**ABSTRACT**

Calculating the Profit and Loss is the everyday activity of workers in Chicken shops. As of workers exercise it by making a note of the student sales day by day in book by writing manually there may be chances that may change details of sales.

Sometimes the book may get lost if book get lost owner may couldn’t get idea about profits or loss. In this paperwork we present a web based project that performs the activity of calculating the sales rates and also storing that details more conveniently and accurately. The web product we proposed reduces paperwork and making the exercise of taking day to day details. Deatils can be added by workers in branches and The details is updated and calculation is done and automatically posted to Owner.

Our application is called **Poultry Bazaar** replaces the use of register books which prevents the use of paper. Here ease to calculate Profits, by taking current date from the system details will be in track. Owner(Admin) can view the details of each branch by selecting the branch and date, it stores history of last 30days.

**CONTENTS**

**Page No.**

1. Introduction…………………………………………………………………………01
   1. Description……………………………………………………………………...01
   2. Existing System…………………………………………………………………
   3. Proposed System…………………………………………………………….
   4. Objectives………………………………………………………………………02
2. Literature Survey...................………………………………………………………07
   1. Android Architecture…………………………………………………………....07
      1. Linux Kernel…………………………………………………………….08
      2. Libraries…………………………………………………………………09
      3. Android Runtime………………………………………………………...09
      4. Application Framework………………………………………………….10
   2. Eclipse……………………………………………………………………18
   3. Spring Framework……………………………………………………………......18
   4. Hibernate Framework…………………………………………………………….19
   5. MySQL...…………………………………………………………………………19
3. Analysis Document ………………………………………………………………….20
   1. SRS Document……………………………………………………………….......20
      1. Hardware Specifications………………………………………………….21
      2. Software Specifications…………………………………………………...21
   2. Functional Requirements…………………………………...…………………….21
   3. Data Flow Diagrams………………………………………………………………23
   4. Flow charts………………………………………………………………………...25
   5. ER Diagram
4. Design Document……………………………………………………………………...28
   1. Modules Overview...………………………………………………………………28
      1. Administrator Module……………………………………………………...28
      2. Staff Module…………………………………………………….……….28
      3. Staff on Android Module…………………………………………………...29
      4. Email and Print Module………………………………………………….
      5. Server Module………………………………………………………………29
5. Program Code…………………………………………………………………………...40
6. System Requirements…………………………………………………………………
   1. System requiremens for Web……………………………………………………
   2. System requiremnts for Android……………………………………………..
7. Testing Process………………………………………………………………………….109
   1. A Software Testing Cycle………………………………………………………….111
   2. Testing Fundamentals………………………………………………………………112
   3. Testing Objectives………………………………………………………………....113
   4. Test Cases…………...…………………………………………………………….114
      1. Unit Testing ………………………………………………………………...114
      2. Integration Testing………………………………………………………….114
      3. System Testing…...……………………………………………………….114
8. Limitations of the Project…………………………………………………………….….121
9. Future Enhancement of the project………………………………………………...…....122
10. Reference…………………………………………………………………….…...…123

**Chapter -1**

**Introduction**

* 1. **Description**

**“**Poultry Bazaar” is software developed for maintaining the sales details on the daily basis. Here the branch workers, will be responsible to submit sales details daily. Each branch will be given with a separate username and password. An accurate detail they submitted is generated here. This system will also help in how much profit they got.

Poultry Bazaar deals with the maintenance of the sales details. Poultry Bazaar replaces the use of registers which prevents the use of paper. In this details reports can be generated and assist selling persons about rates entry and ease to calculate rates to equate losses.

The main purpose of developing this app is to reduce paperwork and also to ease the work like calculation of rates. All the calculation is automated with this applicatoin. The recorded details can be reached the owner instantly.

* 1. **Existing system**
* The Existing system is over calcualtions for selling rate wastages and total sales of the day.
* Here the details will be carried out in the hand written registers.
* It will be a tedious job to maintain the record.
* The human effort is more here.
* Sales details will be reached to owners weekly.
* The retrieval of the information is not as easy as the records are maintained in the hand written registers.
  1. **Proposed System**
* To overcome the drawbacks of the existing system, the proposed system has been evolved.
* This project aims to reduce the paper work and saving time to generate accurate deatails from the dealers.
* The system provides with the simple interface to intaract.
* The efficient reports can be generated by using this proposed system
* With the proposed system it is easy to manage sales history.

**1.4 Objectives**

* To store sales details from individual dealer and send to Admin
* It is trouble free to use for Admin
* It is relatively fast approach to send details
* Poultry Bazaar is highly reliable and can except accurate result from the user
* It also has the simple interface
* How much sold in that day, whether the profit or loss will be calculated and hence efficient report can be generated

**Chapter 4**

**Design Document**

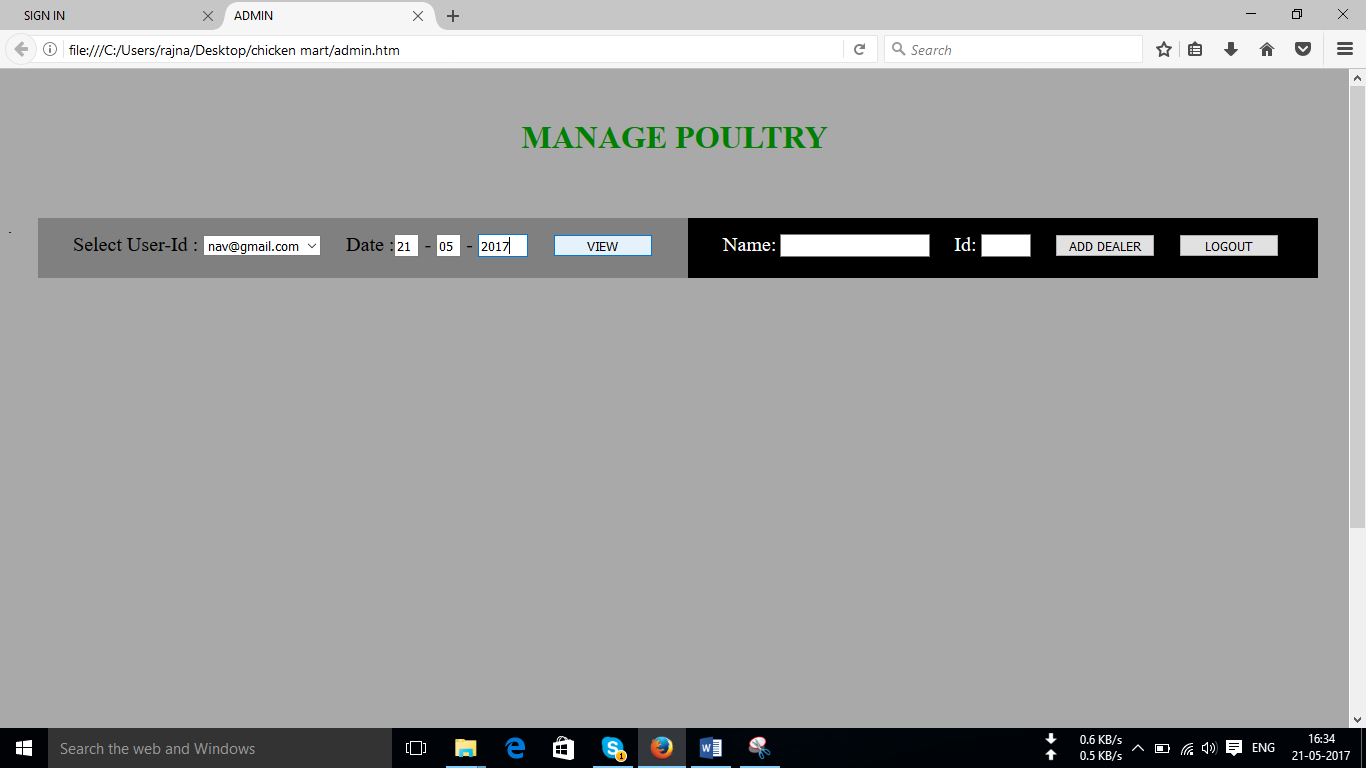
**4.1 Modules**

This project mainly has four modules; they are as follows:

* Administrator Module
* Sign-up&Sign-in Module
* Dealer Module

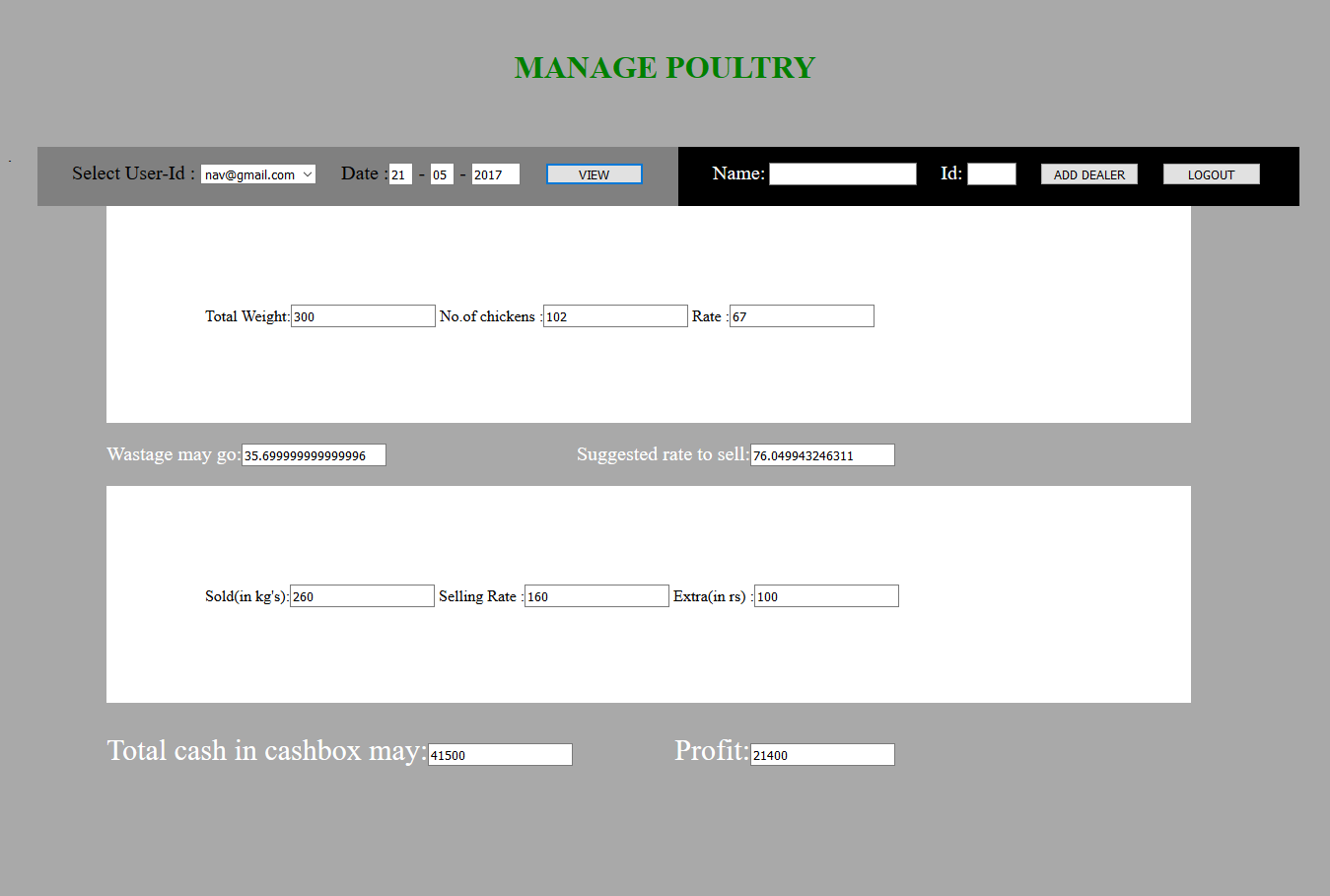
**4.1.1 Administrator Module**

This module is the central authority for this project and it is used by the OWNER to perform administrator actions such as adding new Dealers by giving Id and Name. In addition to adding details, it is also provided with the facility to view deatails. Admin can also get the sales details of each dealers day by day basis. In addition to all these functions admin can also send rate details to dealers. The Admin module works on the system.

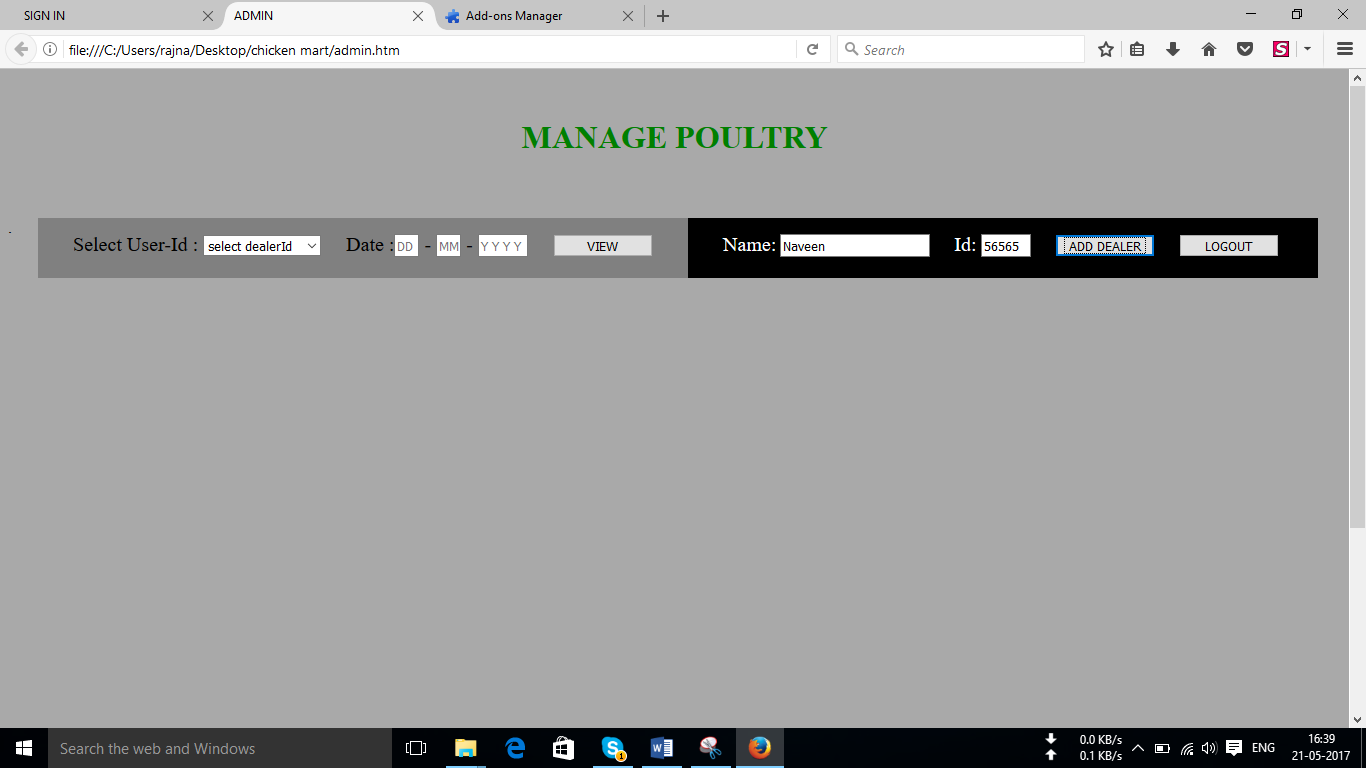


Viewing the Sale details:

* Select the User ID of dealer
* Enter the date which you wanted to see

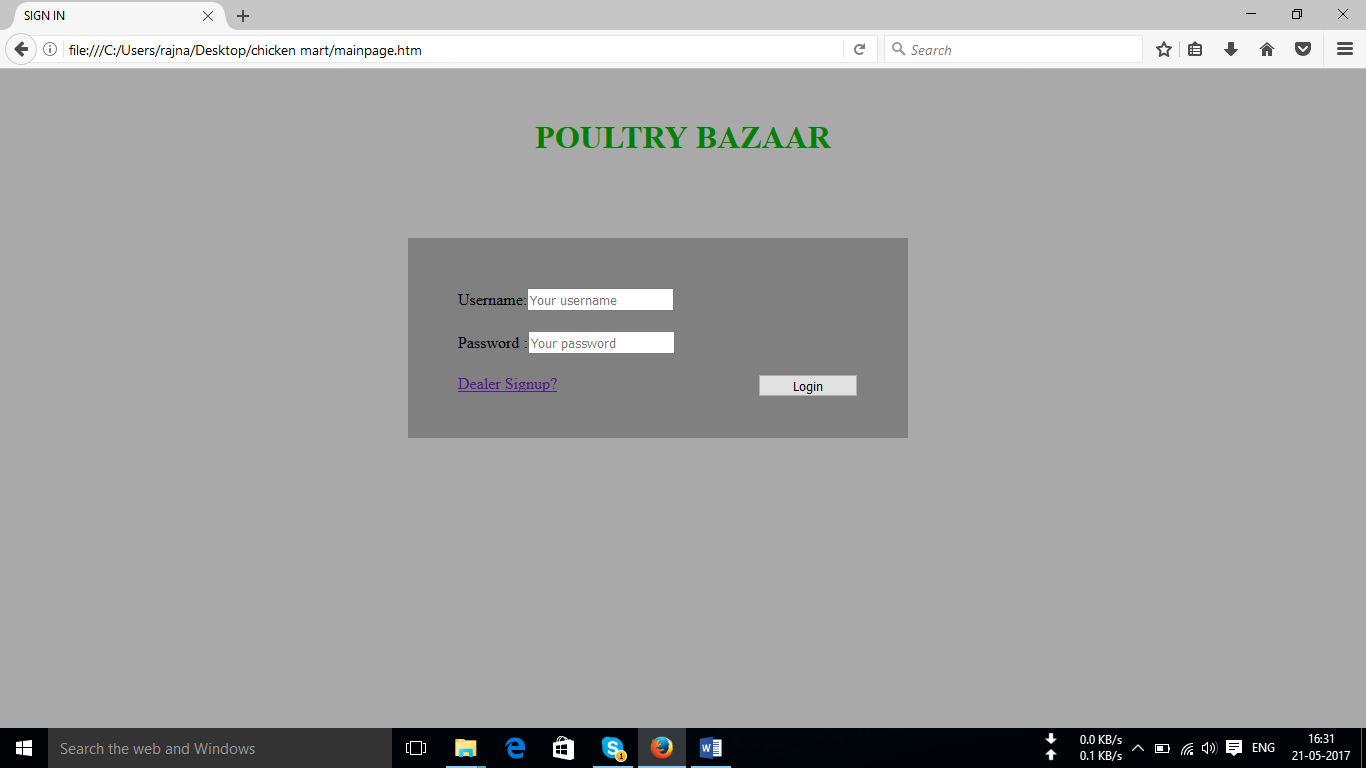


Adding Dealer (Name,Id):



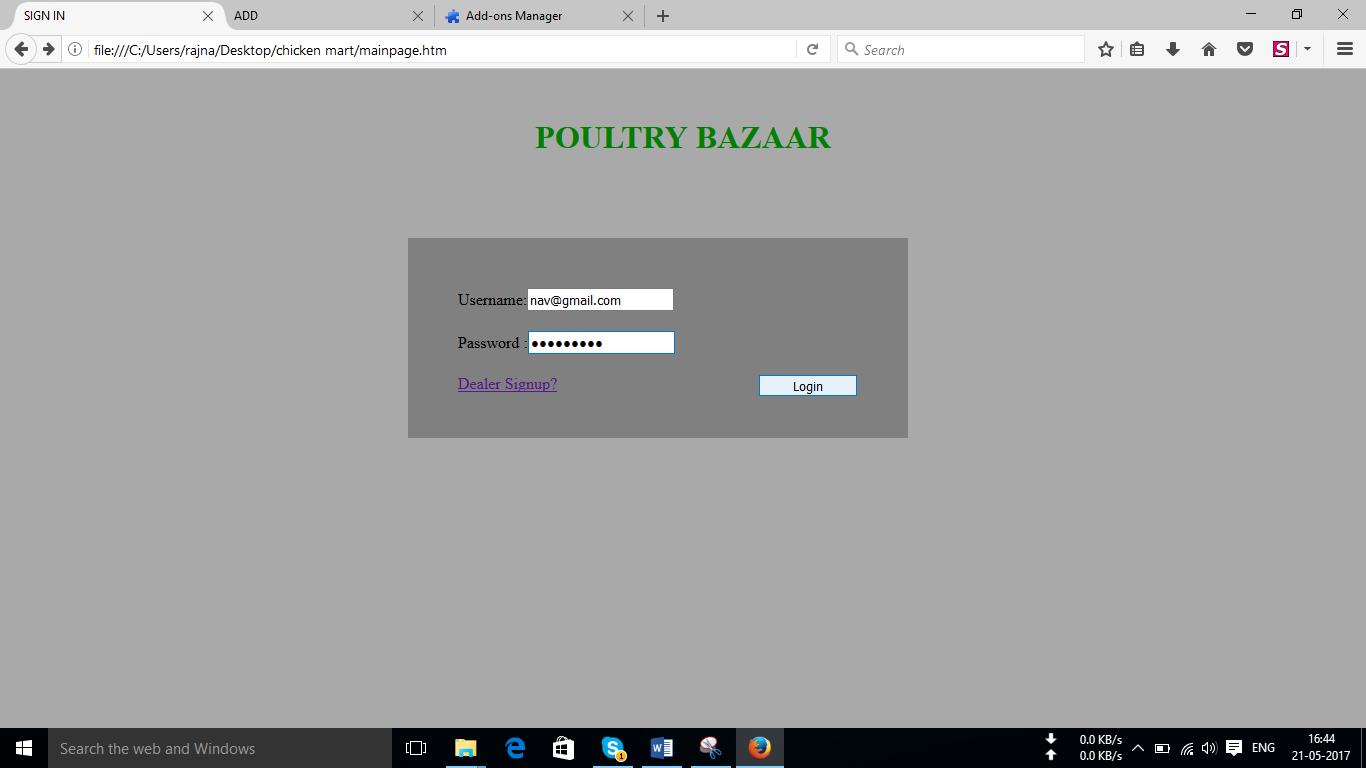
**4.1.2 SignIn & SignUp Module**

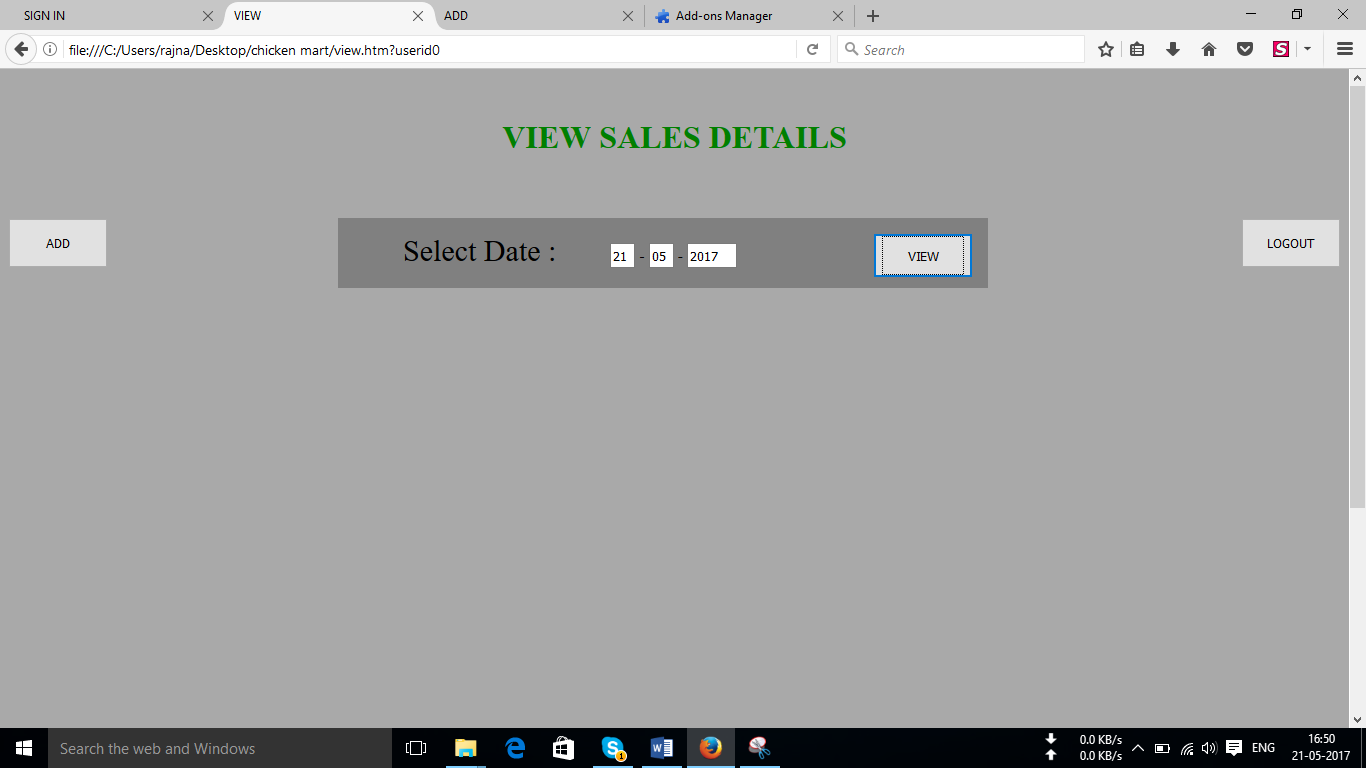
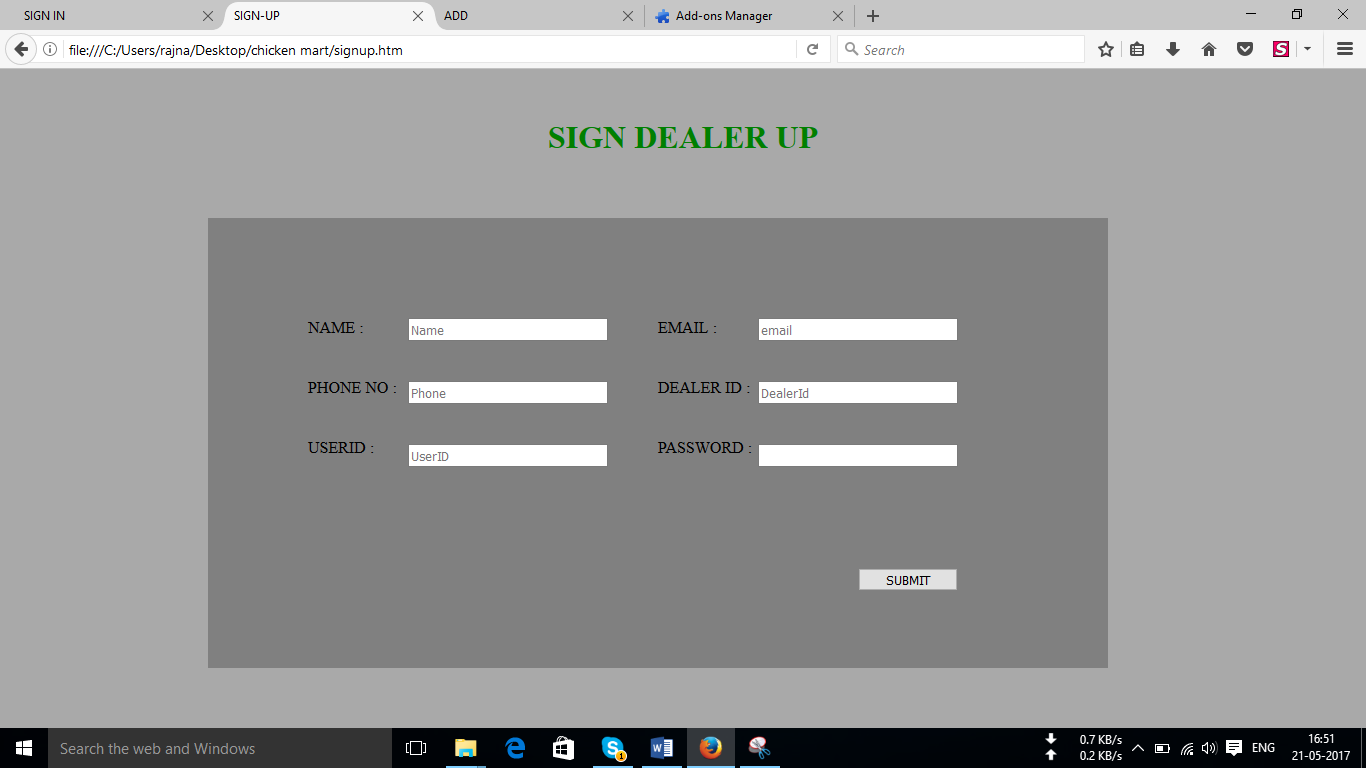
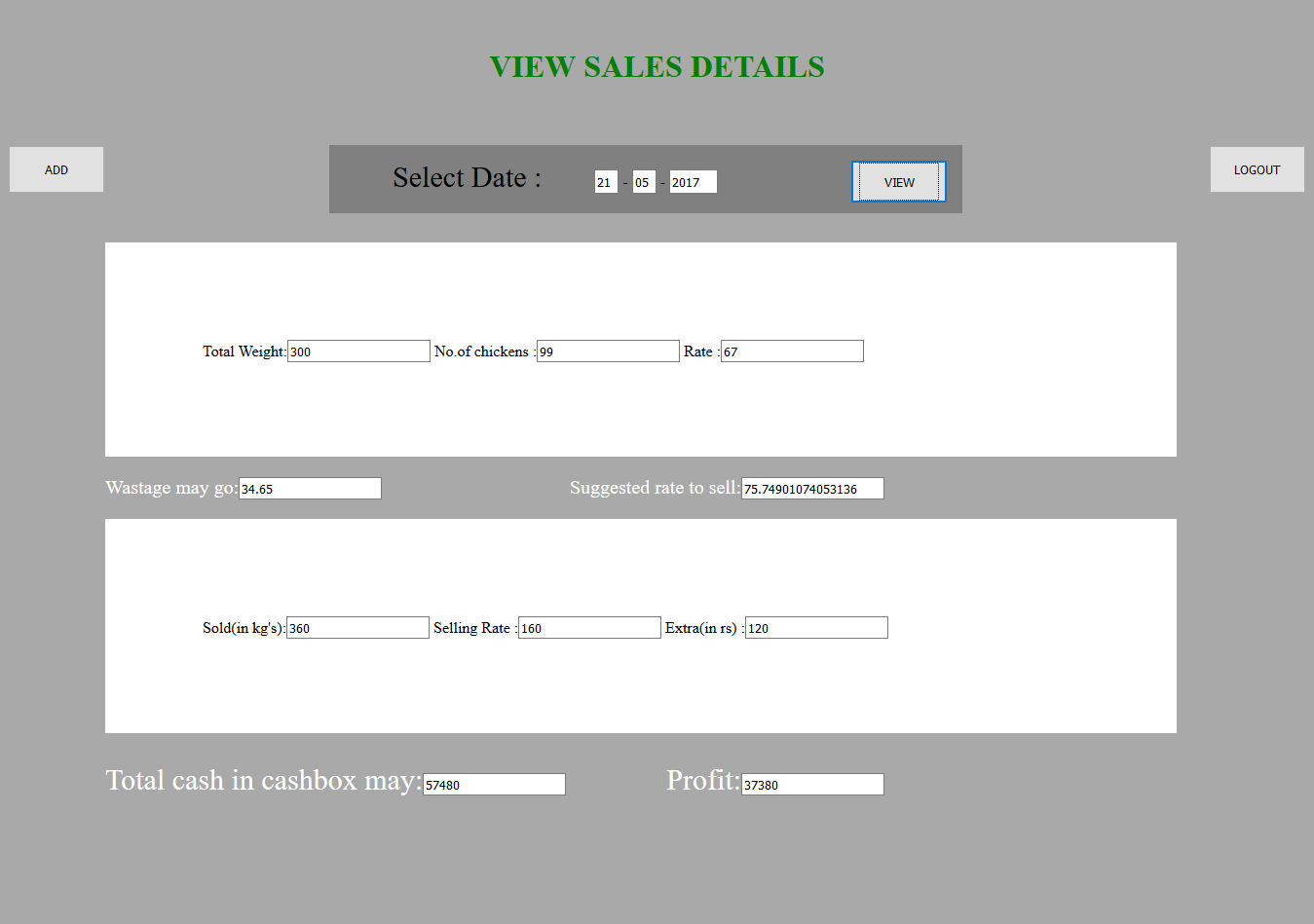
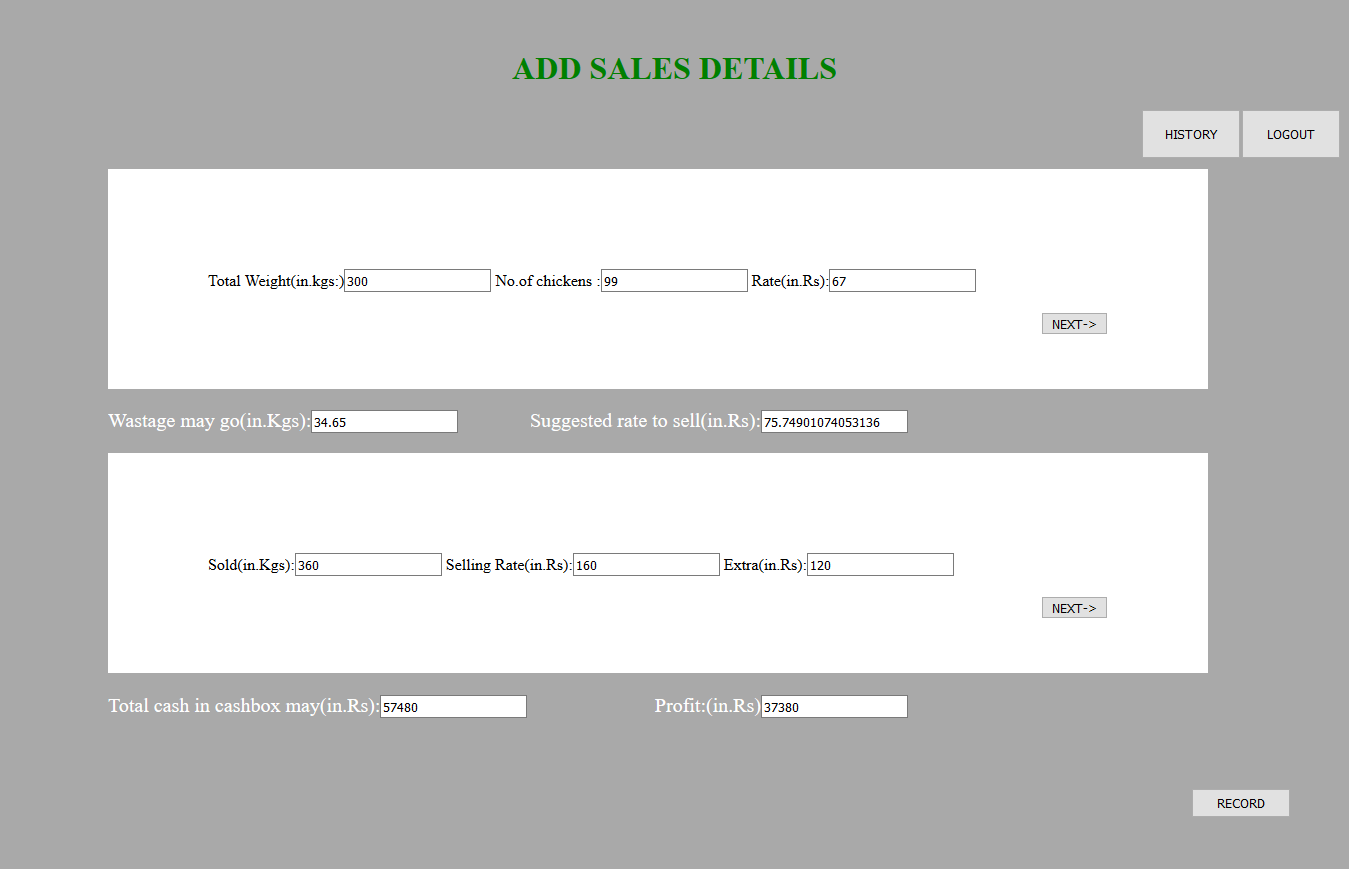
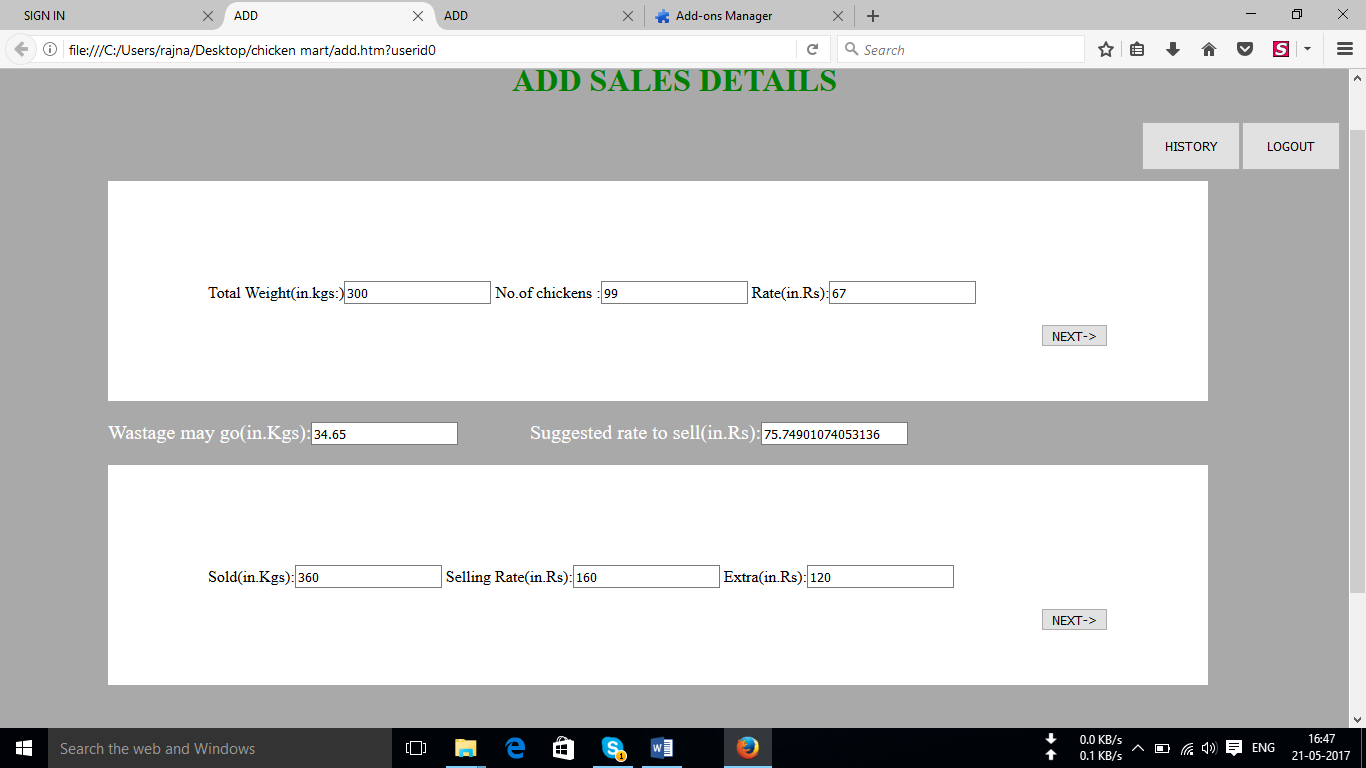
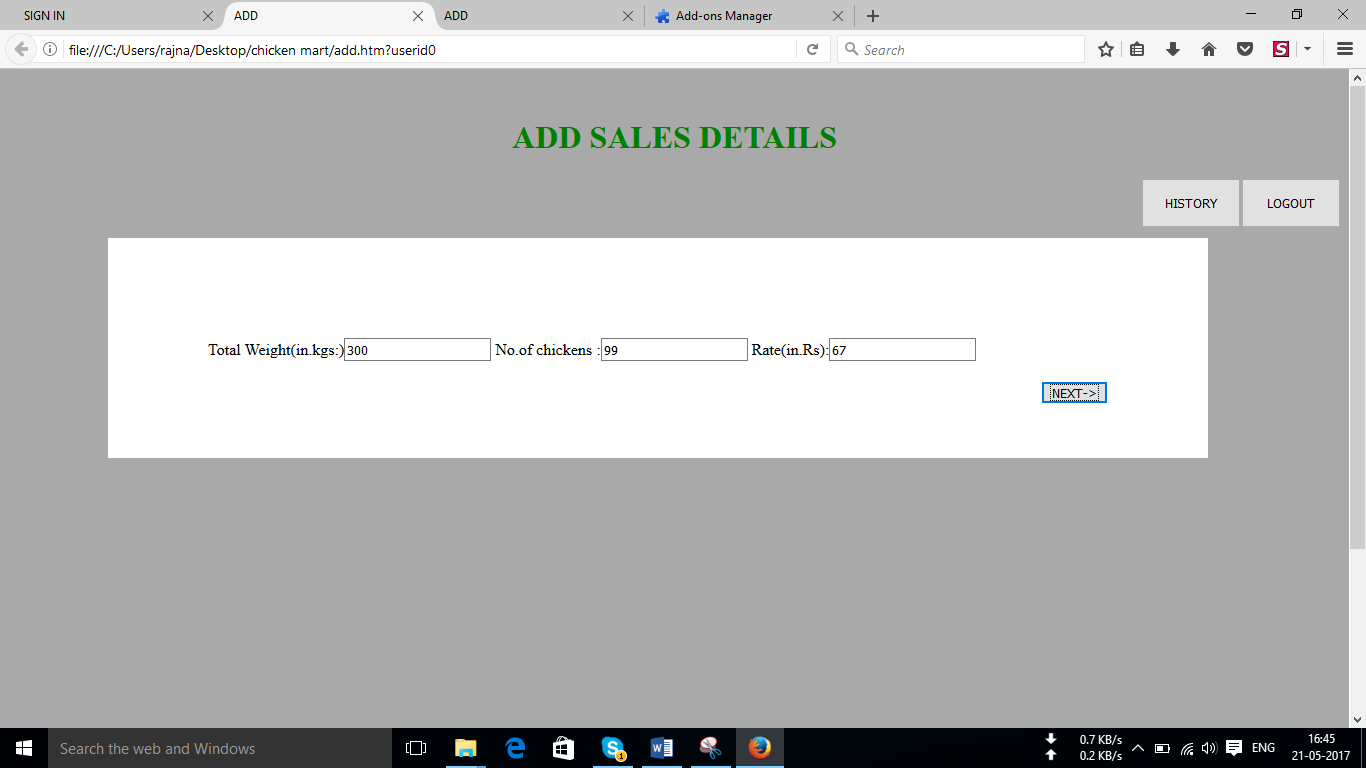
This module is used to perform login. such as adding new Dealers by giving Id and Name. In addition to adding details, it is also provided with the facility to view deatails. Admin can also get the sales details of each dealers day by day basis. In addition to all these functions admin can also send rate details to dealers. The Admin module works on the system.



**4.1.3 Dealer Module**

This module is designed for the use of the Staff. This module helps out the Staff in getting the Student attendance. Staff can get the student attendance on from day to day by selecting the from and to dates. Staff can send the shortage student details to admin for notifying the parents about student attendance. Staff can see the total strength of the students in every semester. In addition to the Attendance, staff can also manage student internal test marks and average of the test marks.





**Program Code**

**Admin Code:**

<!DOCTYPE html>

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

<%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>

<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>

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

<head>

<meta name="keywords" content="" />

<meta name="description" content="" />

<meta http-equiv="content-type" content="text/html; charset=utf-8" />

<title>Student Management</title>

<link

href="http://cdnjs.cloudflare.com/ajax/libs/normalize/3.0.1/normalize.css"

rel="stylesheet" type="text/css" />

<spring:url value="/resources/css/style.css" var="coreCss" />

<spring:url value="/resources/images/" var="images" />

<link href="${coreCss}" rel="stylesheet" type="text/css" media="screen" />

<link rel="stylesheet"

href="http://code.jquery.com/ui/1.11.4/themes/smoothness/jquery-ui.css">

<script

src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>

<script src="http://code.jquery.com/jquery-2.1.1.js"></script>

<script src="http://code.jquery.com/ui/1.11.4/jquery-ui.js"></script>

<script>

function changeFormActionURL(id,path) {

$(id).attr("action",path);

}

function onloadMethod(){

var contextPath = "${pageContext.request.contextPath}/";

<%String projctId = "";

String loginType = "";

if (request.getSession() != null) {

if (request.getSession().getAttribute("projectId") != null) {

projctId = request.getSession().getAttribute("projectId").toString();

}

if (request.getSession().getAttribute("typeLogin") != null) {

loginType = request.getSession().getAttribute("typeLogin").toString();

}

}%>

var SesVar = '<%=projctId%>';

var typeofLogin = '<%=loginType%>';

if (typeofLogin == "Admin") {

document.getElementById("AdminList").style.display = "block";

}

if (typeofLogin == "Staff") {

document.getElementById("StaffList").style.display = "block";

}

$("#AddDprojectID").val(SesVar);

$("#deptupdateProjectId").val(SesVar);

}

function showDiv(id, addDiv, updateDiv) {

document.getElementById(addDiv).style.display = "none";

document.getElementById(updateDiv).style.display = "none";

document.getElementById(id).style.display = "block";

}

function showDiv1(id, addDiv, updateDiv, viewDiv) {

document.getElementById(addDiv).style.display = "none";

document.getElementById(updateDiv).style.display = "none";

document.getElementById(viewDiv).style.display = "none";

document.getElementById(id).style.display = "block";

}

function removeElementVal(id) {

var x = document.getElementById(id);

x.remove(x.selectedIndex);

}

$(document)

.ready(

function() {

$('#addSubject')

.click(

function() {

var length = $("#semisterbu").val();

if (length > 0) {

for (var i = 1; i <= length; i++) {

var idBox = "addTextbox"

+ i;

$('#tableData')

.append(

'<tr><td> Enter the Semester '

+ i

+ ' Subjects with ,(comma) separted </td> <td>'

+ ' <textarea id="'

+ idBox

+ '" name="subjectList['

+ (i - 1)

+ ']" /> </td></tr> ');

document

.getElementById(idBox).value = "Sem\_"

+ i + "-";

}

$('#tableData')

.append(

'<tr> <td colspan="2" align="center"> <input type="submit" class="btn btn-small js-modal-close" value="Add Department" /> </td> </tr>');

} else {

alert("Please enter the no of semisters")

}

});

});

</script>

</head>

<!\_\_-----------------------start of body--------------------\_\_>

<body onload="onloadMethod();">

<div id="menu-wrapper">

<div id="menu">

<ul>

<li><a href="<c:url value="/"/>">Home</a></li>

<li class="current\_page\_item" style="display: none;" id="AdminList"><a

href="<c:url value="/admin"/>">Admin</a></li>

<li style="display: none;" id="StaffList"><a

href="<c:url value="/staff"/>">Staff</a></li>

<li><a href="<c:url value="/logout"/>">Log Out</a></li>

</ul>

</div>

<!-- end #menu -->

</div>

<!\_\_displaying the the admin sub modules with image\_\_>

<div id="header-wrapper">

<div id="header">

<div id="logo">

<h1>

<a href="/"> Student Attendance Management </a>

</h1>

</div>

<div style="line-height: 15px; background-color: #63D7D5; height: 220px; width: 820px; float: center; padding: 55px;">

<table style="border-collapse: separate; border-spacing: 10px; ">

<tr>

<td><a class="js-open-modal"

href="<c:url value="/getDepartment"/>" data-modal-id="popup1"

class="js-open-modal"><img src="${images}departments.png"

width="200" height="200" alt="" /> </a>

</td>

<td>

<a class="js-open-modal"

href="<c:url value="/staffRegister"/>" data-modal-id="popup2"

class="js-open-modal"><img src="${images}staf.png"

width="200" height="200" alt="" /></a></td>

<td><a class="js-open-modal"

href="<c:url value="/getDepartment"/>" data-modal-id="popup3"

class="js-open-modal"><img src="${images}student.gif"

width="200" height="200" alt="" /></a></td>

<td><a class="js-open-modal"

href="<c:url value="/getDepartment"/>" data-modal-id="popup4"

class="js-open-modal"><img src="${images}attendance.jpg"

width="200" height="200" alt="" /></a></td>

</tr>

<tr >

<td align="center"> Add Department </td>

<td align="center"> Add Staff </td>

<td align="center"> Add Student </td>

<td align="center"> View Student Attendence Details </td>

</tr>

</table>

</div>

</div>

</div>

<!\_\_Department module\_\_>

<div id="popup1" class="modal-box">

<header>

<div id="menu">

<ul>

<li><a href=""

onclick="showDiv('AddDepartment','AddDepartment','UpdateDepartment');"

data-modal-id="AddDepartment">Add</a></li>

<li><a href=""

onclick="showDiv('UpdateDepartment','AddDepartment','UpdateDepartment');"

data-modal-id="UpdateDepartment">Update/Delete Dept</a></li>

</ul>

</div>

</header>

<!\_\_Add Department submodule\_\_>

<div class="modal-body"

style="overflow-y: scroll; overflow-x: hidden; height: 400px; width: 1000px">

<div id="AddDepartment" style="display: none;">

<form:form method="POST" action="getDepartment"

commandName="adddepartment" enctype="multipart/form-data">

<table border="1" align="center" id="tableData">

<tr>

<td align="center"><form:input type="hidden" path="projectId" value="${adddepartment.projectId}"

id="AddDprojectID" DISABLED="DISABLED" /></td>

</tr>

<tr>

<td align="center">Department :</td>

<td align="center"><form:input path="departmentName" /></td>

</tr>

<tr>

<td align="center">No of Semesters :</td>

<td align="center"><form:input path="noOfSemsisters"

id="semisterbu" type="text" /></td>

<td colspan="2" align="center"><input id="addSubject"

type="button" value="Add Subjects" /></td>

</tr>

</table>

<table border="0">

<tr>

<td colspan="2" align="center"><h2>${ADDDEPT}</h2></td>

</tr>

</table>

</form:form>

</div>

<!\_\_Update Department submodule\_\_>

<div id="UpdateDepartment" style="display: none;">

<form:form method="POST" action="updateDepartment"

commandName="departmentTB" enctype="multipart/form-data"

id="updateDepartmentForm">

<table border="1" align="center" id="updateDepartmentTable">

<tr>

<td align="center"><form:input path="projectId" type="hidden" value="${departmentTB.projectId}"

id="deptupdateProjectId" DISABLED="DISABLED" /></td>

</tr>

<tr>

<td align="center">Department :</td>

<td><form:select path="departmentName"

onchange="addSubject('departmentBoxUpdate');"

id="departmentBoxUpdate">

<form:option value="select">Select Department</form:option>

<c:forEach var="depart" items="${departmentlist}">

<form:option value="${depart}">${depart}</form:option>

</c:forEach>

</form:select></td>

</tr>

</table>

</form:form>

</div>

</div>

<footer>

<a href="<c:url value="/admin"/>"

class="btn btn-small js-modal-close">Close</a>

</footer>

</div>

<!\_\_Staff module\_\_>

<div id="popup2" class="modal-box">

<header>

<div id="menu">

<ul>

<li><a href=""

onclick="showDiv('AddStaff','AddStaff','UpdateStaff');"

data-modal-id="AddStaff">Add Staff</a></li>

<li><a href=""

onclick="showDiv('UpdateStaff','AddStaff','UpdateStaff');"

data-modal-id="UpdateStaff">Update/Delete Staff</a></li>

</ul>

</div>

</header>

<!\_\_ Add staff submodule\_\_>

<div class="modal-body"

style="overflow-y: scroll; overflow-x: hidden; height: 400px; width: 1000px">

<div id="AddStaff" style="display: none;">

<form:form method="POST" action="staffRegister"

commandName="staffRegister" enctype="multipart/form-data"

id="registerForm" autocomplete="off">

<table border="1" align="center" id="tableDataStaff">

<tr>

<td ROWSPAN="11" align="center"><img id="chngimage"

src="${images}profile.jpg" height="300" width="220" /></td>

</tr>

<tr>

<td align="center">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;First

Name :

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td>

<td align="center"><form:input path="firstName"

id="firstName" /></td>

</tr>

<tr>

<td align="center">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;Last

Name : &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td>

<td align="center"><form:input path="lastName" id="lastName" /></td>

</tr>

<tr>

<td align="center">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;Phone

No : &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td>

<td align="center"><form:input path="phoneNo" id="phoneNo" onchange="verph(this.id);"/></td>

</tr>

<tr>

<td align="center">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;Email

Id : &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td>

<td align="center"><form:input path="emailId" id="emailId" onchange="checkEID(this.id);"/></td>

</tr>

<tr>

<td align="center">Select Profile Photo</td>

<td align="center"><form:input type="file"

path="profilePhoto" id="profilePhoto"

onchange="readURL(this,'#chngimage',0);"

accept="image/gif, image/jpeg, image/png" /></td>

</tr>

<tr>

<td align="center">Department :</td>

<td><form:select path="department" id="departmentBox"

onchange="changeFunc('departmentBox','#semisterBox');">

<form:option value="select">Select Department</form:option>

<c:forEach var="depart" items="${departmentlist}">

<form:option value="${depart}">${depart}</form:option>

</c:forEach>

</form:select></td>

</tr>

<tr>

<td align="center">Semester :</td>

<td><select id="semisterBox"

onchange="changeSubject('departmentBox','semisterBox','#subjectBox');">

<option value="select">Select Semester</option>

</select></td>

</tr>

<tr>

<td align="center">SubjectS :</td>

<td><select id="subjectBox"

onchange="addselectedList('departmentBox','semisterBox','subjectBox','#selectedList');">

<option value="select">Select Subjects</option>

</select></td>

</tr>

<tr>

<td align="center">Selected Subject List :</td>

<td><form:select path="semisterList" id="selectedList">

</form:select> <input type="button" class="btn btn-small" value="Remove"

onclick="removeElementVal('selectedList');" /></td>

</tr>

<tr>

<td colspan="2" align="center"><input type="submit"

class="btn btn-small " value="Add Details" /></td>

</tr>

</table>

</form:form>

</div>

<!\_\_ Update staff sub-module\_\_>

<div id="UpdateStaff" style="display: none;">

<form:form method="POST" action="" commandName="staffRegister"

enctype="multipart/form-data" id="updateStaffForm"

autocomplete="off">

<table border="1" align="center" id="tableDataStaff">

<tr>

<td ROWSPAN="11" align="center"><img

id="chngupdateStaffimage" src="${images}profile.jpg"

height="300" width="220" /></td>

</tr>

<tr>

<td align="center">First Name :</td>

<td align="center"><form:select path="firstName"

id="departmentfirstName" onchange="staffDetails();">

<form:option value="select">Select Name</form:option>

<c:forEach var="staff" items="${staffList}">

<form:option value="${staff}">${staff}</form:option>

</c:forEach>

</form:select></td>

</tr>

<tr>

<td align="center">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;Last

Name : &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td>

<td align="center"><form:input path="lastName"

id="lastUpdateName" /></td>

</tr>

<tr>

<td align="center">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;Phone

No : &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td>

<td align="center"><form:input path="phoneNo"

id="phoneUpdateNo" onchange="verph(this.id);"/></td>

</tr>

<tr>

<td align="center">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;Email

Id : &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td>

<td align="center"><form:input path="emailId"

id="emailUpdateId" onchange="checkEID(this.id);"/></td>

</tr>

<tr>

<td align="center">Select Profile Photo</td>

<td align="center"><form:input type="file"

path="profilePhoto" id="profileUpdatePhoto"

onchange="readURL(this,'#chngupdateStaffimage',0);"

accept="image/gif, image/jpeg, image/png" /></td>

</tr>

<tr>

<td align="center">Department :</td>

<td align="center"><form:select path="department"

id="departmentUpdateBox"

onchange="changeFunc('departmentUpdateBox','#semisterUpdateBox');">

<form:option value="select">Select Department</form:option>

<c:forEach var="depart" items="${departmentlist}">

<form:option value="${depart}">${depart}</form:option>

</c:forEach>

</form:select></td>

</tr>

<tr>

<td align="center">Semester :</td>

<td><select id="semisterUpdateBox"

onchange="changeSubject('departmentUpdateBox','semisterUpdateBox','#subjectUpdateBox');">

<option value="select">Select Semester</option>

</select></td>

</tr>

<tr>

<td align="center">SubjectS :</td>

<td><select id="subjectUpdateBox"

onchange="addselectedList('departmentUpdateBox','semisterUpdateBox','subjectUpdateBox','#selectedUpdateList');">

<option value="select">Select Subjects</option>

</select></td>

</tr>

<tr>

<td align="center">Selected Subject List :</td>

<td><form:select path="semisterList" id="selectedUpdateList">

</form:select> <input type="button" class="btn btn-small"

onclick="removeElementVal('selectedUpdateList');" value="Remove" />

</td>

</tr>

<tr>

<td colspan="2" align="center"><input type="submit"

class="btn btn-small"

onclick="changeFormActionURL('#updateStaffForm','getStaffDetails')"

value="Update" /> <input type="submit" class="btn btn-small "

onclick="changeFormActionURL('#updateStaffForm','deleteStaffDetails')"

value="Delete" /></td>

</tr>

</table>

</form:form>

</div>

</div>

<footer>

<a href="<c:url value="/admin"/>"

class="btn btn-small js-modal-close">Close</a>

</footer>

</div>

<!\_\_ Student module\_\_>

<div id="popup3" class="modal-box">

<header>

<div id="menu">

<ul>

<li><a href=""

onclick="showDiv1('AddStudent','AddStudent','UpdateStudent','ViewStudent');"

data-modal-id="AddDepartment">Add</a></li>

<li><a href=""

onclick="showDiv1('UpdateStudent','AddStudent','UpdateStudent','ViewStudent');"

data-modal-id="UpdateDepartment">Update/Delete Student</a></li>

<li><a href=""

onclick="showDiv1('ViewStudent','ViewStudent','UpdateStudent','ViewStudent');"

data-modal-id="ViewStudent">View All Students</a></li>

</ul>

</div>

</header>

<!\_\_ Add student submodule\_\_>

<div class="modal-body"

style="overflow-y: scroll; overflow-x: hidden; height: 400px; width: 1000px">

<div id="AddStudent" style="display: none;">

<form:form method="POST" action="studentRegister"

commandName="studentTB" enctype="multipart/form-data"

autocomplete="off">

<table border="1" align="center" id="tableDataStudent">

<tr>

<td ROWSPAN="10" align="center"><img id="chngStudentimage"

src="${images}profile.jpg" height="260" width="220" /></td>

</tr>

<tr>

<td align="center">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;First

Name :

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td>

<td align="center"><form:input path="firstName" /></td>

</tr>

<tr>

<td align="center">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;Last

Name : &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td>

<td align="center"><form:input path="lastName" /></td>

</tr>

<tr>

<td align="center">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;Phone

No : &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td>

<td align="center"><form:input path="phoneNo" id="studentaddphone" onchange="verph(this.id);"/></td>

</tr>

<tr>

<td align="center">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;Email

Id :&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td>

<td align="center"><form:input path="emailId" onchange="checkEID(this.id);"/></td>

</tr>

<tr>

<td align="center">Select Profile Photo</td>

<td align="center"><form:input type="file"

path="profilePhoto" id="studentProfile"

onchange="readURL(this,'#chngStudentimage',0);"

accept="image/gif, image/jpeg, image/png" /></td>

</tr>

<tr>

<td align="center">Department :</td>

<td><form:select path="department" id="departmentStudentBox"

onchange="changeFunc('departmentStudentBox','#semisterStudentBox');">

<form:option value="select">Select Department</form:option>

<c:forEach var="depart" items="${departmentlist}">

<form:option value="${depart}">${depart}</form:option>

</c:forEach>

</form:select></td>

</tr>

<tr>

<td align="center">Semester :</td>

<td><select id="semisterStudentBox" name="semister">

<option value="select">Select Semester</option>

</select></td>

</tr>

<tr>

<td colspan="2" align="center"><input type="submit"

class="btn btn-small " value="Add Student" /></td>

</tr>

</table>

</form:form>

</div>

<!\_\_Update Student submodule\_\_>

<div id="UpdateStudent" style="display: none;">

<form:form method="POST" action="studentRegister"

commandName="studentTB" enctype="multipart/form-data"

autocomplete="off" id="tableStudentUpdateForm">

<table border="1" align="center" id="tableDataStudentUpdate">

<tr>

<td ROWSPAN="11" align="center"><img

id="chngStudentUdateimage" src="${images}profile.jpg"

height="260" width="220" /></td>

</tr>

<tr>

<td align="center">Department :</td>

<td><form:select path="department"

id="departmentUpdateStudentBox"

onchange="changeFunc('departmentUpdateStudentBox','#semisterUpdateStudentBox');">

<form:option value="select">Select Department</form:option>

<c:forEach var="depart" items="${departmentlist}">

<form:option value="${depart}">${depart}</form:option>

</c:forEach>

</form:select></td>

</tr>

<tr>

<td align="center">Semester :</td>

<td><select id="semisterUpdateStudentBox" name="semister"

onchange="changeFuncStudent('departmentUpdateStudentBox','semisterUpdateStudentBox','#studentUpdateStudentBox');">

<option value="select">Select Semester</option>

</select></td>

</tr>

<tr>

<td align="center">Student Id and Name :</td>

<td><select id="studentUpdateStudentBox"

onchange="getIndividualStudentDetails();">

<option value="select">Student Id</option>

</select></td>

</tr>

<tr>

<td align="center">Student ID:</td>

<td><form:input path="studentId"

id="studentId" />

</td>

</tr>

<tr>

<td align="center">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;First

Name :

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td>

<td align="center"><form:input path="firstName"

id="ustudentFName" /></td>

</tr>

<tr>

<td align="center">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;Last

Name : &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td>

<td align="center"><form:input path="lastName"

id="ustudentLName" /></td>

</tr>

<tr>

<td align="center">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;Phone

No : &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td>

<td align="center"><form:input path="phoneNo"

id="ustudentPNO" onchange="verph(this.id);"/></td>

</tr>

<tr>

<td align="center">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;Email

Id : &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td>

<td align="center"><form:input path="emailId"

id="ustudentEmail" onchange="checkEID(this.id);"/></td>

</tr>

<tr>

<td align="center">Select Profile Photo</td>

<td align="center"><form:input type="file"

path="profilePhoto" id="studentUpdateProfile"

onchange="readURL(this,'#chngStudentUdateimage',0);"

accept="image/gif, image/jpeg, image/png" /></td>

</tr>

<tr>

<td colspan="2" align="center"><input type="submit"

class="btn btn-small"

onclick="changeFormActionURL('#tableStudentUpdateForm','updatestudentRegister')"

value="Update" /> <input type="submit" class="btn btn-small "

onclick="changeFormActionURL('#tableStudentUpdateForm','deletestudentRegister')"

value="Delete" /></td>

</tr>

</table>

</form:form>

</div>

<!\_\_ Student Strength submodel\_\_>

<div id="ViewStudent" style="display: none;">

<table border="1" align="center" id="tableDataStudentResult">

<tr>

<td align="center">Department :</td>

<td><select id="departmentResultSheet"

onchange="changeFunc('departmentResultSheet','#semisterResultSheet');">

<option value="select">Select Department</option>

<c:forEach var="depart" items="${departmentlist}">

<option value="${depart}">${depart}</option>

</c:forEach>

</select></td>

</tr>

<tr>

<td align="center">Semester :</td>

<td><select id="semisterResultSheet" name="semister">

<option value="select">Select Semester</option>

</select></td>

</tr>

<tr>

<td colspan="2" align="center"><input type="button"

onclick="getAllStudents('departmentResultSheet','semisterResultSheet');"

class="btn btn-small " value="Get ALL Students" /></td>

</tr>

</table>

<br /> <br />

<table border="1" align="center" id="allStudents">

<tr>

<td><b>StudentId </b></td>

<td><b>FirstName </b></td>

<td><b>LastName </b></td>

<td><b>EmailId </b></td>

<td><b>Department </b></td>

<td><b>Semister </b></td>

</tr>

</table>

</div>

</div>

<footer>

<a href="<c:url value="/admin"/>" class="btn btn-small ">Close</a>

</footer>

</div>

<!\_\_ View Attendance Model\_\_>

<div id="popup4" class="modal-box">

<header>

<div id="menu">

<ul>

<li><a href=""

onclick="showDiv('ResultSheet','ResultSheet','UpdateStudentResult');"

data-modal-id="ResultSheet">View Student Attendance</a></li>

</ul>

</div>

</header>

<div class="modal-body"

style="overflow-y: scroll; overflow-x: hidden; height: 400px; width: 1000px">

<div id="ResultSheet" style="display: none;">

<table border="1" align="center" id="tableDataStudentResult">

<tr>

<td align="center">Department :</td>

<td><select id="departmentResultSheet1"

onchange="changeFunc('departmentResultSheet1','#semisterResultSheet1');">

<option value="select">Select Department</option>

<c:forEach var="depart" items="${departmentlist}">

<option value="${depart}">${depart}</option>

</c:forEach>

</select></td>

</tr>

<tr>

<td align="center">Semester :</td>

<td><select id="semisterResultSheet1" name="semister">

<option value="select">Select Semester</option>

</select></td>

</tr>

<tr>

<td colspan="2" align="center"><input type="button"

onclick="sendEmails('departmentResultSheet1','semisterResultSheet1');"

class="btn btn-small " value="Send Shortage Mail to Parents" /></td>

</tr>

<tr>

<td colspan="2" align="center"><input type="button"

onclick="excelsheet('departmentResultSheet1','semisterResultSheet1');"

class="btn btn-small " value="Prepare Excel Sheet" /></td>

</tr>

<tr>

<label> <a id="mylink" target="\_blank"> Download the ExcelSheet </a> </label>

</tr>

</table>

<br /> <br />

<table border="1" align="center" id="shortageResult">

<tr>

<td><b>StudentId </b></td>

<td><b>StudentName </b></td>

<td><b>Attendance Status </b></td>

<td><b>Email Id </b></td>

</tr>

</table>

</div>

</div>

<footer>

<a href="<c:url value="/admin"/>"

class="btn btn-small js-modal-close">Close</a>

</footer>

</div>

<div style="clear: both;">&nbsp;</div>

<div style="clear: both;">&nbsp;</div>

<div style="clear: both;">&nbsp;</div>

<div style="clear: both;">&nbsp;</div>

<div style="clear: both;">&nbsp;</div>

<div style="clear: both;">&nbsp;</div>

<div style="clear: both;">&nbsp;</div>

<div style="clear: both;">&nbsp;</div>

<div style="clear: both;">&nbsp;</div>

<div style="clear: both;">&nbsp;</div>

<div style="clear: both;">&nbsp;</div>

<div style="clear: both;">&nbsp;</div>

<div style="clear: both;">&nbsp;</div>

<div style="clear: both;">&nbsp;</div>

<div id="footer">

<p>

<a

href="www.sjpcstech.com" rel="nofollow"></a>.

</p>

</div>

<!-- end #footer -->

</body>

<!\_\_ javascript functions\_\_>

<script type="text/javascript">

function addselectedList(dpBox, semBox, subBox, selcList) {

var selectBox = document.getElementById(dpBox);

var selectedValue = selectBox.options[selectBox.selectedIndex].value;

var semBox = document.getElementById(semBox);

var semValue = semBox.options[semBox.selectedIndex].value;

var subjectBox = document.getElementById(subBox);

var subjectBoxValue = subjectBox.options[subjectBox.selectedIndex].value;

var dataVal = selectedValue + '-' + semValue + '-' + subjectBoxValue;

var newOption = $('<option value="'+dataVal+'">' + dataVal

+ '</option>');

$(selcList).append(newOption);

}

function addSubject(depBox) {

var selectBox = document.getElementById(depBox);

var selectedValue = selectBox.options[selectBox.selectedIndex].value;

var datastring = "dept=" + selectedValue;

$

.ajax({

type : "GET",

contentType : 'application/json; charset=utf-8',

url : '/SAMS/getDepartementSub/',

dataType : "JSON",

cache : false,

data : {

dept : selectedValue

},

success : function(response) {

if (response != null) {

var tableId = document

.getElementById("updateDepartmentTable");

var x = document

.getElementById("updateDepartmentTable").rows.length;

if (x > 2) {

for (var k = 2; k < x; k++) {

document.getElementById(

"updateDepartmentTable").deleteRow(

2);

}

}

for (var i = 0; i < response.length; i++) {

$('#updateDepartmentTable')

.append(

'<tr><td> Enter the Semester '

+ ' Subjects with ,(comma) separted </td> <td>'

+ ' <textarea id="updateTextbox'+i+'" name="subjectList['+i+']" />'

+ '</td></tr>');

var idUpdateBox = "updateTextbox" + i;

document.getElementById(idUpdateBox).value = response[i];

}

if (response.length > 0) {

$('#updateDepartmentTable')

.append(

"<tr> <td colspan='2' align='center'> <input type='submit' class='btn btn-small ' onclick=\"changeFormActionURL('#updateDepartmentForm','updateDepartment')\" value='Update Department' /> <input type='submit' class='btn btn-small ' onclick=\"changeFormActionURL('#updateDepartmentForm','deleteDepartment')\" value='Delete Department' /> </td> </tr>");

} else {

$('#updateDepartmentTable')

.append(

'<tr> <td colspan="2" align="center">Invalid Project Id</td></tr>');

}

} else {

alert("Response is empty.");

}

},

error : function(xhr) {

alert("ERROR: " + xhr)

}

});

}

function staffDetails() {

var selectBox = document.getElementById("departmentfirstName");

var selectedValue = selectBox.options[selectBox.selectedIndex].value;

var datastring = "staffName=" + selectedValue

$

.ajax({

type : "GET",

contentType : 'application/json; charset=utf-8',

url : '/SAMS/getStaffDetails/',

dataType : "JSON",

cache : false,

data : datastring,

success : function(response) {

if (response != null) {

lastUpdateName, phoneUpdateNo, emailUpdateId,

chngupdateStaffimage, selectedUpdateList

$("#lastUpdateName").val(response.lastName);

$("#phoneUpdateNo").val(response.phoneNo);

$("#emailUpdateId").val(response.emailId);

$("#chngupdateStaffimage").attr('src',

response.profileFilePath);

var tableID;

$("#selectedUpdateList").empty();

for (var i = 0; i < response.semisterList.length; i++) {

var newOption = $('<option value="'+response.semisterList[i]+'">'

+ response.semisterList[i]

+ '</option>');

$('#selectedUpdateList').append(newOption);

}

}

},

error : function(xhr) {

alert("ERROR: " + xhr)

}

});

}

function excelsheet(dep, sem) {

var selectBox = document.getElementById(dep);

var selectedValue = selectBox.options[selectBox.selectedIndex].value;

var semBox = document.getElementById(sem);

var semValue = semBox.options[semBox.selectedIndex].value;

$

.ajax({

type : "GET",

contentType : 'application/json; charset=utf-8',

url : '/SAMS/prepareSheet/',

dataType : "text",

cache : false,

data : {

dept : selectedValue,

sem : semValue

},

success : function(response) {

if (response != null) {

alert(response);

document.getElementById("mylink").href = response;

}

},

error : function(xhr) {

alert("ERROR: " + xhr)

}

});

}

function getAllStudents(dep, sem) {

var selectBox = document.getElementById(dep);

var selectedValue = selectBox.options[selectBox.selectedIndex].value;

var semBox = document.getElementById(sem);

var semValue = semBox.options[semBox.selectedIndex].value;

alert(selectedValue+"-"+semValue);

$

.ajax({

type : "GET",

contentType : 'application/json; charset=utf-8',

url : '/SAMS/allStudents/',

dataType : "JSON",

cache : false,

data : {

dept : selectedValue,

sem : semValue

},

success : function(response) {

if (response != null) {

var tableId = document

.getElementById("allStudents");

var x = document.getElementById("allStudents").rows.length;

if (x > 1) {

for (var k = 1; k < x; k++) {

document.getElementById("allStudents")

.deleteRow(1);

}

}

for (var i = 0; i < response.length; i++) {

$('#allStudents').append(

'<tr><td>' + response[i].studentId

+ '</td><td>'

+ response[i].firstName

+ '</td><td>'

+ response[i].lastName

+ '</td><td>'

+ response[i].emailId

+ '</td><td>'

+ response[i].department

+ '</td><td>'

+ response[i].semister

+ '</td></tr>'

);

}

}

},

error : function(xhr) {

alert("ERROR: " + xhr)

}

});

}

function sendEmails(dep, sem) {

var selectBox = document.getElementById(dep);

var selectedValue = selectBox.options[selectBox.selectedIndex].value;

var semBox = document.getElementById(sem);

var semValue = semBox.options[semBox.selectedIndex].value;

$

.ajax({

type : "GET",

contentType : 'application/json; charset=utf-8',

url : '/SAMS/sendEmail/',

dataType : "JSON",

cache : false,

data : {

dept : selectedValue,

sem : semValue

},

success : function(response) {

if (response != null) {

var tableId = document

.getElementById("shortageResult");

var x = document.getElementById("shortageResult").rows.length;

if (x > 1) {

for (var k = 1; k < x; k++) {

document.getElementById("shortageResult")

.deleteRow(1);

}

}

for (var i = 0; i < response.length; i++) {

$('#shortageResult').append(

'<tr><td>' + response[i].studentId

+ '</td><td>'

+ response[i].studentName

+ '</td><td>'

+ response[i].comments

+ '</td><td>'

+ response[i].emailId

+ '</td></tr>'

);

}

}

},

error : function(xhr) {

alert("ERROR: " + xhr)

}

});

}

function getIndividualStudentDetails() {

var selectBox = document.getElementById("departmentUpdateStudentBox");

var selectedValue = selectBox.options[selectBox.selectedIndex].value;

var semBox = document.getElementById("semisterUpdateStudentBox");

var semValue = semBox.options[semBox.selectedIndex].value;

var studentId = document.getElementById("studentUpdateStudentBox");

var studentValue = studentId.options[studentId.selectedIndex].value;

var datastring = "dept=" + selectedValue + '&sem=' + semValue

+ '&studentId=' + studentValue;

$

.ajax({

type : "GET",

contentType : 'application/json; charset=utf-8',

url : '/SAMS/getStudentIndividual/',

dataType : "JSON",

cache : false,

data : datastring,

success : function(response) {

if (response != null) {

lastUpdateName, phoneUpdateNo, emailUpdateId,

chngupdateStaffimage, selectedUpdateList

$("#studentId").val(response.studentId);

$("#ustudentFName").val(response.firstName);

$("#ustudentLName").val(response.lastName);

$("#ustudentPNO").val(response.phoneNo);

$("#ustudentEmail").val(response.emailId);

$("#chngStudentUdateimage").attr('src',

response.profileFilePath);

var tableID;

$("#selectedUpdateList").empty();

for (var i = 0; i < response.semisterList.length; i++) {

var newOption = $('<option value="'+response.semisterList[i]+'">'

+ response.semisterList[i]

+ '</option>');

$('#selectedUpdateList').append(newOption);

}

}

},

error : function(xhr) {

alert("ERROR: " + xhr)

}

});

}

function changeFuncStudent(StuD, stuSem, idStudet) {

var selectBox = document.getElementById(StuD);

var selectedValue = selectBox.options[selectBox.selectedIndex].value;

var semBox = document.getElementById(stuSem);

var semValue = semBox.options[semBox.selectedIndex].value;

var datastring = "dept=" + selectedValue + '&sem=' + semValue;

$

.ajax({

type : "GET",

contentType : 'application/json; charset=utf-8',

url : '/SAMS/getStudentIDs/',

dataType : "JSON",

cache : false,

data : datastring,

success : function(response) {

if (response != null) {

$(idStudet).empty();

$(idStudet)

.append(

'<option value="select">Select Student Id</option>');

for (var i = 0; i < response.length; i++) {

var newOption = $('<option value="'+response[i]+'">'

+ response[i] + '</option>');

$(idStudet).append(newOption);

}

} else {

alert("Response is empty.");

}

},

error : function(xhr) {

alert("ERROR: " + xhr)

}

});

}

function changeSubject(depBox, semBox, subBox) {

var selectBox = document.getElementById(depBox);

var selectedValue = selectBox.options[selectBox.selectedIndex].value;

var semBox = document.getElementById(semBox);

var semValue = semBox.options[semBox.selectedIndex].value;

var datastring = "dept=" + selectedValue + '&sem=' + semValue;

$.ajax({

type : "GET",

contentType : 'application/json; charset=utf-8',

url : '/SAMS/subjects/',

dataType : "JSON",

cache : false,

data : datastring,

success : function(response) {

if (response != null) {

$(subBox).empty();

$(subBox).append(

'<option value="select">Select Subject</option>');

for (var i = 0; i < response.length; i++) {

var newOption = $('<option value="'+response[i]+'">'

+ response[i] + '</option>');

$(subBox).append(newOption);

}

} else {

alert("Response is empty.");

}

},

error : function(xhr) {

alert("ERROR: " + xhr)

}

});

}

function changeFunc(dept, semsiter) {

var selectBox = document.getElementById(dept);

var selectedValue = selectBox.options[selectBox.selectedIndex].value;

var datastring = "dept=" + selectedValue;

alert(datastring);

$.ajax({

type : "GET",

contentType : 'application/json; charset=utf-8',

url : '/SAMS/semister/',

dataType : "JSON",

cache : false,

data : datastring,

success : function(response) {

if (response != null) {

$(semsiter).empty();

$(semsiter).append(

'<option value="select">Select Semester</option>');

for (var i = 0; i < response.length; i++) {

var newOption = $('<option value="'+response[i]+'">'

+ response[i] + '</option>');

$(semsiter).append(newOption);

}

} else {

alert("Response is empty.");

}

},

error : function(xhr) {

alert("ERROR: " + xhr)

}

});

}

function readURL(input, id, index) {

if (input.files && input.files[index]) {

var reader = new FileReader();

reader.onload = function(e) {

$(id).attr('src', e.target.result).width(220).height(260);

};

reader.readAsDataURL(input.files[index]);

}

}

function resizeImg() {

var thisImg = $('.containerProfile');

var refH = thisImg.height();

var refW = thisImg.width();

var refRatio = refW / refH;

var imgH = thisImg.children("img").height();

var imgW = thisImg.children("img").width();

if ((imgW / imgH) > refRatio) {

thisImg.addClass("portrait");

thisImg.removeClass("landscape");

} else {

thisImg.addClass("landscape");

thisImg.removeClass("portrait");

}

}

$(document).ready(resizeImg())

$(window).resize(function() {

resizeImg();

});

function populateResultTable(response) {

}

$(function() {

var appendthis = ("<div class='modal-overlay js-modal-close menu-wrapper menu'></div>");

$('a[data-modal-id]').click(function(e) {

e.preventDefault();

$("body").append(appendthis);

$(".modal-overlay").fadeTo(500, 0.7);

//$(".js-modalbox").fadeIn(500);

var modalBox = $(this).attr('data-modal-id');

$('#' + modalBox).fadeIn($(this).data());

});

$(".js-modal-close, .modal-overlay").click(function() {

$(".modal-box, .modal-overlay").fadeOut(500, function() {

$(".modal-overlay").remove();

});

});

$(window).resize(function() {

$(".modal-box").css({

top : ($(window).height() - $(".modal-box").outerHeight()) / 2,

left : ($(window).width() - $(".modal-box").outerWidth()) / 2

});

});

$(window).resize();

});

function verph(id) {

if(document.getElementById(id).value != ""){

var y = document.getElementById(id).value;

if(isNaN(y)||y.indexOf(" ")!=-1)

{

alert("Invalid Mobile No.");

document.getElementById(id).value="";

document.getElementById(id).focus();

return false;

}

if (y.length>10 || y.length<10)

{

alert("Mobile No. should be 10 digit");

document.getElementById(id).value="";

document.getElementById(id).focus();

return false;

}

if (!(y.charAt(0)=="9" || y.charAt(0)=="8" || y.charAt(0)=="7"))

{

alert("Mobile No. should start with 9 ,8 or 7 ");

document.getElementById(id).value="";

document.getElementById(id).focus();

return false;

}

}

}

function checkEID(id)

{

var mail=document.getElementById(id).value;

var mail\_pattern=/^\w+([\.-]?\w+)\*@\w+([\.-]?\w+)\*(\.\w{2,3})+$/;

if(mail\_pattern.test(mail)==false)

{

alert("Enter Proper Mail ID");

document.getElementById('mesg\_mail').innerHTML='Enter a Valid E-Mail Id';

}

}

</script>

</html>

**Staff Page Code:**

<!DOCTYPE html>

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

<%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>

<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>

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

<head>

<meta name="keywords" content="" />

<meta name="description" content="" />

<meta http-equiv="content-type" content="text/html; charset=utf-8" />

<title>Student Management</title>

<spring:url value="/resources/css/style.css" var="coreCss" />

<spring:url value="/resources/images/" var="images" />

<link href="http://fonts.googleapis.com/css?family=Arvo"

rel="stylesheet" type="text/css" />

<link href="http://fonts.googleapis.com/css?family=Coda:400,800"

rel="stylesheet" type="text/css" />

<link href="${coreCss}" rel="stylesheet" type="text/css" media="screen" />

<link rel="stylesheet"

href="http://code.jquery.com/ui/1.11.4/themes/smoothness/jquery-ui.css">

<script

src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js"></script>

<script src="http://code.jquery.com/jquery-1.10.2.js"></script>

<script src="http://code.jquery.com/ui/1.11.4/jquery-ui.js"></script>

<!\_\_javascript functions\_\_>

<script type="text/javascript">

$(function() {

$("#datepicker").datepicker();

});

$(function() {

$("#datepicker1").datepicker();

});

function updateInternalSheet(length){

var internalSheet = {

sheet: []

};

for (var i = 0; i < length; i++) {

var internalID = document.getElementById("internal-"+i+"-1").value;

var firstName = document.getElementById("internal-"+i+"-2").value;

var department = document.getElementById("internal-"+i+"-3").value;

var semister = document.getElementById("internal-"+i+"-4").value;

var subject = document.getElementById("internal-"+i+"-5").value;

var internals1 = document.getElementById("internal-"+i+"-6").value;

var internals2 = document.getElementById("internal-"+i+"-7").value;

var internals3 = document.getElementById("internal-"+i+"-8").value;

internalSheet.sheet.push({

"internalId" : internalID,

"firstName" : firstName,

"department" : department,

"semister" : semister,

"subject" : subject,

"internals1" : internals1,

"internals2" : internals2,

"internals3" : internals3

});

}

var jsonText = JSON.stringify(internalSheet);

var urlVal = '/SAMS/storeInternals';

$.ajax({

url : urlVal,

type: 'POST',

contentType: 'application/json; charset=utf-8',

cache : false,

data: jsonText,

success : function(response) {

alert(response);

},

error : function(xhr) {

alert("ERROR: " + xhr)

}

});

}

function deleteInternalSheet(){

var department = document.getElementById("internal-0-3").value;

var semister = document.getElementById("internal-0-4").value;

var subject = document.getElementById("internal-0-5").value;

alert(department+"-"+semister+"-"+subject);

var dataVal="sheet=" + department+"-"+semister+"-"+subject;

$.ajax({

url : '/SAMS/deleteSheet/',

type: 'GET',

contentType: 'application/json; charset=utf-8',

cache : false,

data: dataVal,

dataType : "text",

success : function(response) {

alert(response);

},

error : function(xhr) {

alert("ERROR: " + xhr)

}

});

}

function intrnalMarks(subject){

var selectBox = document.getElementById(subject);

var selectedValue = selectBox.options[selectBox.selectedIndex].value;

document.getElementById("internalsDiv").style.display = "block";

var urlVal='/SAMS/addInternalMarks/';

$.ajax({

type : "GET",

contentType : 'application/json; charset=utf-8',

url : urlVal,

dataType : "JSON",

cache : false,

data : {

value : selectedValue

},

success : function(response) {

if (response != null)

{

var tableId = document.getElementById("internltable");

var x = document.getElementById("internltable").rows.length;

if (x > 1) {

for (var k = 1; k < x; k++) {

document.getElementById("internltable").deleteRow(2);

}

}

for (var i = 0; i < response.length; i++) {

if(response[i]!=null){

var formData='';

formData+='<tr>'

+'<td> <input id="internal-'+i+'-1" name="response['+i+'].internalId}" value="'+response[i].internalId+'" readonly="readonly"/> </td>'

+'<td> <input id="internal-'+i+'-2" name="response['+i+'].firstName" value="'+response[i].firstName+'" readonly="readonly"/> </td>'

+'<td> <input id="internal-'+i+'-6" name="response['+i+'].internals1" value="'+response[i].internals1+'" /> </td>'

+'<td> <input id="internal-'+i+'-7" name="response['+i+'].internals2" value="'+response[i].internals2+'" /> </td>'

+'<td> <input id="internal-'+i+'-8" name="response['+i+'].internals3" value="'+response[i].internals3+'" /> </td>'

+'<td> <input id="internal-'+i+'-9" name="response['+i+'].percentage" value="'+response[i].percentage+' %" /> </td>'

+'<td> <input id="internal-'+i+'-3" name="response['+i+'].department" value="'+response[i].department+'" readonly="readonly"/> </td>'

+'<td> <input id="internal-'+i+'-4" name="response['+i+'].semister" value="'+response[i].semister+'" readonly="readonly"/> </td>'

+'<td> <input id="internal-'+i+'-5" name="response['+i+'].subject" value="'+response[i].subject+'" readonly="readonly"/> </td>'

+'</tr>';

$('#internltable').append(formData);

}

}

if(response.length>0){

$('#internltable').append('<tr><td> </td><td> </td><td> </td><td> </td><td> </td><td> </td><td><input type="button" onclick="updateInternalSheet('+response.length+');" value="Update" /> <td><input type="button" onclick="deleteInternalSheet();" value="Delete Data" /> </td></tr>');

}

/\* var formData='<form:form action="storeInternals" id="loginFormId" method="post" commandName="internalSheet" >'

+'<c:if test="${not empty internalSheet}">'

+'<c:forEach var="sheet" items="${internalSheet}">'

+'<tr>'

+'<td> <form:input path="${sheet.internalId}" value="${sheet.internalId}" readonly="readonly"/> </td>'

+'</tr>'

+'</c:forEach>'

+'</c:if>'

+'</form:form>'; \*/

}

},

error : function(xhr) {

alert("ERROR: " + xhr)

}

});

}

function changesubjects(subject,from,to,tableId,idVal){

var selectBox = document.getElementById(subject);

var selectedValue = selectBox.options[selectBox.selectedIndex].value;

var date1 = document.getElementById(from).value;

var date2 = document.getElementById(to).value;

var datastring = "value=" + selectedValue + '&from=' + date1+ '&to=' + date2;

var urlVal="";

if(idVal==0){

urlVal='/SAMS/getAttendanceResult/'

}else{

urlVal='/SAMS/storeResultSheetDB/'

}

$.ajax({

type : "GET",

contentType : 'application/json; charset=utf-8',

url : urlVal,

dataType : "JSON",

cache : false,

data : {

value : selectedValue,

from : date1,

to : date2

},

success : function(response) {

if (response != null)

{

var tableId = document.getElementById("tabledata");

var x = document.getElementById("tabledata").rows.length;

if (x > 1) {

for (var k = 1; k < x; k++) {

document.getElementById("tabledata").deleteRow(1);

}

}

for (var i = 0; i < response.length; i++) {

$('#tabledata')

.append(

'<tr> <td>'+response[i].studentId +'</td>'+

' <td>'+response[i].studentName +'</td>'+

' <td>'+response[i].subject +'</td>'+

' <td>'+response[i].totalNoofDays +'</td>'+

' <td>'+response[i].noofDaysAttended +'</td>'+

' <td>'+response[i].percentage +'% </td> </tr>'

);

}

document.getElementById("attendenceDiv").style.display = "block";

}

},

error : function(xhr) {

alert("ERROR: " + xhr)

}

});

}

function onloadMethod(){

<%String projctId = "";

String loginType = "";

if (request.getSession() != null) {

if (request.getSession().getAttribute("projectId") != null) {

projctId = request.getSession().getAttribute("projectId").toString();

}

if (request.getSession().getAttribute("typeLogin") != null) {

loginType = request.getSession().getAttribute("typeLogin").toString();

}

}%>

var SesVar = '<%=projctId%>';

var typeofLogin = '<%=loginType%>';

if (typeofLogin == "Admin") {

document.getElementById("AdminList").style.display = "block";

}

if (typeofLogin == "Staff") {

document.getElementById("StaffList").style.display = "block";

}

}

</script>

</head>

<!\_\_ start of body\_\_>

<body onload="onloadMethod();">

<div id="menu-wrapper">

<div id="menu">

<ul>

<li><a href="<c:url value="/"/>">Home</a></li>

<li style="display: none;" id="AdminList"><a

href="<c:url value="/admin"/>">Admin</a></li>

<li class="current\_page\_item" style="display: none;" id="StaffList"><a

href="<c:url value="/staff"/>">Staff</a></li>

<li><a href="<c:url value="/logout"/>">Log Out</a></li>

</ul>

</div>

<!-- end #menu -->

</div>

<div id="header-wrapper">

<div id="header">

<div id="logo">

<h1>

<a href="/"> Student Attendance Management System </a>

</h1>

</div>

</div>

</div>

<div

style="background-color: #63D7D5; height: 450px; width: 1300px; margin-left: 10px; overflow-y: scroll; overflow-x: scroll;">

<div>

<table>

<tr>

<td>

<div

style="background-color: #D6FAF5; height: 450px; width: 250px; margin-left: 0px;">

<form>

<table border="0" align="center">

<tr>

<td colspan="2" align="center">Student Details</td>

</tr>

<%-- <tr>

<td align="left">Department</td>

<td align="right"><select id="department">

<option value="select">Select Department</option>

<c:forEach var="staff" items="${StaffDepartment}">

<option value="${staff.department}">${staff.department}</option>

</c:forEach>

</select></td>

</tr>

<tr>

<td align="left">Semester</td>

<td align="left"><select id="semister">

<option value="select">Select Semester</option>

<c:forEach var="staff" items="${StaffDepartment}">

<option value="${staff.semister}">${staff.semister}</option>

</c:forEach>

</select></td>

</tr> --%>

<tr>

<td align="left">Subject</td>

<td align="left"><select id="subject">

<option value="select">Select Subject</option>

<c:forEach var="staff" items="${StaffDepartment}">

<c:forEach var="subjects" items="${staff.subjectList}">

<!\_\_ to select subject and dates\_\_>

<option

value="${staff.department}-${staff.semister}-${subjects}">${staff.department}-${staff.semister}-${subjects}</option>

</c:forEach>

</c:forEach>

</select></td>

</tr>

<tr>

<td align="left">From</td>

<td align="right"><input name="insuranceDueDate"

id="datepicker" /></td>

</tr>

<tr>

<td align="left">To</td>

<td align="right"><input name="insuranceDueDate"

id="datepicker1" /></td>

</tr>

<tr>

<td colspan="2" align="center"><input type="button"

onclick="changesubjects('subject','datepicker','datepicker1','#tabledata',0);"

value="Get Details to View" /></td>

</tr>

<tr>

<td colspan="2" align="center"><input type="button"

onclick="changesubjects('subject','datepicker','datepicker1','#tabledata',1);"

value="Get Details & Store in DB" /></td>

</tr>

<tr>

<td colspan="2" align="center"><input type="button"

onclick="intrnalMarks('subject');"

value="Add Internals Marks" /></td>

</tr>

</table>

</form>

</div>

</td>

<td valign="top">

<div id="attendenceDiv" style="display: none;">

<table border="1" id="tabledata">

<tr>

<th

style="max-width: 120px; min-width: 100px; max-height: 30px; min-height: 20px; width: 150px; height: 0px;">Student

Id</th>

<th

style="max-width: 120px; min-width: 100px; max-height: 30px; min-height: 20px; width: 150px; height: 0px;">Student

Name</th>

<th

style="max-width: 120px; min-width: 100px; max-height: 30px; min-height: 20px; width: 150px; height: 0px;">Subject</th>

<th

style="max-width: 120px; min-width: 100px; max-height: 30px; min-height: 20px; width: 150px; height: 0px;">Total

No of Classes</th>

<th

style="max-width: 120px; min-width: 100px; max-height: 30px; min-height: 20px; width: 150px; height: 0px;">No

of Classes Attended</th>

<th

style="max-width: 120px; min-width: 100px; max-height: 30px; min-height: 20px; width: 150px; height: 0px;">Percentage</th>

</tr>

</table>

</div>

<!\_\_Internal marks Sheet\_\_>

<div id="internalsDiv" style="display: none;">

<table border="1" id="internltable">

<tr>

<th

style="max-width: 120px; min-width: 100px; max-height: 30px; min-height: 20px; width: 150px; height: 0px;">Internal Id

Id</th>

<th

style="max-width: 120px; min-width: 100px; max-height: 30px; min-height: 20px; width: 150px; height: 0px;">Student

Name</th>

<th

style="max-width: 120px; min-width: 100px; max-height: 30px; min-height: 20px; width: 150px; height: 0px;">Internals 1</th>

<th

style="max-width: 120px; min-width: 100px; max-height: 30px; min-height: 20px; width: 150px; height: 0px;">Internals 2</th>

<th

style="max-width: 120px; min-width: 100px; max-height: 30px; min-height: 20px; width: 150px; height: 0px;">Internals 3</th>

<th

style="max-width: 120px; min-width: 100px; max-height: 30px; min-height: 20px; width: 150px; height: 0px;">Average</th>

<th

style="max-width: 120px; min-width: 100px; max-height: 30px; min-height: 20px; width: 150px; height: 0px;">Department

</th>

<th

style="max-width: 120px; min-width: 100px; max-height: 30px; min-height: 20px; width: 150px; height: 0px;">Semester

</th>

<th

style="max-width: 120px; min-width: 100px; max-height: 30px; min-height: 20px; width: 150px; height: 0px;">Subject

</th>

</tr>

</table>

</div>

</td>

</tr>

</table>

</div>

</div>

<div style="clear: both;">&nbsp;</div>

<div id="footer">

<p>

<a

href="www.sjpcstech.com" rel="nofollow"></a>.

</p>

</div>

<!-- end #footer -->

</body>

</html>

**Staff on Android**

**LoginActivity.java**

package com.sjpcs.sams.activities;

import java.io.IOException;

import org.apache.http.HttpResponse;

import org.apache.http.HttpStatus;

import org.apache.http.client.ClientProtocolException;

import org.apache.http.client.HttpClient;

import org.apache.http.client.methods.HttpGet;

import org.apache.http.impl.client.DefaultHttpClient;

import android.app.ActionBar;

import android.app.Activity;

import android.app.AlertDialog;

import android.app.ProgressDialog;

import android.content.Context;

import android.content.DialogInterface;

import android.content.Intent;

import android.graphics.Color;

import android.graphics.LightingColorFilter;

import android.graphics.Typeface;

import android.graphics.drawable.ColorDrawable;

import android.os.AsyncTask;

import android.os.Bundle;

import android.text.Editable;

import android.text.TextWatcher;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.EditText;

import android.widget.TextView;

import com.sjpcs.sams.R;

import com.sjpcs.sams.model.SamsBO;

import com.sjpcs.sams.model.StaffTB;

import com.sjpcs.sams.utilities.AlertDialogClass;

import com.sjpcs.sams.utilities.JsonParser;

public class LoginActivity extends Activity implements OnClickListener {

Button btn\_LogIn;

EditText et\_LogIn,et\_Password,et\_CollegeId;

JsonParser parser = new JsonParser();

Context context;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_login);

//to change background of activity

getWindow().getDecorView().setBackgroundDrawable(new ColorDrawable(Color.parseColor("#ff31321e")));

//to change the title bar color

ActionBar bar=getActionBar();

bar.setBackgroundDrawable(new ColorDrawable(Color.parseColor("#ff6d6f36")));

//to change button background

Button bt= (Button)findViewById(R.id.btn\_login);

bt.getBackground().setColorFilter(new LightingColorFilter(0xFF31321E,0x00000000));

//to set font for text

TextView tv=(TextView)findViewById(R.id.logtext);

Typeface face=Typeface.createFromAsset(getAssets(), "adv.ttf");

tv.setTypeface(face);

set\_Id\_View\_Components();

bindListeners();

// to hide and unhide the login button

if(!(et\_LogIn.getText().toString().isEmpty() ||

et\_Password.getText().toString().isEmpty() ||

et\_CollegeId.getText().toString().isEmpty())){

btn\_LogIn.setEnabled(true);

} else{

btn\_LogIn.setEnabled(false);

}

}

private void bindListeners() {

btn\_LogIn.setOnClickListener(this);

et\_LogIn.addTextChangedListener(watcher);

et\_Password.addTextChangedListener(watcher);

et\_CollegeId.addTextChangedListener(watcher);

}

//to get the username and passwords to verify

private void set\_Id\_View\_Components() {

context = this;

btn\_LogIn = (Button) findViewById(R.id.btn\_login);

et\_LogIn = (EditText) findViewById(R.id.et\_login\_id);

et\_Password = (EditText) findViewById(R.id.et\_password);

et\_CollegeId = (EditText) findViewById(R.id.et\_college\_id);

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

// Handle action bar item clicks here. The action bar will

// automatically handle clicks on the Home/Up button, so long

// as you specify a parent activity in AndroidManifest.xml.

int id = item.getItemId();

if (id == R.id.action\_settings) {

return true;

}

return super.onOptionsItemSelected(item);

}

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

new GetLoginAuth().execute();

/\*Intent intent\_Sem\_Subject = new Intent(context,SelectSemSubActivity.class);

startActivity(intent\_Sem\_Subject);\*/

}

TextWatcher watcher = new TextWatcher() {

@Override

public void onTextChanged(CharSequence s, int start, int before, int count) {

if(!(et\_LogIn.getText().toString().isEmpty() ||

et\_Password.getText().toString().isEmpty() ||

et\_CollegeId.getText().toString().isEmpty())){

btn\_LogIn.setEnabled(true);

} else{

btn\_LogIn.setEnabled(false);

}

}

@Override

public void beforeTextChanged(CharSequence s, int start, int count, int after) {

// TODO Auto-generated method stub

}

@Override

public void afterTextChanged(Editable s) {

// TODO Auto-generated method stub

}

};

//Asynctask to communicate with the server db

public class GetLoginAuth extends AsyncTask<Void, Void, Boolean>{

HttpClient hc;

HttpGet get;

HttpResponse resp;

boolean result;

ProgressDialog progressDialog;

@Override

protected void onPreExecute(){

progressDialog = ProgressDialog.show(context, context.getString(R.string.please\_wait), context.getString(R.string.logging\_in\_),true);

}

@Override

protected Boolean doInBackground(Void... params) {

try {

hc = new DefaultHttpClient();

get = new HttpGet("http://"+SamsBO.getInstance().getIpAddress()+":"+SamsBO.getInstance().getPortNum()+"/SAMS/login/"+et\_LogIn.getText().toString()+"/"+et\_Password.getText().toString()+"/"+et\_CollegeId.getText().toString());

try {

resp = hc.execute(get);

} catch (ClientProtocolException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

if(resp!=null){

if(resp.getStatusLine().getStatusCode()==HttpStatus.SC\_OK){

/\*try {

parser.parseJsonLoginResponce(parser.ParseEntity(resp.getEntity()));

} catch (Exception e) {

e.printStackTrace();

}\*/

try {

SamsBO.getInstance().setStaffTB(parser.getJOSNResponse(resp, new StaffTB()));

result = true;

} catch (Exception e) {

e.printStackTrace();

}

} else{

result = false;}}

} catch (Exception e) {

e.printStackTrace();

result = false;

}

return result;

}

@Override

protected void onPostExecute(Boolean result){

if(progressDialog!=null && progressDialog.isShowing()){

progressDialog.dismiss();

}

if(result){

Intent intent\_Sem\_Subject = new Intent(context,SelectSemSubActivity.class);

startActivity(intent\_Sem\_Subject);

} else{

AlertDialogClass alertDialogObj = new AlertDialogClass(context);

alertDialogObj.showDialog(R.string.login\_failed\_please\_try\_again\_later\_);

}

}

}}

**StudentAttendanceActivity.java**

package com.sjpcs.sams.activities;

import java.io.IOException;

import java.io.UnsupportedEncodingException;

import java.util.ArrayList;

import java.util.List;

import org.apache.http.HttpResponse;

import org.apache.http.HttpStatus;

import org.apache.http.client.ClientProtocolException;

import org.apache.http.client.HttpClient;

import org.apache.http.client.methods.HttpPost;

import org.apache.http.entity.StringEntity;

import org.apache.http.impl.client.DefaultHttpClient;

import org.codehaus.jackson.JsonGenerationException;

import org.codehaus.jackson.map.JsonMappingException;

import org.codehaus.jackson.map.ObjectMapper;

import android.app.ActionBar;

import android.app.Activity;

import android.app.ProgressDialog;

import android.content.Context;

import android.graphics.Color;

import android.graphics.drawable.ColorDrawable;

import android.os.AsyncTask;

import android.os.Bundle;

import android.util.SparseBooleanArray;

import android.view.ActionMode;

import android.view.Menu;

import android.view.MenuItem;

import android.widget.AbsListView.MultiChoiceModeListener;

import android.widget.ListView;

import com.sjpcs.sams.R;

import com.sjpcs.sams.model.SamsBO;

import com.sjpcs.sams.model.Student;

import com.sjpcs.sams.utilities.AlertDialogClass;

public class StudentAttendanceActivity extends Activity {

ListView list;

ListViewAdapter listAdapter;

String[] studentsList;

List<String> studentArray = new ArrayList<String>();

Context context;

Activity activity;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_student\_attendance);

//to change background of activity

getWindow().getDecorView().setBackgroundDrawable(new ColorDrawable(Color.parseColor("#ff31321e")));

//to change the title bar color

ActionBar bar=getActionBar();

bar.setBackgroundDrawable(new ColorDrawable(Color.parseColor("#ff6d6f36")));

context = this;

activity = this;

//to show the student list under that semester

for(Student student: SamsBO.getInstance().getAttendence().getStudents()){

studentArray.add(student.getStudentName());

}

list = (ListView) findViewById(R.id.listview);

listAdapter = new ListViewAdapter(this, android.R.layout.simple\_list\_item\_1,studentArray.toArray(new String[studentArray.size()]));

list.setAdapter(listAdapter);

list.setChoiceMode(ListView.CHOICE\_MODE\_MULTIPLE\_MODAL);

list.setMultiChoiceModeListener(new MultiChoiceModeListener() {

@Override

public boolean onPrepareActionMode(ActionMode arg0, Menu arg1) {

return false;

}

@Override

public void onDestroyActionMode(ActionMode arg0) {

listAdapter.removeSelection();

}

@Override

public boolean onCreateActionMode(ActionMode mode, Menu menu) {

mode.getMenuInflater().inflate(R.menu.student\_attendance, menu);

return true;

}

@Override

public boolean onActionItemClicked(ActionMode mode, MenuItem item) {

switch(item.getItemId()){

case R.id.action\_submit:

SparseBooleanArray selected = listAdapter.getSelectedIds();

System.out.println("hello sparse : "+ selected);

new AttendanceResult().execute();

return true;

default:

return false;

}

}

@Override

public void onItemCheckedStateChanged(ActionMode mode, int position, long id, boolean checked) {

final int checkedCount = list.getCheckedItemCount();

for(int i=0;i<SamsBO.getInstance().getAttendence().getStudents().size();i++){

if(SamsBO.getInstance().getAttendence().getStudents().get(i).getStudentName()

.equals(list.getItemAtPosition(position).toString())){

if(checked){

SamsBO.getInstance().getAttendence().getStudents().get(i).setStatus(true);

} else{

SamsBO.getInstance().getAttendence().getStudents().get(i).setStatus(false);

}

}

}

list.getItemAtPosition(position);

list.setSelector(android.R.color.holo\_blue\_bright);

mode.setTitle(checkedCount + " Students selected");

listAdapter.toggleSelection(position);

}

});

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.student\_attendance, menu);

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

// Handle action bar item clicks here. The action bar will

// automatically handle clicks on the Home/Up button, so long

// as you specify a parent activity in AndroidManifest.xml.

int id = item.getItemId();

if (id == R.id.action\_submit) {

new AttendanceResult().execute();

}

return super.onOptionsItemSelected(item);

}

//to get the student list from the server db

class AttendanceResult extends AsyncTask<Void, Void, Boolean>{

HttpClient hc;

HttpPost post;

HttpResponse resp;

String jsonString;

ProgressDialog progressDialog;

boolean result;

@Override

protected void onPreExecute(){

progressDialog = ProgressDialog.show(context, context.getString(R.string.please\_wait), context.getString(R.string.submitting\_data),true);

}

@Override

protected Boolean doInBackground(Void...params){

try {

hc = new DefaultHttpClient();

ObjectMapper mapper = new ObjectMapper();

try {

jsonString = mapper.writeValueAsString(SamsBO.getInstance().getAttendence());

} catch (JsonGenerationException e) {

e.printStackTrace();

} catch (JsonMappingException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

post = new HttpPost("http://"+SamsBO.getInstance().getIpAddress()+":"+SamsBO.getInstance().getPortNum()+"/SAMS/attendenceResult");

try {

post.setEntity(new StringEntity(jsonString,"UTF8"));

} catch (UnsupportedEncodingException e) {

e.printStackTrace();

}

post.setHeader("Content-type", "application/json");

try {

resp = hc.execute(post);

} catch (ClientProtocolException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

if(resp!=null && resp.getStatusLine().getStatusCode()==HttpStatus.SC\_OK){

result = true;

} else{

result = false;

}

} catch (Exception e) {

e.printStackTrace();

result = false;

}

return result;

}

@Override

protected void onPostExecute(Boolean result){

if(progressDialog!=null && progressDialog.isShowing()){

progressDialog.dismiss();

}

if(!result){

AlertDialogClass alertDialogObj = new AlertDialogClass(context);

alertDialogObj.showDialog(R.string.failed\_to\_submit);

}

activity.finish(); }}}

**Chapter-6**

**Testing Process**

**6.1 Introduction**

Testing is the process used to help identify the correctness, completeness, security, and quality of developed computer software. A technical investigation, performed on behalf of stakeholders, that is intended to reveal quality-related information about the product with respect to the context in which it is intended to operate.

Process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**6.1.1 White-box and black-box testing**

White box and black box testing are terms used to describe the point of view a test engineer takes when designing test cases. Black box being an external view of the test object and white box being an internal view. Software testing is partly intuitive, but largely systematic. Good testing involves much more than just running the program a few times to see whether it works. Thorough analysis of the program under test, backed by a broad knowledge of testing techniques and tools are prerequisites to systematic testing. Software Testing is the process of executing software in a controlled manner; in order to answer the question “Does this software behave as specified?” Software testing is used in association with Verification and Validation. Verification is the checking of or testing of items, including software, for conformance and consistency with an associated specification. Software testing is just one kind of verification, which also uses techniques as reviews, inspections, walk-through. Validation is the process of checking what has been specified is what the user wanted actually.

* Validation: Are we doing the right job?
* Verification: Are we doing the job right?

**6.1.2 Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**6.1.3 Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

**6.1.4 System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**6.1.5 Performance Test**

The Performance test ensures that the output be produced within the time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

**6.1.6 Integration Testing**

The phase of software testing in which individual software modules are combined and tested as a group. The purpose of integration testing is to verify functional, performance and reliability requirements placed on major design items. These "design items", i.e. assemblages (or groups of units), are exercised through their interfaces using black box testing, success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface. Test cases are constructed to test that all components within assemblages interact correctly, for example across procedure calls or process activations, and this is done after testing individual modules, i.e. unit testing.

The overall idea is a "building block" approach, in which verified assemblages are added to a verified base which is then used to support the integration testing of further assemblages, in this approach, all or most of the developed modules are coupled together to form a complete software system or major part of the system and then used for integration testing.

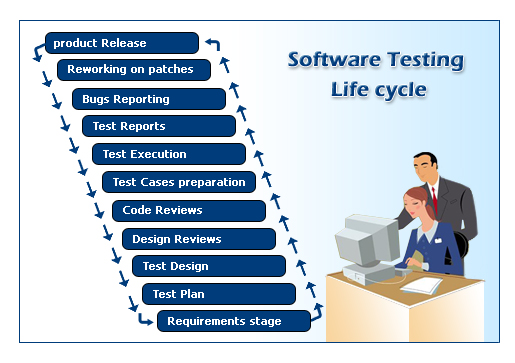
**6.2 A Software testing cycle:**

Fig 6.1

Software Testing Life Cycle

Although testing varies between organizations, there is a cycle to testing:

1. Requirements Analysis: Testing should begin in the requirements phase of the software development life cycle.
2. During the design phase, testers work with developers in determining what aspects of a design are testable and under what parameter those tests work.
3. Test Planning: Test Strategy, Test Plan(s), Test Bed creation.
4. Test Development: Test Procedures, Test Scenarios, Test Cases, and Test Scripts to use in testing software.
5. Test Execution: Testers execute the software based on the plans and tests and report any errors found to the development team.
6. Test Reporting: Once testing is completed, testers generate metrics and make final reports on their test effort and whether or not the software tested is ready for release.
7. Retesting the Defects

Not all errors or defects reported must be fixed by a software development team. Some may be caused by errors in configuring the test software to match the development or production environment. Some defects can be handled by a workaround in the production environment. Others might be deferred to future releases of the software, or the deficiency might be accepted by the business user. There are yet other defects that may be rejected by the development team (of course, with due reason) if they deem it inappropriate to be called a defect.

**6.3 Testing fundamentals**

Software testing is an important element of S/W quality assurance and represents the ultimate review of specification, design and coding. The increasing visibility of S/W as a system element and the costs associated with a S/W failure are motivating forces for well planned, through testing.

Though the test phase is often thought of as separate and distinct from the development effort--first develop, and then test--testing is a concurrent process that provides valuable information for the development team.

There are at least three options for integrating Project

Builder into the test phase:

* Testers do not install Project Builder, use Project Builder functionality to compile and source-control the modules to be tested and hand them off to the testers, whose process remains unchanged.
* the testers import the same project or projects that the developers use.
* Create a project based on the development project but customized for the testers (for example, it does not include support documents, specs, or source), who import it.

A combination of the second and third options works best. Associating the application with a project can be useful during the testing phase, as well. We can create actions to automatically run test scripts or add script types and make them dependent on the modules to test.

**6.4 Testing objectives**

There are several rules that can serve as testing objectives.

They are

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

If testing is conducted successfully according to the objectives stated above, it will uncover errors in the software. Also, testing demonstrates that software functions appear to the working according to specification, that performance requirements appear to have been met.

**6.5 OBJECT ORIENTED TESTING**

**6.5.1 UNIT TESTING**

Unit testing focuses the verification effort on the smallest unit of S/W design i.e., the module. The unit testing is always white-box oriented and the step can be conducted in parallel for modules.

During unit test, testers can use the same project or projects as the developers, if functional units organize the project, or separate projects have been created for functional units. The project or projects can also be exported, so unit test can take place in a variety of environments and on a variety of platforms.

**6.5.2 Unit test considerations**

The tests that occur as part of unit testing. The module ‘interface’ is tested to ensure that information properly flows into and out of the program unit under test. The ‘local data structures’ are examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithms execution.

‘Boundary Conditions’ are tested to ensure that the module operates properly at boundaries established to limit or restrict processing. All ‘independent paths’ through the control structures are exercised to ensure that all statements in a module have been executed at least once. Finally, all ‘error-handling paths’ are tested.

**6.5.3 Unit test procedures**

Unit testing is considered an equivalent to the coding step. After the source level code has been developed, reviewed and verified for correct syntax, unit test case design begins since a module is not a standalone program, ‘driver’ and/or ‘stub’ S/W must be developed for each unit test. In most applications, a driver is nothing more than a main program that accepts test case data, passes such data to the module to be tested, and prints the relevant results. The stubs serve to replace modules that are subordinates called by the modules to be tested. A stub or a dummy stub or a dummy subprogram uses the subordinate modules interface, may do minimal data manipulation, prints verification of entry, and returns. The drivers and scrubs represent overhead i.e., both are S/W that must be written but that is not delivered with the final S/W product. If the drivers and the stub are kept simple, then the overhead is low.

The Unit Test is carried out in this project, and is found successful. The data is flowing correctly to all part of the project.

**6.5.4 INTEGRATION TESTING**

Integration testing is a systematic technique for constructing the program structure while at the same time conducting test to uncover errors associated with interfacing. The objective is to take unit-tested modules and build a program structure that has been dictated by design.

**6.5.5 Top-down integration**

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module.

**Chapter-6**

**Security implementations of the developed software**

* All the transactions are done within the localhost, hence no third parties can steal or tamper any information.
* As admin registers the staff there can be no fake id’s.
* Each staff as unique username and password, hence there is no visibility of each other account information.
* There is only one Admin who adds all the staff and students of all the departments, there can be no false id’s of staff.
* Authentication is required for the staff to mark attendance.
* Only the staff registered by the Admin can mark attendance.

**6.7 Test Cases**

|  |  |
| --- | --- |
| Test case No. | UTC – 1 |
| Test case Description | Admin Login |
| Input given | Enter correct Admin credentials, click signin |
| Expected output | Admin session should be open |
| Actual output | Admin tab created |
| Remarks | Success |

|  |  |
| --- | --- |
| Test case No. | UTC – 2 |
| Test case Description | Test for Department details adding |
| Input given | Enter department details, click add details button |
| Expected output | Department details should appear in Database |
| Actual output | Found department details in DB |
| Remarks | Success |

|  |  |
| --- | --- |
| Test case No. | UTC – 3 |
| Test case Description | Test for Staff details adding |
| Input given | Enter the Staff details, select the subject they handle, click add button |
| Expected output | Staff details should be there in DB |
| Actual output | Added details found in Staff table in DB |
| Remarks | Success |

|  |  |
| --- | --- |
| Test case No. | UTC – 4 |
| Test case Description | Test for Student details adding |
| Input given | Enter the Student details, select the department and semester, click add button |
| Expected output | Student details should be there in DB |
| Actual output | Added details found under Student table in DB |
| Remarks | Success |

|  |  |
| --- | --- |
| Test case No. | UTC – |
| Test case Description | Test for Staff login |
| Input given | Enter the username and password of Staff |
| Expected output | Staff session should be created |
| Actual output | Staff tab created |
| Remarks | Success |

|  |  |
| --- | --- |
| Test case No. | UTC – |
| Test case Description | Test to view student attendance |
| Input given | Update the attendance for 3 times, select period for one day |
| Expected output | Total no of classes should be shown as 3 |
| Actual output | Total no of classes shown as 3 |
| Remarks | Success |

|  |  |
| --- | --- |
| Test case No. | UTC – |
| Test case Description | Test for calculation of avg of Internal Marks |
| Input given | Enter the two internals for students, click update button |
| Expected output | Average of IA marks should be update |
| Actual output | Average shown |
| Remarks | Success |

|  |  |
| --- | --- |
| Test case No. | UTC – 6 |
| Test case Description | Test for login of staff in mobile |
| Input given | Enter username and password in android mobile, click login |
| Expected output | Selecting semester and subject activity should be lauch |
| Actual output | Semesters and subject activity shown |
| Remarks | Success |

**fuck**

|  |  |
| --- | --- |
| Test case No. | TC – 7 |
| Test case Description | Test for getting |
| Input given | Click on the add subjects |
| Expected output | Added subjects details of that semester will be shown |
| Actual output | Subjects details will be shown |
| Remarks | Success |

|  |  |
| --- | --- |
| Test case No. | TC – 8 |
| Test case Description | Test to check whether the add staff icon is working or not |
| Input given | Click on add staff button to add the details of the staff |
| Expected output | Enter the staff details and the subject list they handle and press add details |
| Actual output | The given details will be displayed |
| Remarks | Success |

|  |  |
| --- | --- |
| Test case No. | TC – 9 |
| Test case Description | Test to check whether the update staff is working or not |
| Input given | Click on the particular icon, if any updation in the staff details, and click on update button |
| Expected output | The given information must be updated |
| Actual output | Update is successful |
| Remarks | Success |

|  |  |
| --- | --- |
| Test case No. | TC – 10 |
| Test case Description | Test to check whether the delete staff is working or not |
| Input given | Click on the particular icon, if any deletion in the staff details, and click on update button |
| Expected output | The given information must be deleted |
| Actual output | Delete is successful |
| Remarks | Success |

|  |  |
| --- | --- |
| Test case No. | TC – 11 |
| Test case Description | Test to check whether the add student icon is working or not |
| Input given | Add student to the particular department and semester(if new) and press the add student button |
| Expected output | The new student details must be displayed |
| Actual output | Student details is shown |
| Remarks | Success |

|  |  |
| --- | --- |
| Test case No. | TC – 12 |
| Test case Description | Test to check whether the updated student details is working or not |
| Input given | To update the student ID and name of the student |
| Expected output | The new student ID and name of the student must be shown |
| Actual output | The update is successful |
| Remarks | Success |

|  |  |
| --- | --- |
| Test case No. | TC – 13 |
| Test case Description | Test to check whether the delete student details is working or not |
| Input given | Select the branch and semester and go view student details |
| Expected output | In view student details the particular student details should not be there in the semester |
| Actual output | Student not found in the semester |
| Remarks | Success |

|  |  |
| --- | --- |
| Test case No. | TC – 14 |
| Test case Description | To view shortage details of student |
| Input given | Select the branch and semester, then click on the get details |
| Expected output | Shortage list of students with less than 75% of attendance must be shown |
| Actual output | Students with all below 75% attendance is shown |
| Remarks | Success |

|  |  |
| --- | --- |
| Test case No. | TC – 15 |
| Test case Description | Test to check whether the logout option is working or not |
| Input given | Click the logout button |
| Expected output | Admin session should be terminated |
| Actual output | Admin tag is disappeared |
| Remarks | Success |

**Chapter-7**

**Limitations of the Project**

There are very slight limitations to the current implementation of the project and the project can be further improved in the future to overcome these limitations. Some of the limitations of this project are:

* The server must be on and running on localhost system in order the run the both web and mobile app.
* The passwords of Staff and Admin once registered cannot be changed. It can be enhanced in future.
* The mobile app runs only on android device, and there is no supports for platforms

like windows and IOS.

* There is no separate module for students to view his/her IA details. The only way to view their attendance is when the staff provide them the print of the attendance.
* Staff can only login but cannot logout. The session expires as soon the staff comes out of the application.

**Chapter 8**

**Future Enhancement of the project**

Even though this project proves to be effective there are some enhancements that may be done in future. They are:

* IA marks editing module in android application.
* The web app can be made more user friendly i.e., in the current app the staff cannot change his/her Account credentials once registered.
* Time Table also can be added and can editable.
* The shortage updates of students are sent to the e-mail of parent, in future it can be made to send to their mobile number itself.
* Separate module for student to view IA details.
* Enhancements like proper login and logout for staff in android module can be made.

**Chapter-9**

**Bibliography**

**9.1 Reference:**

* http://developer.android.com/guide/index.html
* http://www.java2s.com/
* http://buckytutorials.com
* http://www.eclipse.org/
* http://www.stackoverflow.com/
* http://github.org/
* http://bestcss.in/
* http://www.oracle.com
* http://www.w3schools.com/