="col-lg-6">

<label class="Form-label--tick">

<input type="radio" name="gender" class="Form-label-radio" value="male" />

<span class="Form-label-text"> Male</span>

</label>

<label class="Form-label--tick">

<input type="radio" name="gender" class="Form-label-radio" value="female" />

<span class="Form-label-text"> Female</span>

</label>

<label class="Form-label--tick">

<input type="radio" name="gender" class="Form-label-radio" value="others" />

<span class="Form-label-text"> Others</span>

</label>

<label for="gender" generated="true" class="error" style="font-weight:bold;color:#FF0000;margin-left:7%;"></label>

</div><!-- ./lg-6-->

</div><!-- ./form-gropu for Gender -->

<div class="clearfix" style="height: 10px;clear: both;"></div>

<!-- Date Of birth -->

<div class="form-group">

<label class="col-lg-2 control-label" for="dob">Date of Birth <span class="text-danger">\*</span></label>

<div class="col-lg-6 input-group" style="margin-left:18%;">

<div class="input-group-addon">

<i class="fa fa-calendar"></i>

</div><!-- ./Input group addon -->

<input type="text" class="form-control input-sm" name="dob" style="width:94%;" data-inputmask="'alias':'dd/mm/yyyy'" data-mask />

</div><!-- ./col-lg-6 and input-group-->

<label for="dob" generated="true" class="error" style="margin-left:18%;"></label>

</div><!-- ./form-group for DOB -->

<!-- Religion -->

<div class="form-group">

<label class="col-lg-2 control-label" for="religion">Religion <span class="text-danger">\*</span></label>

<div class="col-lg-6 col-sm-5">

<select class="form-control select2 input-sm" name="religion" data-placeholder="Select a Religion">

<option value="default">--Select--</option>

<option>Hindu</option>

<option>Christian</option>

<option>Islam</option>

<option>Jain</option>

<option>Sikhism</option>

</select>

<label for="religion" generated="true" class="error"></label>

</div>

</div><!-- ./form-group for religion -->

<div class="clearfix" style="height: 10px;clear: both;"></div>

<!-- Community -->

<div class="form-group">

<label class="col-lg-2 control-label" for="Community">Community <span class="text-danger">\*</span></label>

<div id="totopscroller"></div>

<?php

//Include the footer page

include("../../footer.php");

//Include the sidepane page

include("../../sidepane.php");

?>

</div><!-- ./wrapper -->

</body>

</html>

**CHAPTER 5**

**TESTING**

**5.1 Testing Strategy**

A testing strategy is a general approach to the testing process rather than a method of devising particular system or component tests.

The main objectives of testing it as follows:

* Testing is the process of executing program with the intent of finding error.
* A good test is one that has a high probability of finding an as yet undiscovered error.
* A successful test is one that uncovers an as yet undiscovered error.
* To affirm the quality of the project.
* To find and eliminate any errors from previous stages.
* To validate the software and to eliminate the operation.
* Reliability of the system.

**5.1.1 Unit Testing**

Unit testing is one of the types of testing strategies that focuses on testing the software module and its boundaries.

Specify the minimum degree of comprehensiveness desired. Identify the techniques which will be used to judge the comprehensiveness of the testing effort. Specify any additional completion criteria. The techniques to be used to trace requirements should be specified.

Unit testing is the phase of testing that testing the basic functionality and the structure of the code. The module is testing to ensure that information flows is proper into and out of the program under unit testing. Each of the individual reports has been tested extensively to determine whether the required output has been obtained or not.

Unit testing is done every individual unit in the system.

In Report Builder, I included login and staff Login creational module as a sample for unit testing. It registers new staff and student into the system. Once the registration for the staff is successful, the details will be stored into the system and the staff Username will be generated and send mail to the staff. Open Load Excel page and Excel name will be displayed left text box Display the message as ‘The file has been loaded’ and data to be displayed Table.

**5.1.1.1 TEST PLAN**

***Table 5.1 – Test Plan***

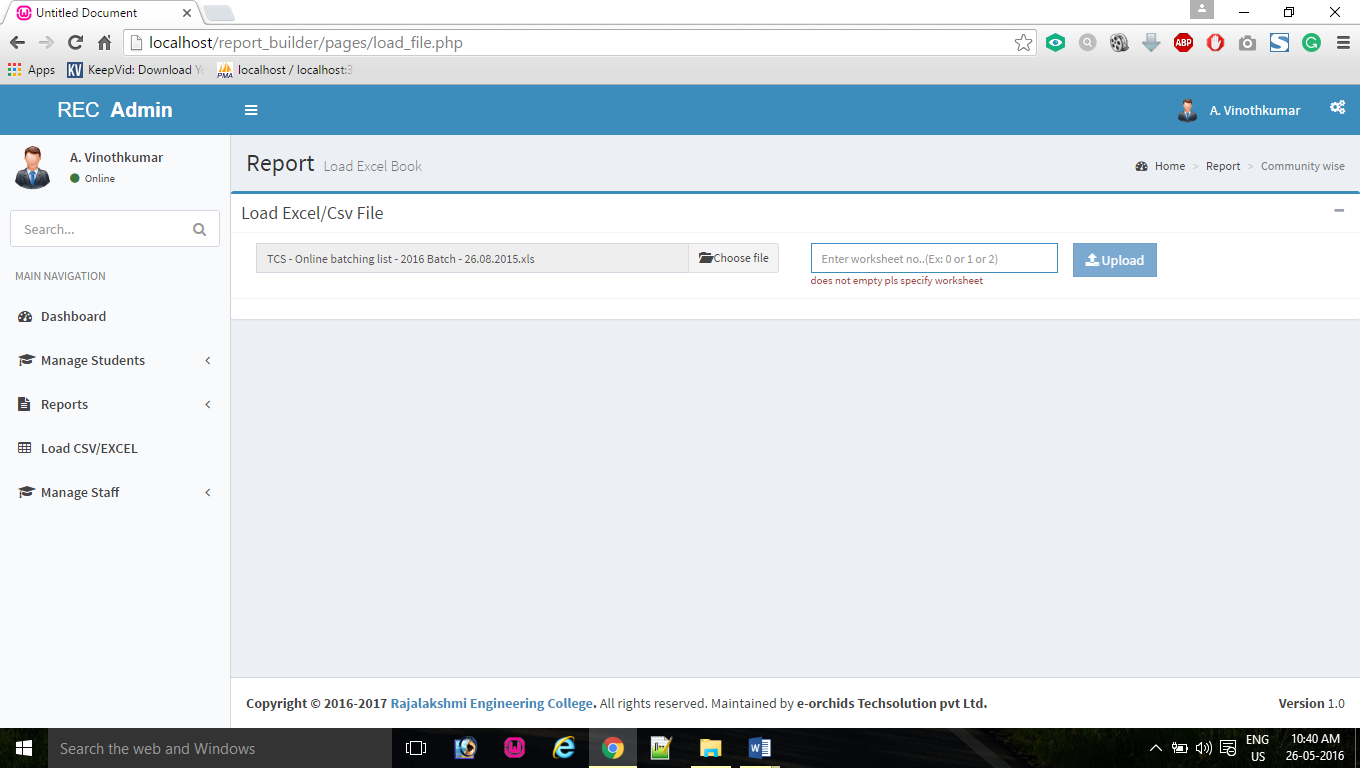
|  |  |  |
| --- | --- | --- |
| **Module Name** | **Type of Test** | **Description about the test** |
| Manage Students | Unit test,  Functional test | If the super admin enters the valid student data. It will show next screen. And store data into database  If invalid details given it will show the error message |
| Manage staff | Unit test,  Functional test | If the staff enters the valid username and password. It will show next screen.  If invalid details given it will show the error message |
| Load Excel | Unit test,  Functional test | If the staff load Excel file. It will show Excel data in page.  If invalid file format given it will show the error message |

**5.1.1.2 Test Cases:**

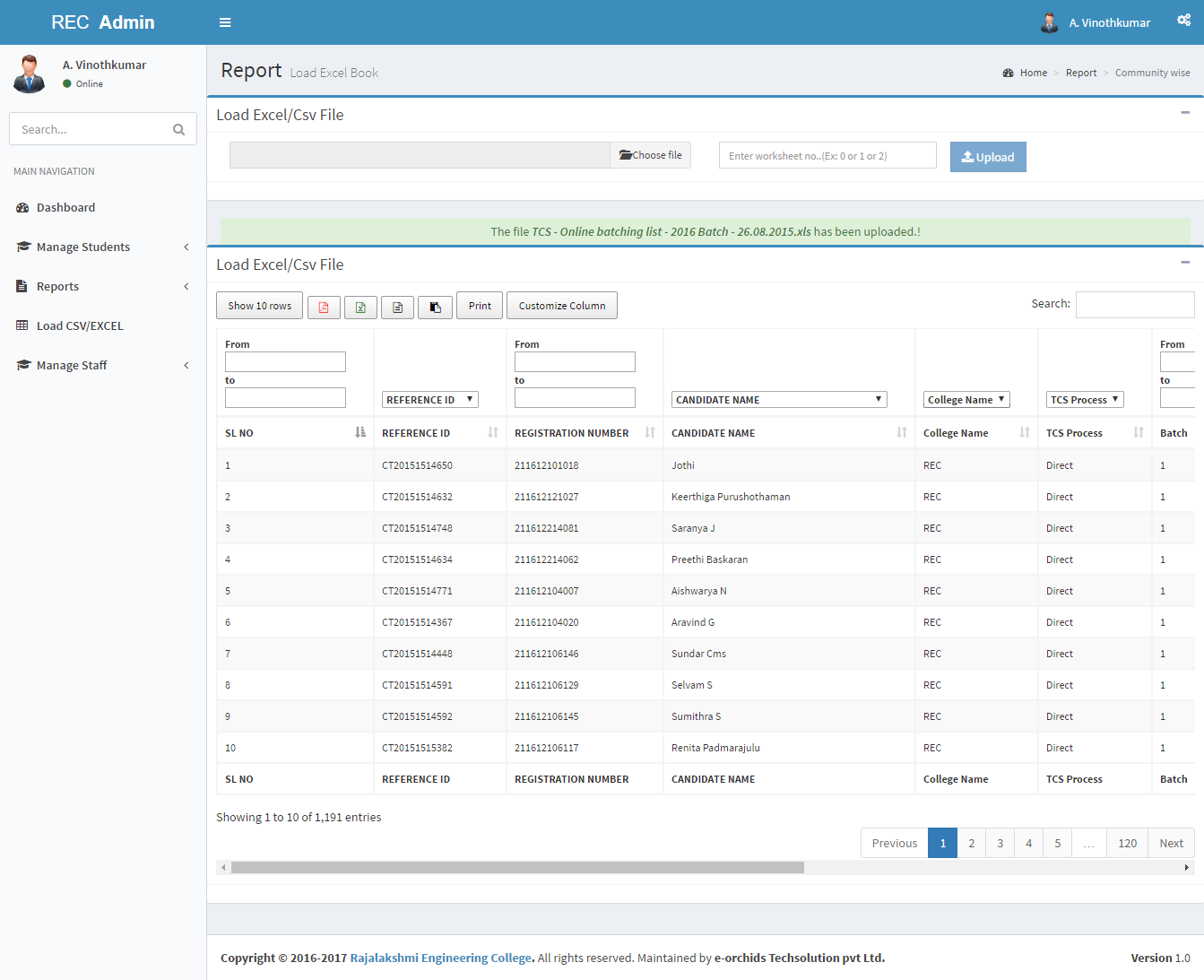
**Test Case1**:

***Table - 5.1 Test case – Load Excel***

|  |  |  |  |
| --- | --- | --- | --- |
| ID | | 1 | |
| Title | | Load Excel Sheet | |
| Priority | | High | |
| Module | | Load Excel/CSV | |
| Execution result | | Success | |
| Purpose | | Load external Excel file for report generation | |
| Created by | | Balakumar B, 2016-05-11 09:40 AM | |
| Test Environment | | WAMP | |
| Pre-conditions: User has Excel/ CSV file with first row as a Header | | | |
|  | Description | | Expected result |
| Step 1 | Go to <http://localhost/report_builder/pages/load_file.php> | | Open Load Excel page |
| Step 2 | Choose Excel file from your PC | | Excel name will be displayed left text box |
| Step 3 | Enter work sheet Number from excel sheet | |  |
| Step 4 | Click ‘upload’ button | | Display the message as ‘The file has been loaded’ and data to be displayed Table |
| Post-conditions: The Excel Data Loaded Successfully. | | | |



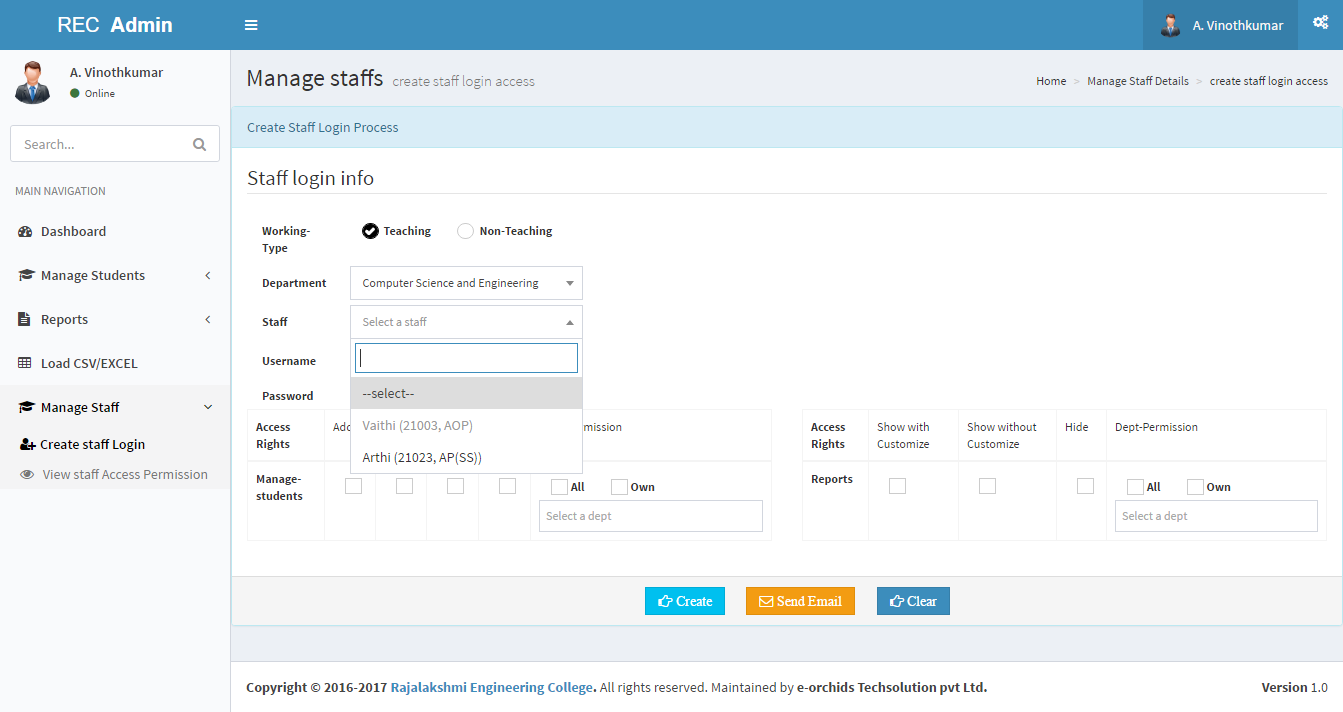
***Fig. 5.1(a) - Invalid Details for Load excel file***

***Fig. 5.1(a) - Valid Details for Load excel file***

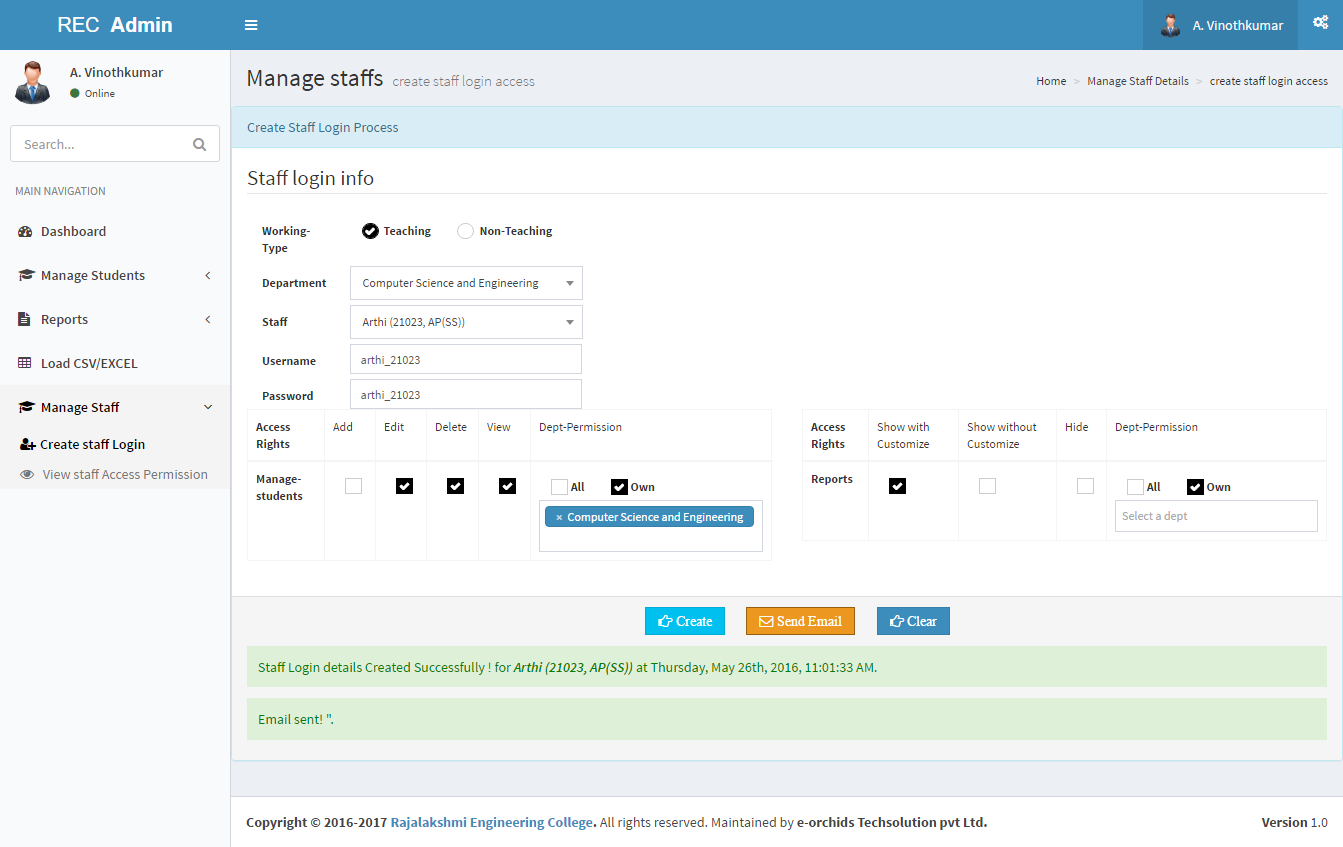
**Test Case2**:

***Table - 5.2 Test case – create staff Login***

|  |  |  |  |
| --- | --- | --- | --- |
| ID | | 2 | |
| Title | | Create staff login | |
| Priority | | High | |
| Module | | Manage staff | |
| Execution result | | Success | |
| Purpose | | Provide username and password in individual login for staff. | |
| Created by | | Balakumar B, 2016-05-11 03:40 PM | |
| Test Environment | | WAMP | |
| Pre-conditions: super-admin only can create staff login and staff details should be there in DB | | | |
|  | Description | | Expected result |
| Step 1 | Go to <http://localhost/report_builder/pages/manage_staffs/add.php> | | Open create staff login page |
| Step 2 | Ensure Choose Teaching or Non-Teaching | | If choosing Teaching load Teaching departments in Department Dropdown List or If choosing non-Teaching Load Non-Teaching departments in Department dropdown. |
| Step 3 | Should be Select Department as ‘Computer science and Engineering’ | | Load staff details in Staff Dropdown list like *Arthi (21023, (AP(SS)),* If staff already have username and password show disabled |
| Step 4 | Ensure Select staff as ‘Arthi (21023, (AOP))’ | | Username and password automatically generated. username as ‘*arthi\_21023*’ and password as ‘*arthi\_21023*’ and automatically filled at Username and password fields |
| Step 5 | Assign access permissions as ‘edit, delete, view’ for manage students module and select dept.-permission as ‘own’ and assign access permission as ‘show without customize’ for reports modules and select dept.-permission as ‘own’. | |  |
| Step 6 | Click ‘create’ button | | Show response message as ‘staff Login created successfully for Arthi(21023, AP(SS))’ |
| Step 7 | Click ‘Send Email’ button | | Show response message as ‘mail sent successfully’ |
| Step 8 | Click ‘clear’ button | | Clear all selected and filled fields. |
| Post-conditions: Staff login details and access permission details stored in database successfully | | | |



***Fig. 5.2(a) - Invalid Details for staff login creational***

****

***Fig. 5.2(a) - Valid Details for staff login creational***

**5.1.2 Validation Testing**

Validation testing is used to uncover and correct interfacing errors. It succeeds when the software functions in the manner that is reasonably expected. The various inputs in the system is checked and validated in the client side and provide response before sending the data to the server.

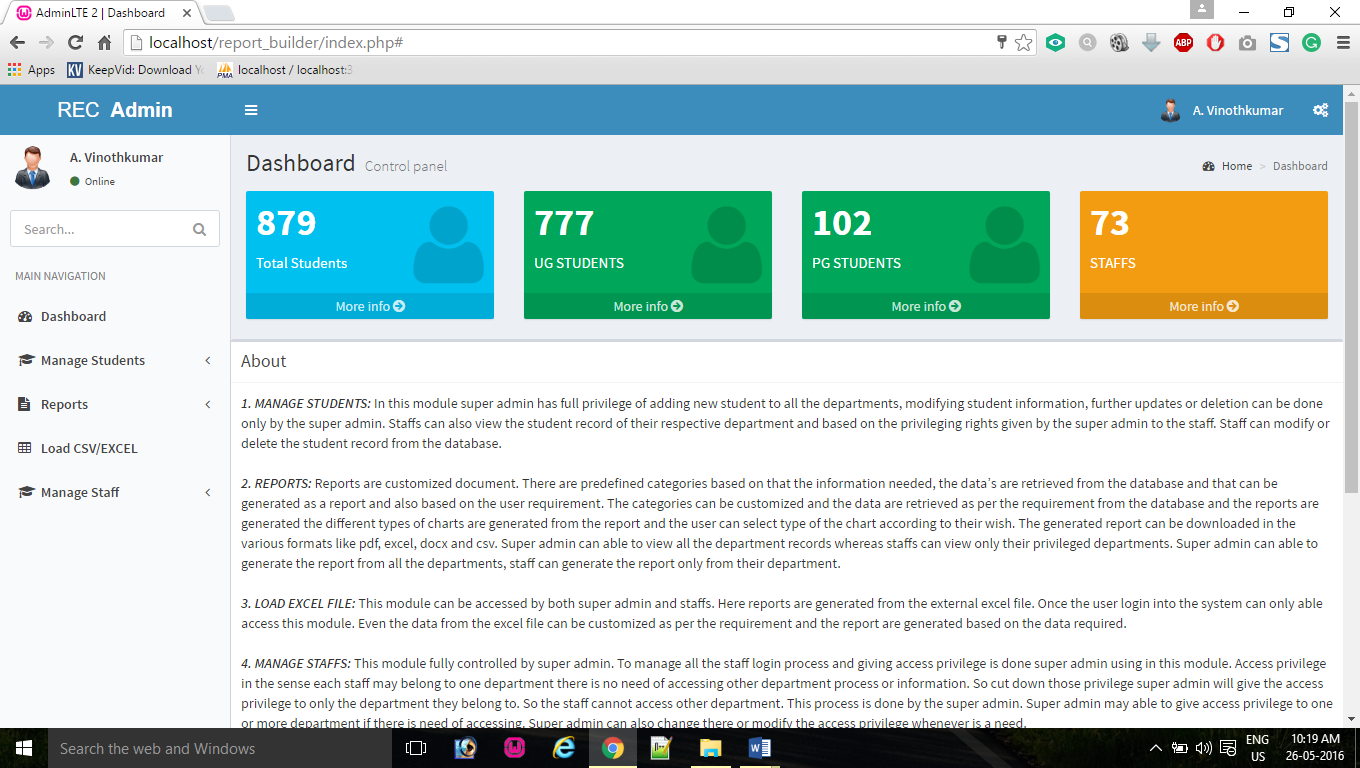
Validation testing plays a vital role in maintaining data consistency and avoiding incorrect data to store into the system.

Validation test cases for admin login is given below

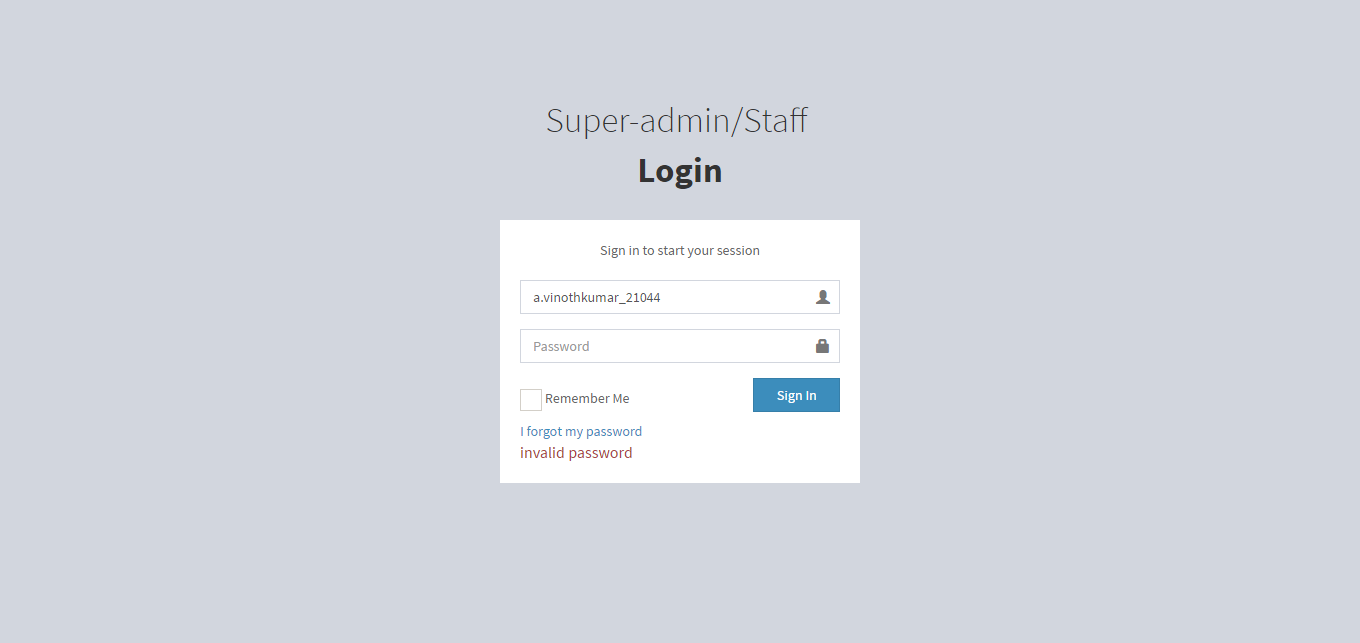
**Test Case 3:**

***Table 5.3 - Test case- Staff Login***

|  |  |  |  |
| --- | --- | --- | --- |
| ID | | 3 | |
| Title | | Log in | |
| Priority | | High | |
| Execution result | | Success | |
| Purpose | | Verify Login with valid login credential | |
| Created by | | Balakumar B, 2016-05-12 10:40 AM | |
| Test Environment | | WAMP | |
| Pre-conditions: User has valid username and password | | | |
|  | Description | | Expected result |
| Step 1 | Open a Browser | | The browser opens |
| Step 2 | Go to <http://localhost/report_builder/> | | The Login page is shown |
| Step 3 | Enter ‘[*a.vinothkumar\_21044*](http://localhost/phpmyadmin/sql.php?db=report_builder1&table=login_info&pos=0&sql_query=SELECT+%2A+FROM+%60report_builder1%60.%60login_info%60+WHERE+%60username%60+%3D+%27a.vinothkumar_21044%27&token=0bdd5227658c8b2462d2505a3b9296fa)’ as the username and ‘[*a.vinothkumar\_21044*](http://localhost/phpmyadmin/sql.php?db=report_builder1&table=login_info&pos=0&sql_query=SELECT+%2A+FROM+%60report_builder1%60.%60login_info%60+WHERE+%60username%60+%3D+%27a.vinothkumar_21044%27&token=0bdd5227658c8b2462d2505a3b9296fa)’as the password. | |  |
| Step 4 | Click ‘sign in’ button | | The System logs in to your request  Redirect Dashboard page/home page |
| Post-conditions: User is validated with database and successfully login to account. The account session details are logged in database | | | |



***Fig. 5.3(a)–Staff Login, Valid Details***

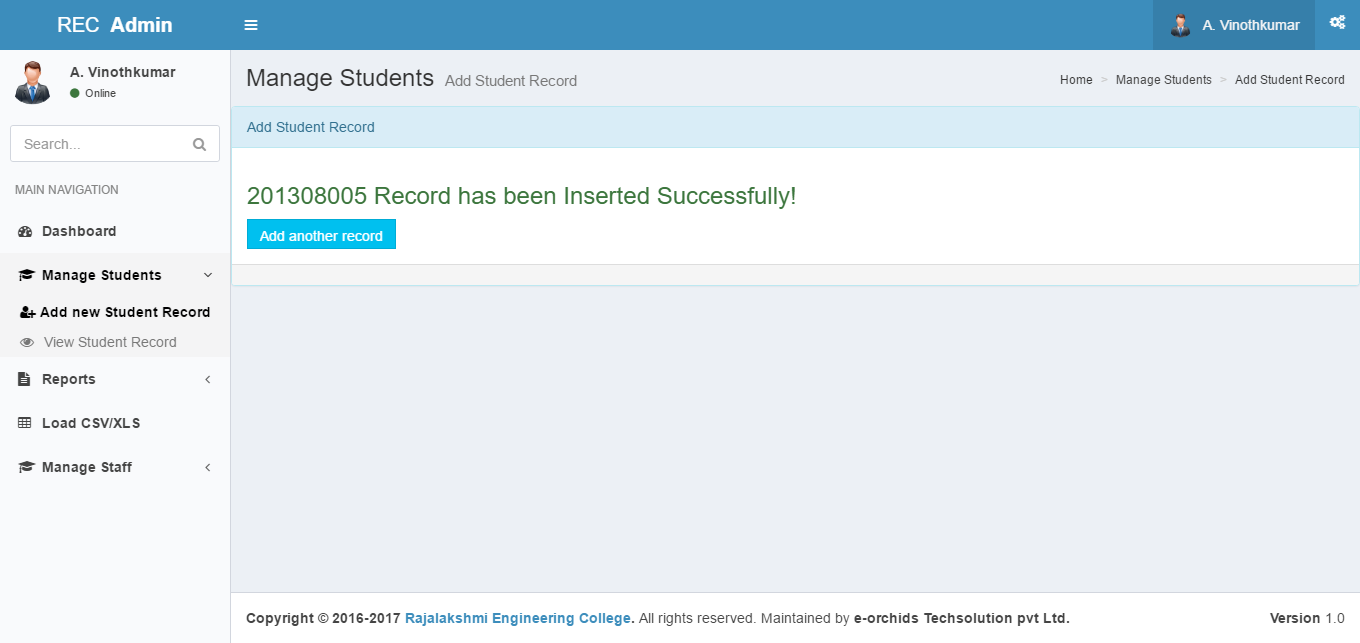
****

***Fig. 5.3(b)–Staff login, Invalid Data***

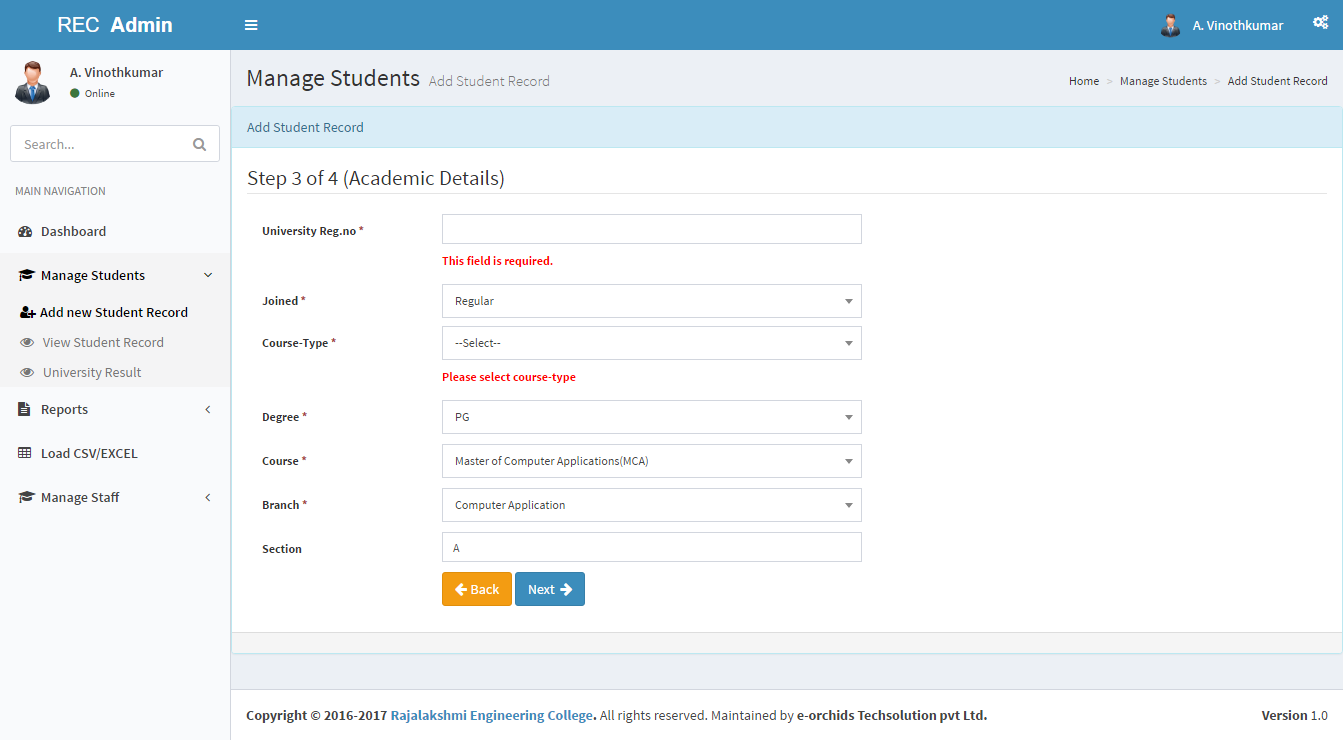
**Test Case 4:**

***Table 5.4 - Test case – Add new Student Record***

|  |  |  |  |
| --- | --- | --- | --- |
| ID | | 4 | |
| Title | | Add Student Individual record | |
| Priority | | Medium | |
| Module Name | | Manage Students | |
| Execution result | | Success | |
| Purpose | | Store student personal, contact and academic details | |
| Created by | | Balakumar B, 2016-05-11 03:40 | |
| Test Environment | | WAMP | |
| Pre-conditions: User has complete record from student like personal details, contact details, academic details and user has permission for Add new student | | | |
|  | Description | | Expected result |
| Step 1 | Go to <http://localhost/report_builder/pages/manage_students/add.php> | | The Add student record page shown |
| Step 2 | Enter student personal details, admission details for admission no, admission date, and select admission quota then enter roll no, first name, last name, etc... And all fields are mandatory except mother’s maiden name and languages known. | |  |
| Step 3 | Click ‘Next’ Button | | Doesn’t show error message and show contact details fields |
| Step 4 | Enter student contact details for email, parent email, mobile, parent mobile, and present address, permanent address contains house no, street, area, city, district, state, country and pin code. these all fields are mandatory and student email and parent email shout not be same | |  |
| Step 5 | Click ‘Next’ Button | | Doesn’t show error message and show current academic details fields. |
| Step 6 | Enter student current course details like university register number, and select join mode, course-type, degree, course, and branch and enter section. All fields are mandatory except section. | |  |
| Step 7 | Click ‘Next’ Button | | Doesn’t show error message. And show previous academic details fields |
| Step 8 | Default show X degree fields if you want add one more degree record click Add row button for top of table, dynamically add one more degree fields and then select degree then enter details. If not need one more degree fields click Remove row button dynamically remove bottom of the row fields | |  |
| Step 9 | Click ‘submit’ Button | | Doesn’t show error message and display response message for Student record successfully Inserted. |
| Post-conditions: Student all details stored in database | | | |

****

***Fig. 5.4(a) – Add new student record, Valid Details***

****

***Fig. 5.4(b) – Add new student record, Invalid Data***

**5.1.3 CYCLOMATIC COMPLEXITY**

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of the software implementation. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs (errors or other defects).

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. Path testing is a structural method for unit testing. Its goal is to ensure that all statements in the program, and both sides of every (two-sided) test in the program, are executed or followed by at least one test. It can be applied to source code or to reasonably detailed pseudo code, provided that all tests and loops in the source code are explicitly shown in the pseudo code as well. Path testing includes two major steps:

1. Use the source code to produce a Flow Graph

2. Using the flow graph and source code as references, produce a set of test for the given program.

Consider the following source code which authenticate the users and provides them respective functions.

**S1:** $stu\_id=$\_POST [“stu\_rollno”];

**S2:** $stu\_name=$\_POST [“stu\_name”];

**S3:** $email=$\_POST [“email”];

**S4:** $stu\_dob=$\_POST [“stu dob”];

**S5:** $address=$\_POST [“address”];

**S6:** $state=$\_POST [“state”];

**S7:** $pin code=$\_POST [“pincode”];

**S8:** $country=$\_POST [“country”];

**S9:** $district=$\_POST [“district”];

**S10:** $mob\_no=$\_POST [“mob\_no”];

**C1:** if (empty ($name))

**S11:** echo “Please enter name”;

**C2:** else if (empty ($address))

**S12:** echo “Please enter address”;

**C3:** else if (empty ($email))

**S13:** echo “Please enter email address”;

**C4:** else if (empty ($mobile))

**S14:** echo “Please enter mobile number”;

The flow – graph for the above code is shown in Fig 5.5

S1

S5

S4

S4

S3

S2

C4

C3

S10

S9

S8

S7

S12

S6

S13

S11

C2

S14

C1

No Yes

No Yes

No Yes

Yes

No

END

*Fig 5.5 Flow**Graph – Basic Path Testing*

**Path 1** S1-S2-S3-S4-S5-S6-S7-S8-S9-S10-C1-S11

**Path 2** S1-S2-S3-S4-S5-S6-S7-S8-S9-S10-C2-S12

**Path 3** S1-S2-S3-S4-S5-S6-S7-S8-S9-S10-C3-S13

**Path 4** S1-S2-S3-S4-S5-S6-S7-S8-S9-S10-C4-S14

Cyclomatic complexity is a source code complexity measurement that is being correlated to a number of coding errors. It is calculated by developing a Control Flow Graph of the code that measures the number of linearly-independent paths through a program module.

Lowe the program’s cyclomatic complexity, lower the risk to modify and easier to understand. It can be represented using the below formula:

Cyclomatic complexity = E- N + P,

Where,

E = number of edges in the flow graph.

N = number of nodes in the flow graph.

P = number of nodes that have exit points.

The Cyclomatic complexity is calculated using the above control flow diagram that shows twenty nodes (shapes) and twenty one edges (lines), hence the cyclomatic complexity is derived as.

Cyclomatic complexity = 15-4+1

= 10

**CHAPTER 6**

**CONCLUSION & FUTURE ENHANCEMENT**

**6.1 Conclusion:**

Thus the Report builder provides an interface between User and the system. The application reads Excel book data and Database record then generate the Report from user request to the web server.

**6.2 Future Enhancement**

This system can further be enhanced by including the following are

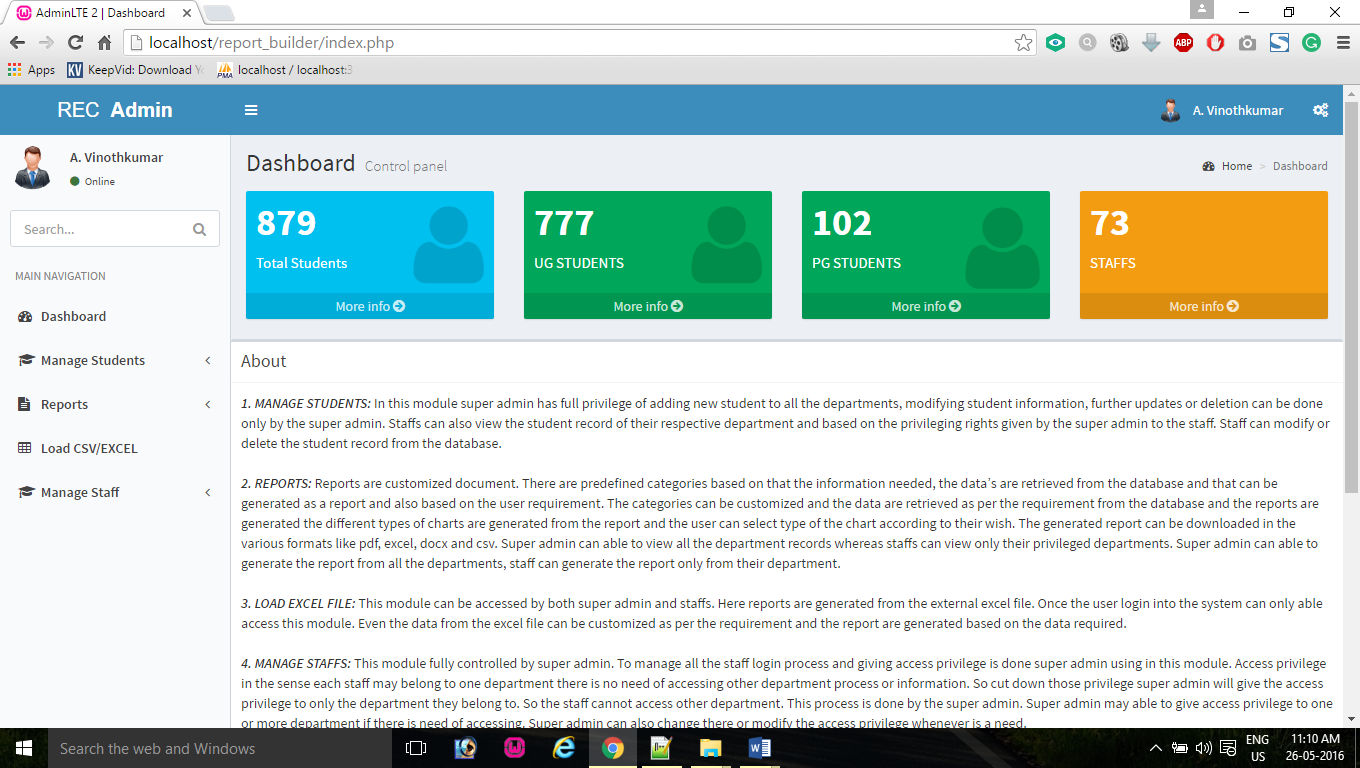
Up until now the builder is developed only for student purposes. This report builder can be extensively developed for an organization or for the whole college including staff, employees, library, and Accounts data and others can also be used in this report builder.

**CHAPTER 7**

**APPENDICES**

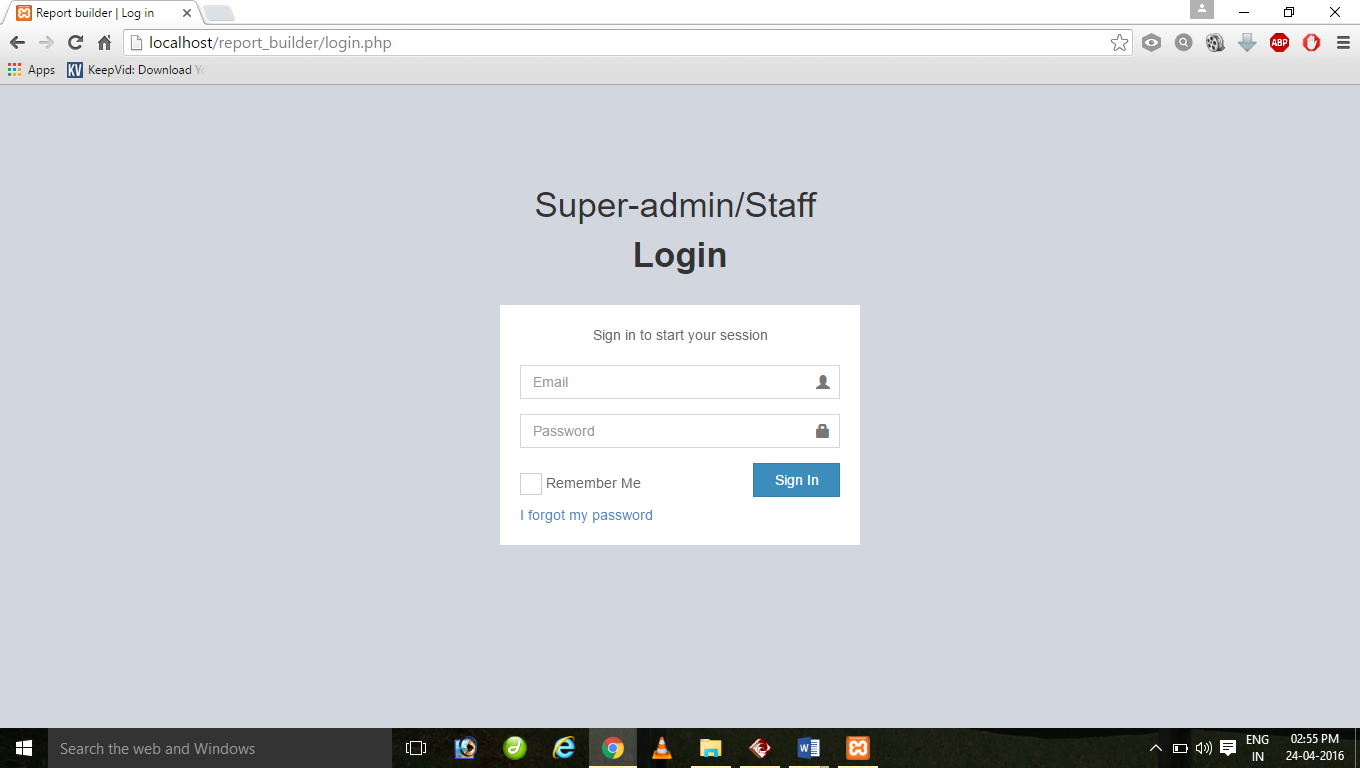
**7.1 Sample screens**

**FIG 7.1.1 Index page**



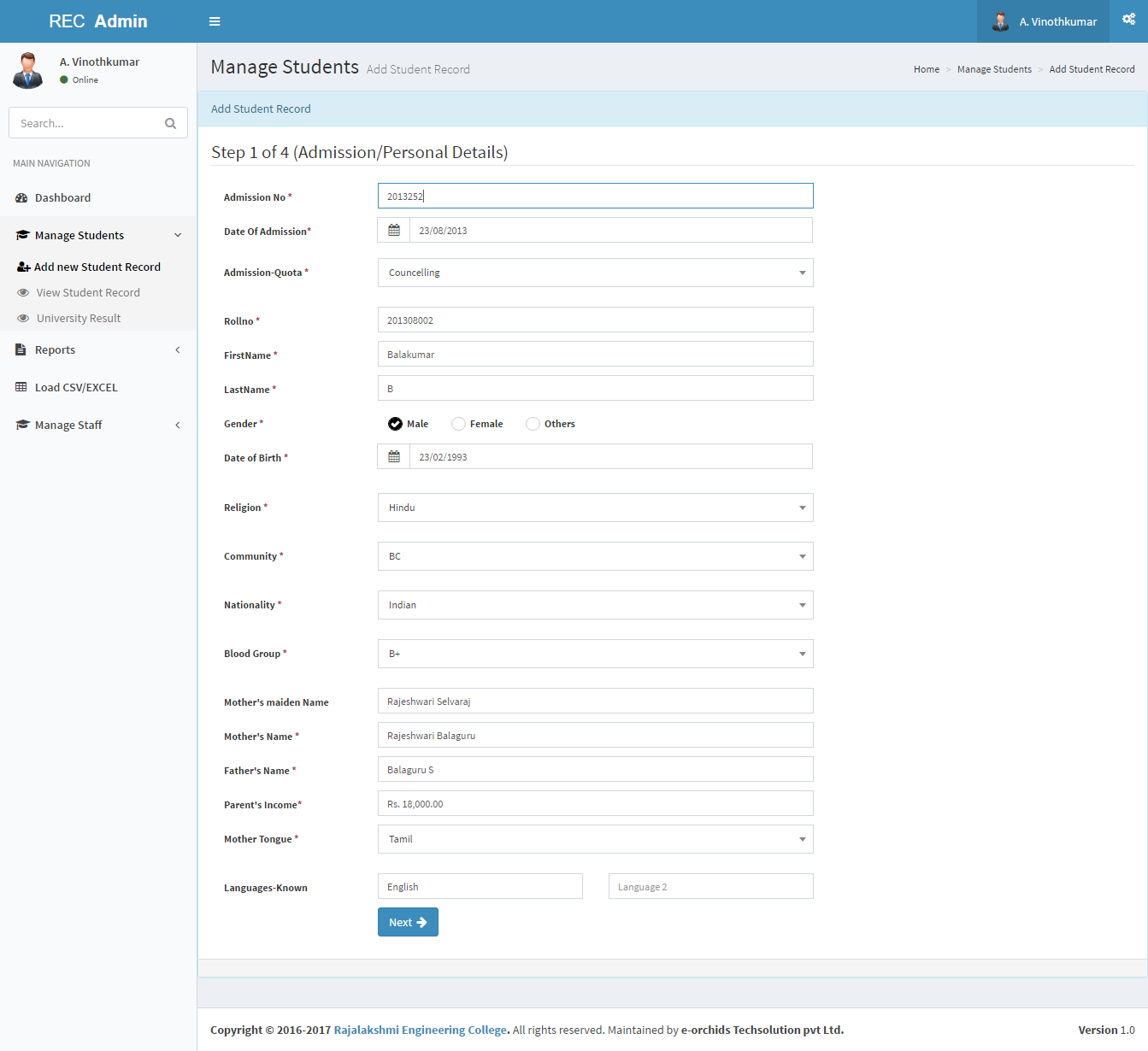
*FIG 7.1.1: Index page*

**FIG 7.1.2 Login page**



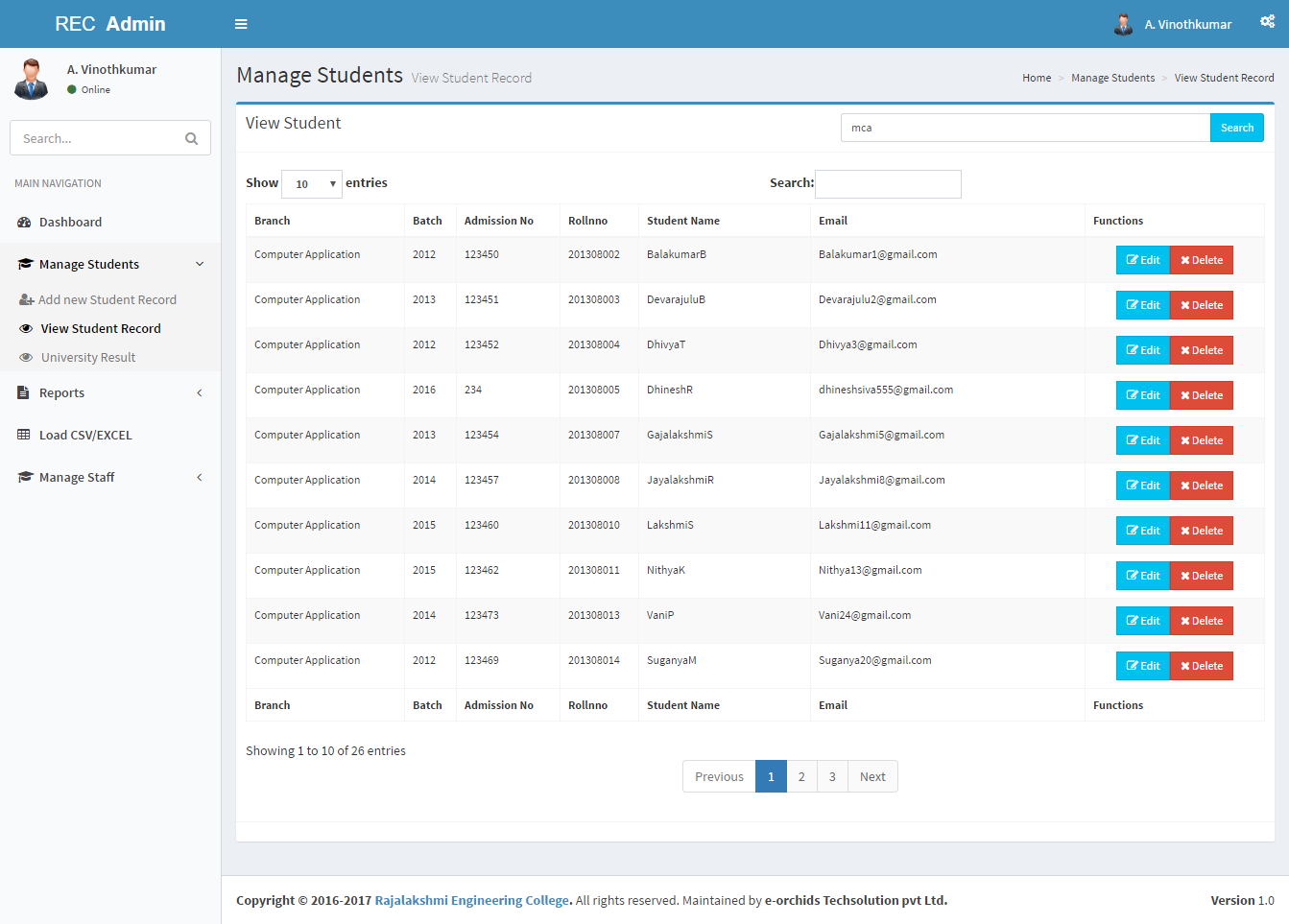
*FIG 7.1.2: Login page*

**FIG 7.1.3 Manage student (add student record):**



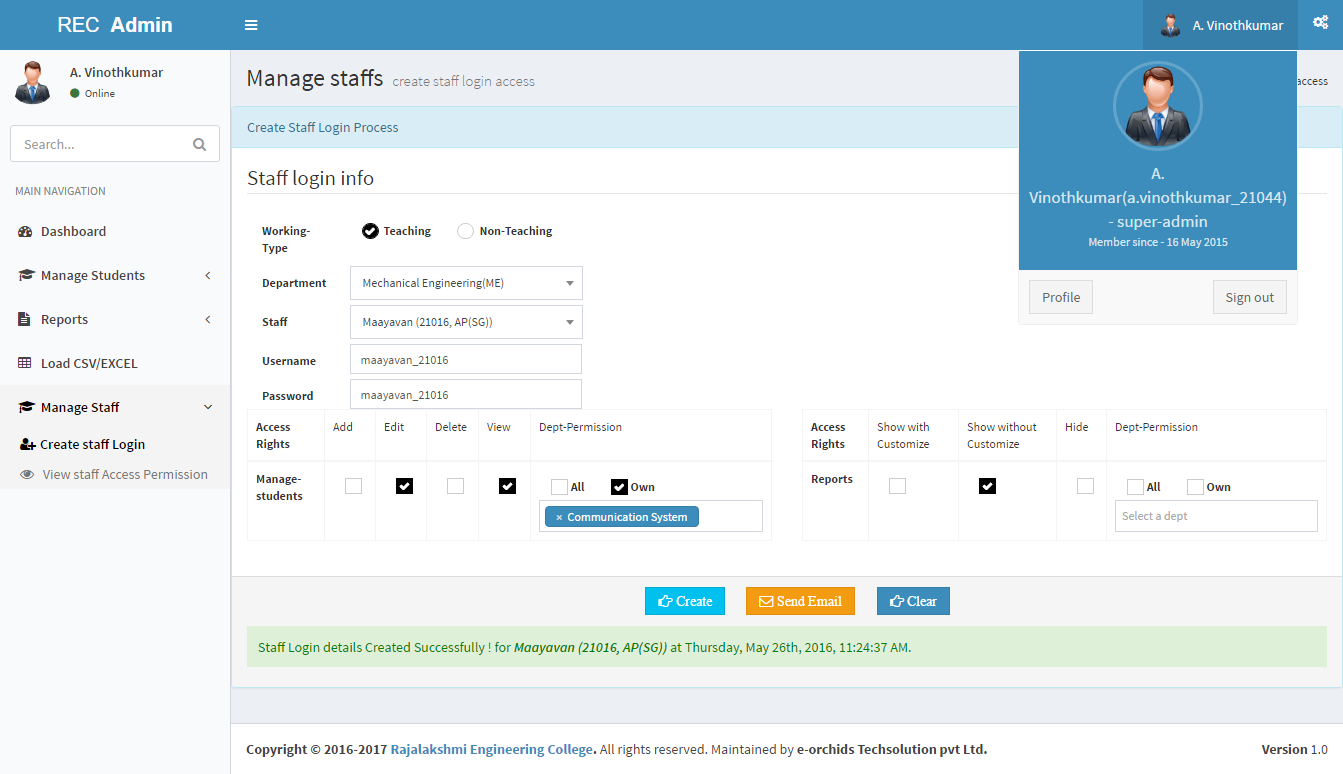
*FIG 7.1.3: Manage students (add student record)*

**FIG 7.1.4 View student:**

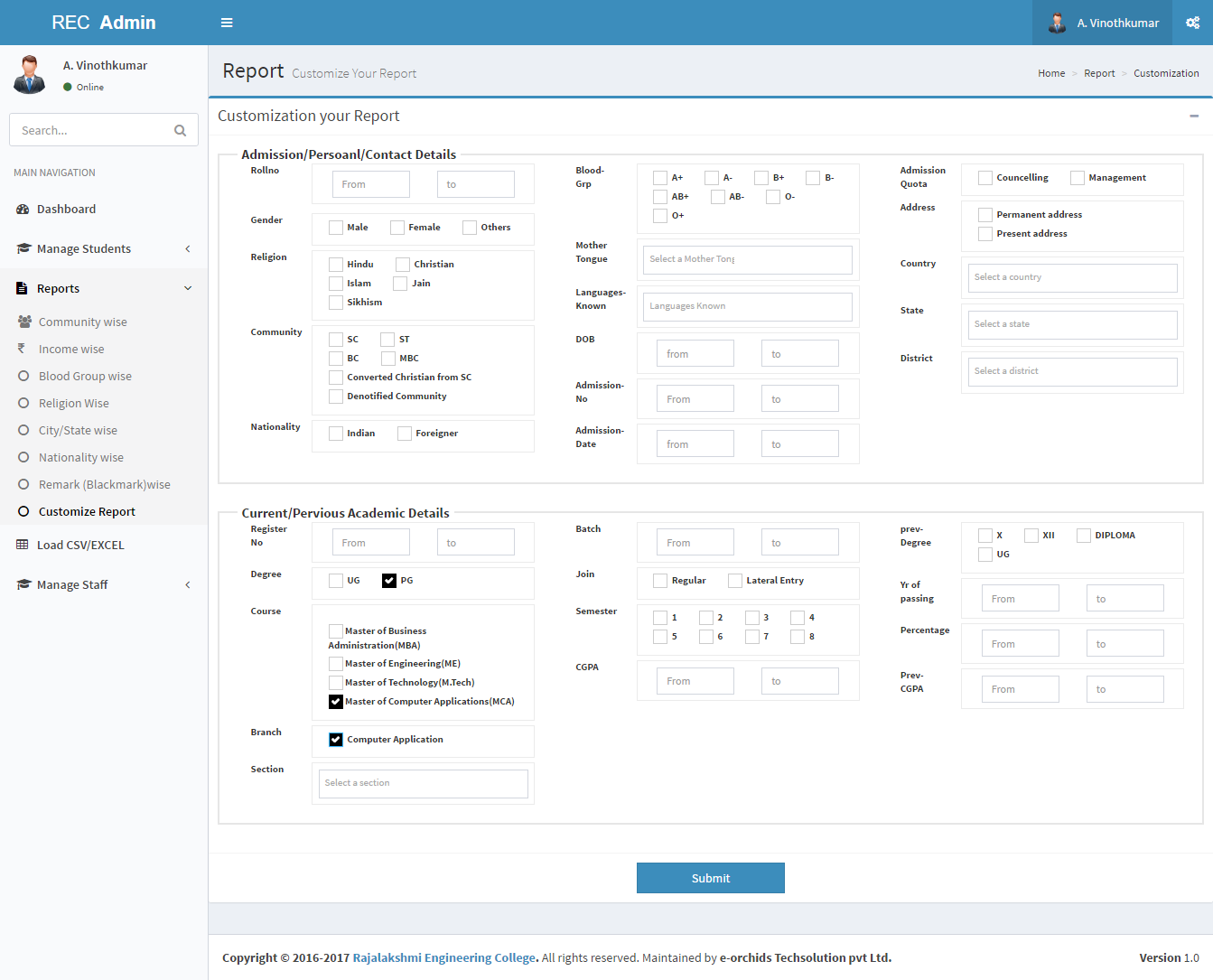


*FIG 7.1.4: View student*

**FIG 7.1.5 Create Staff Login:**

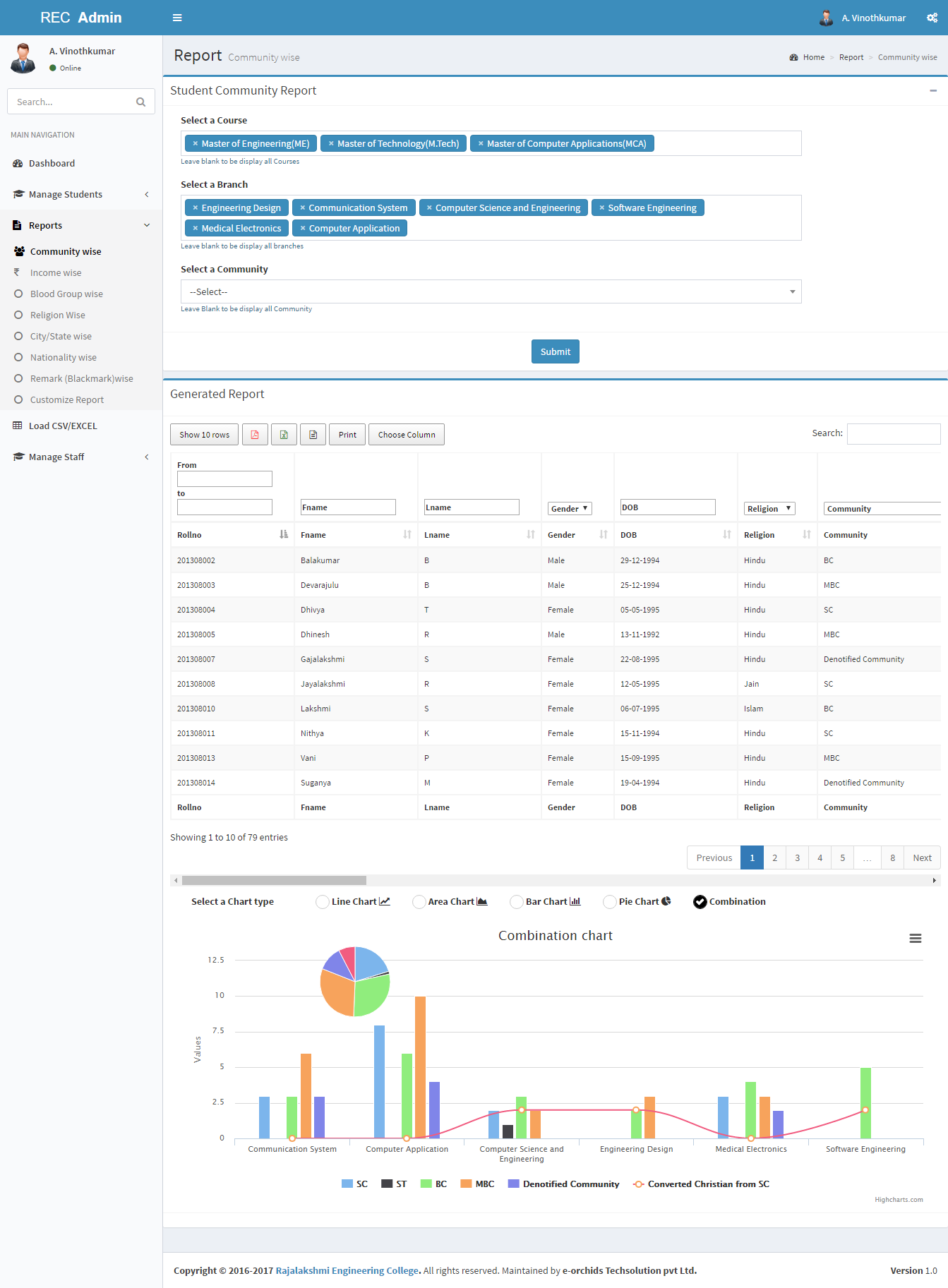
*FIG 7.1.4: Create staff Login*

**FIG 7.1.6 Customize Report:**

****

*FIG 7.1.6: Customize report*

**FIG 7.1.7 Reports:**



*FIG 7.1.6: Reports*

**CHAPTER 8**

**REFERNCES**

[www.stackoverflow.com](http://www.stackoverflow.com)

<http://api.highcharts.com>

[www.getbootstrap.com](http://www.getbootstrap.com)

<http://www.jqueryrain.com>

<https://www.mysql.com>