Student Minor Research Project

STUDENTS WEBSITE IN JSP

Under RUSA 2.0 Scheme

(Through Ch.S.D.St.Theresa's College for Women (Autonomous), Eluru, AP)

Submitted by

Ms K Sravani, III B.Sc. MECs (Reg.No.11705002)

Ms Y Divya, III B.Sc. MECs (Reg.No.11705007)

Under the guidance of

Ms K R Sravanthi

Assistant Professor & Project Advisor

DEPARTMENT OF COMPUTER SCIENCE

SRI Y N COLLEGE

(AUTONOMOUS)

Thrice Accredited by NAAC at 'A' Grade

Recognized by UGC as "College with Potential for Excellence"

Narsapur-534275, AP, India

December-2019

DEPARTMENT OF COMPUTER SCIENCE

SRI Y N COLLEGE

(AUTONOMOUS)

Thrice Accredited by NAAC at 'A' Grade

Recognized by UGC as "College with Potential for Excellence"

Narsapur-534275, AP, India

CERTIFICATE

*This is to certify that the project work entitled '***Students Website in JSP***' is bonafide work carried out by Ms K Sravani (Reg.No: 11705002), Ms Y Divya (Reg.No: 11705007), submitted in Third Year of the degree B.Sc. in Computer Science during the year 2019-20 is an authentic work under my supervision and guidance.*

To the best of my knowledge, the matter embodied in the project work has not been submitted to any other College/Institute.

Date: 29-12-2019

Ms K R Sravanthi
Project Advisor
Department of Computer Science

PREFACE

This project **“Students website in JSP”** provides us a simple interface for maintenance of student information. It can be used by educational institutes or colleges to maintain the records of old students easily. Achieving this objective is difficult using a manual system as the information is scattered, can be redundant and collecting relevant information may be very time consuming. All these problems are solved using this project.

Throughout the project the focus has been on presenting information in an easy and intelligible manner. The project is very useful for those who want to know about Student Information Management Systems and want to develop softwares/websites based on the same concept.

The project provides facilities like online registration and profile creation of students thus reducing paperwork and automating the record generation process in an educational institution

ACKNOWLEDGEMENT

We place on record and warmly acknowledge the continuous encouragement, invaluable supervision, timely suggestions and inspired guidance offered by our Project advisor, ***Ms K R Sravanthi,*** *Assistant Professor, Department of Computer Science,* ***Sri Y N College (Autonomous), Narsapur*** *in bringing this report to a successful completion.*

We are grateful to ***Ms Ch Lakshmi****, Assistant Professor, Department of Computer Science for permitting us to make use of the facilities available in the department to carry out the project successfully. Last but not the least we express our sincere thanks to all of our friends who have patiently extended all sorts of help for accomplishing this undertaking.*

Finally we extend our gratefulness to one and all who are directly or indirectly involved in the successful completion of this project work.

Ms K Sravani
III.B.Sc.MECs
Reg. No 11705002

Ms Y Divya
III.B.Sc.MECs
Reg. No.11705007

DECLARATION

We, the undersigned, declare that the project entitled "**Students Website in JSP**", being submitted in Third Year of Bachelor of Science in Computer Science, Sri Y N College (Autonomous), is the work carried out by us.

Ms K Sravani	**Ms Y Divya**
III.B.Sc.MECs	III.B.Sc.MECs
Reg. No 11705002	Reg. No.11705007

TABLE OF CONTENTS

Contents **Page No.**

1. SYNOPSIS

Abstract:

Students Website in JSP can be used by education institutes to maintain the records of students easily. Achieving this objective is difficult using a manual system as the information is scattered, can be redundant and collecting relevant information may be very time consuming. All these problems are solved using this project. It allows old students of a college (or any institute) to share their information with other students. It enables a student to provide his/her information to others and also get information about other students.

Name of the Project: Students Website in JSP

Existing System:

In the educational institutes or colleges to maintain the records of old students is difficult using a manual system as the information is scattered, can be redundant and collecting relevant information may be very time consuming. All these problems are solved using this project.

Proposed System:

This project **"Students website in JSP"** provides us a simple interface for maintenance of student information. It can be used by educational institutes or colleges to maintain the records of old students easily. Throughout the project the focus has been on presenting information in an easy and intelligible manner. The project is very useful for those who want to know about Student Information Management Systems and want to develop softwares/websites based on the same concept.

The project provides facilities like online registration and profile creation of students thus reducing paperwork and automating the record generation process in an educational institution

Objectives:

- Online registration of students
- Maintenance of student records
- Searching student records

Users Views:

- Administrator
- Student

Platform:

Operating System: Microsoft Windows

Technologies Used:

- **Front End**: HTML and JavaScript
- **Web designing language**: JSP
- **Back end**: Oracle8i/Oracle9i

Software Requirements:

- Tomcat 5.x
- Oracle8i/Oracle9i
- JBoss
- SMTP Server
- Java Server Pages (JSP)
- Java Beans
- Enterprise Java Beans
- Java Mail

Hardware Requirements:

- Intel Pentium IV processor or equivalent or higher
- 2 GB RAM or Higher
- 250 GB HDD or Higher
- Network Connectivity

2. SOFTWARE REQUIRMENT SPECIFICATION

1. Introduction

1.1 Purpose:

The objective of **Students Website** is to allow the administrator of any organization to edit and find out the personal details of a student and allows the student to keep up to date his profile .It'll also facilitate keeping all the records of students, such as their id, name, mailing address, phone number, DOB etc. So all the information about the student will be available in a few seconds.

Overall, it'll make Student Information Management an easier job for the administrator and the student of any organization.

The main purpose of this SRS document is to illustrate the requirements of the project **Students Website** and is intended to help any organization to maintain and manage its student's personal data.

1.2 Scope:

Without a **Students Website**, managing and maintaining the details of the student is a tedious job for any organization.

Student Information system will store all the details of the students including their background information, educational qualifications, personal details and all the information related to their resume .

1.3 Modules:

Login module: Login module will help in authentication of user accounts

.Users who have valid login id and password can only login into their respective accounts.

Search module: Suppose there are hundreds of students and from this we have to search a particular student and we know the name of the student .In manual system it is a tedious task though we know the name of the student, but using this module we can easily search the student by specifying the name of the student in the search criteria. Thus this module will help the administrator in searching the student with various criteria easily.

Registration Module and Account Management: This module will help the student get registered from anywhere if internet is present .This module will really simplify the task of on paper registration. Also after successful registration the user can update information and change their password as and when required.

User Management: This module will help the administrator in enabling/disabling a user account and updating user information as required.

Purpose of project is to maintain details of the students such as storing information about:

- Student id
- Student password
- Student name
- Student DOB
- Student mailing address
- Gender
- Registration date
- Student status
- Contact no
- Qualification
- City
- Resume
- Image

1.4 Definitions, Acronyms and Abbreviations:

- **Personal details:** Details of student such as user id, phone number, address, image, resume, e-mail address etc.
- **Contact details:** Details of contact associated with the student.
- **SRS:** System requirement Specification
- **WWW**: World Wide Web
- **Administrator:** A Login Id representing the user is an administrator & can access all the records details

1.5 **Technologies:**

- JSP
- ORACLE
- JAVASCRIPT
- HTML
- CSS

1.6 Overview:

The rest of this SRS is organized as follows:

Section 2 gives an overall description of the software. It gives what level of proficiency is expected of the user, some general constraints while making the software.

Section 3 gives specific requirements which the software is expected to deliver. Some performance requirements and constraints are also given and deal with other Non-Functional Requirements.

Section 4 deals with External Interface Requirements like Hardware and Software Interface.

2. Overall description

2.1 Product Perspective:

The website **Student Website** is aimed towards recording a considerable number of student records and needs online assistance for managing records of students. Website should be user-friendly, 'quick to learn' and reliable website for the above purpose.

Student Website is intended to be a stand-alone product and should not depend on the availability of other website. The system will also have an administrator who has full-fledged rights with regards to performing all actions related to control and management of the website.

2.2 Product Functions:

There are two different users who will be using this product:

- ➢ Administrator who can view and edit the details of any students.
- ➢ Students who can view their details as well as they can edit their details.

The features that are available to the Administrator are:

- ➢ An Administrator can login into the system and perform any of the available operations.
- ➢ Can enable/disable student.
- ➢ Can edit student information to the database.
- ➢ Can make search for a specific student.
- ➢ Can access all the details of the student.

The features that are available to the student are:

- ➢ Student can login into the system and can perform any of the available options.
- ➢ Can view his/her personal details.
- ➢ Can edit his/her personal details
- ➢ Can upload his/her resume.
- ➢ Can upload his/her image.

2.3 User Classes and Characteristics:

There are mainly two kinds of users for the product. The users include:

- ➢ Administrator
- ➢ Student

2.4 Operating Environment:

The product can run on any browser.

2.5 Constraints:

- ➢ Every user must be comfortable using computer.
- ➢ All operations are in English so user must have basic knowledge of English.

2.6 Use case model:

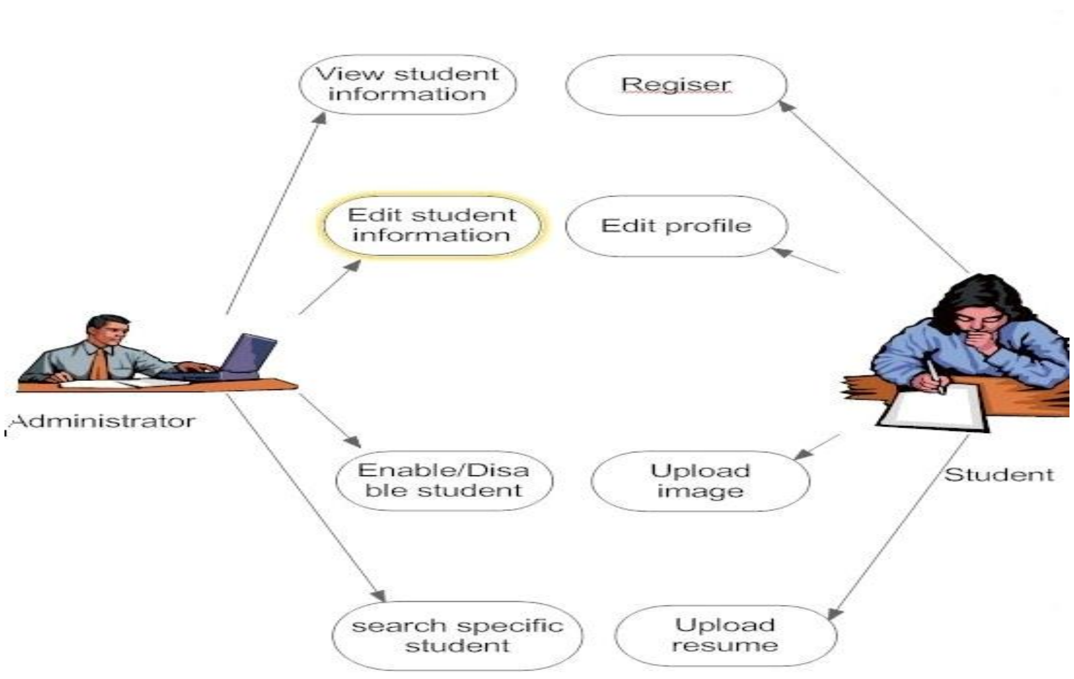


Fig 2(a) : Use Case Model

1. **Administrator:** Responsible for managing student records.

   Login into the website

   - ➢ Update student details
   - ➢ Search student details
   - ➢ Display student details
   - ➢ Enable/Disable student

2. **Student:** Has the access rights to view and edit their personal details.
   - Login into the website
   - Display student details
   - Edit their details
   - Upload their images
   - Upload their resumes

2.7 Assumptions & dependencies

- Administrator is created in the system already.
- Roles and tasks are predefined.

3. Specific Requirements:

3.1 Use Case Reports

1. Administrator: Responsible for managing student details.

Use-case: Login into the website

Goal in context: Gain access to the website

Brief Description: This use case is used when the administrator wants to access the website to enable/disable/update the personal details of the student.

Preconditions: The Administrator must be logged onto the website in order for this use case to begin.

Basic Flow:

- The Website prompts the administrator for the user name and password.
- The Administrator enters the user name and password.
- The Website verifies the password and sets the user’s authorization.
- The Administrator is given access to the Website to perform his tasks.

Alternative Flow:

- The administrator enters invalid username and password then he will not be allowed to enter the website**.**

Post conditions: The website state is unchanged by this use case.

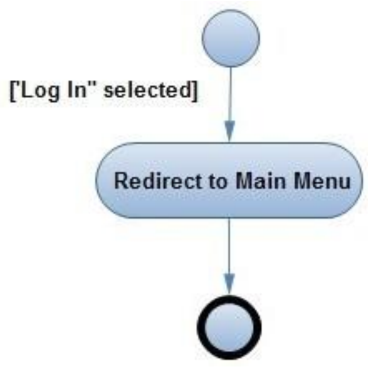


Fig 2(b): Use Case Report- Login into the website

Use Case : Display student details

Goal in context: View the details of a student

Brief Description: This use case is used when the administrator wants to view the personal details of the students already existing in the database on the screen.

Preconditions:

- The Administrator must be logged into the system in order for this use case to begin
- The details of the student must pre-exist in the database
- The student id must be entered correctly.

Basic Flow:

- The Administrator logs onto the System.
- The Administrator search the student from following keys:-
    - Student id
    - First/last name
    - Registration date
    - status
- The System prompts for the student detail from one of the above keys.
- The student details are displayed on the screen.

Alternative Flow:

Student Not Found

If in the **Display a student** sub-flows, a student with the specified id number does not exist, The system displays an error message. The Administrator can then enter a different id number or cancel the operation, at which point the use case ends.

Post conditions:

The student details are displayed on the screen already existing in the system. The state of the system remains unchanged**.**

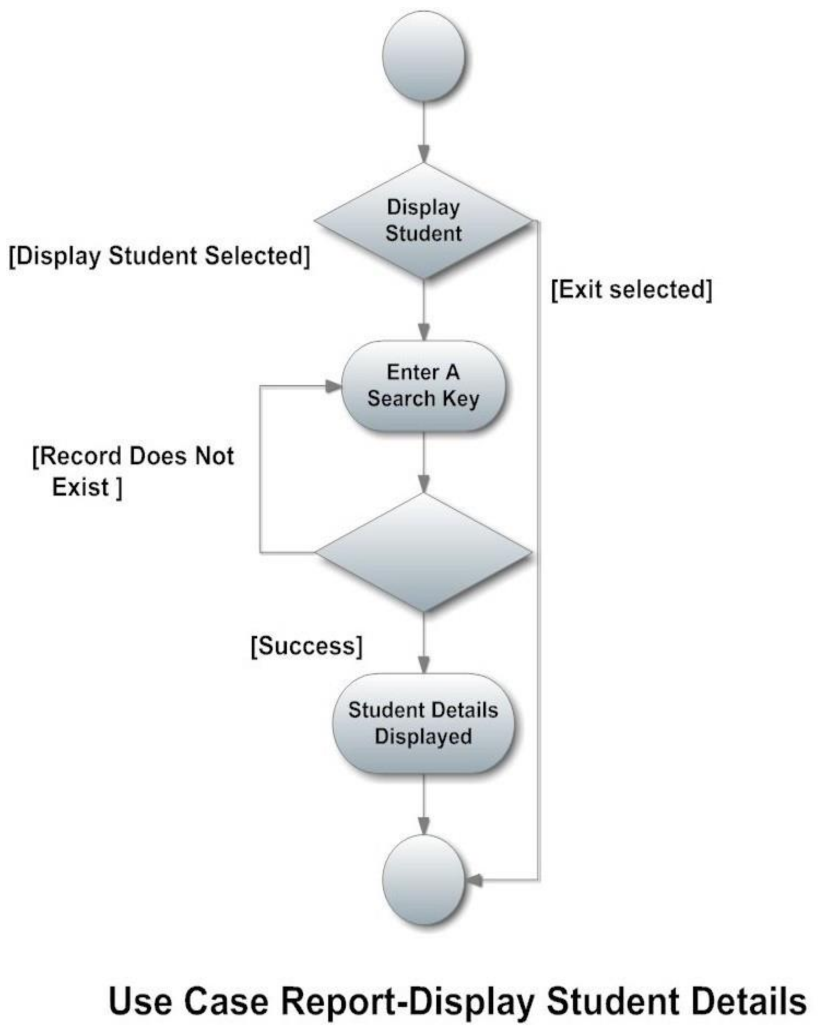

Display Student
[Display Student Selected]
[Exit selected]
Enter A Search Key
[Record Does Not Exist]
[Success]
Student Details Displayed
Use Case Report-Display Student Details

Fig 2(c): Use Case Report-Display Student Details

Use Case : Edit student details

Goal in context: Edit the details of a student

Brief Description: This use case is used when the administrator wants to edit the personal details of the himself/herself already existing in the database.

Preconditions:

- The Administrator must be logged into the system in order for this use case to begin.
- The details of the student must pre-exist in the database

Basic Flow:

- The Administrator logs onto the System.
- The Administrator can edit following keys:-
    - First/last name
    - Gender
    - DOB
    - Contact no
    - Qualification
    - City
    - Email1
    - Email2
    - Address
    - Description
- The Website updates the database according to edited details.
- The student details are edited in the database.

Alternative Flow:

There is no alternative flow of this use case diagram.

Post conditions:

The student details get updated in the database**.**

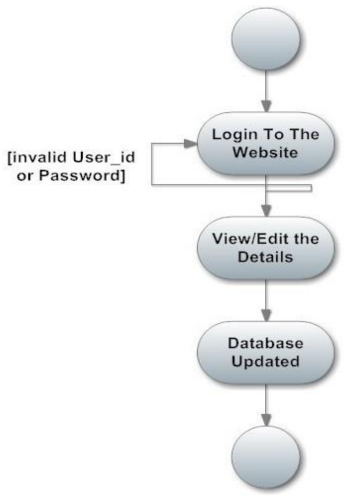


Fig 2(d): Use Case Report- Edit student detail into the website

2. **Student:**

Use Case : student registration

Goal in context: Registration of a student

Brief Description: This use case is used when the student register himself/herself in the database online.

Preconditions:

- ➢ The Student must accessed the website in order for this use case to begin.
- ➢ The user id must be unique and entered correctly.

Basic Flow:

- ➢ The Student enters into the website.
- ➢ The student fill his/her details from the following keys:-
    - Student id
    - password
    - First/last name
    - Status
    - Gender
    - DOB
    - Contact no
    - Qualification
    - City
    - Email1
    - Email2
    - Address
    - Description
    - Resume
    - Image
- ➢ The System details are added to the database.
- ➢ The student details are displayed on the screen.

Alternative Flow:

User ID not unique: if the user id entered is not unique then it will show an error message.

Post conditions:

The students get registered on the website and to login into that particular the administrator must enable it.

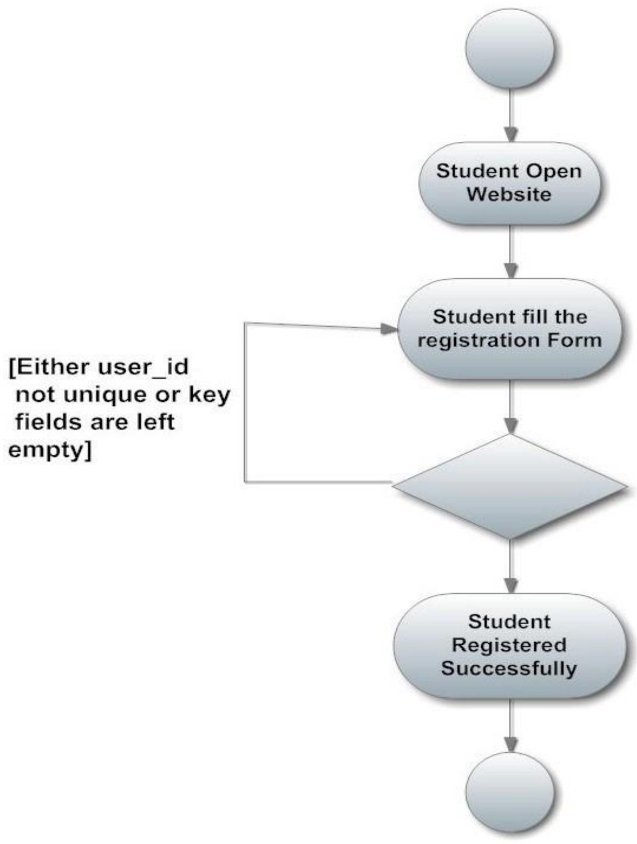

Student Open Website
Student fill the registration Form
[Either user_id not unique or key fields are left empty]
Student Registered Successfully

Fig 2(e): Use Case Report- Register student on website

Use-case: Login into the website

Goal in context: Gain access to the website

Brief Description: This use case is used when the student wants to access the website

Preconditions: The Administrator must enable the particular student onto the website in order for this use case to begin.

Basic Flow:

- The website prompts the student for the user name and password.
- The Student enters the user name and password.
- The website verifies the password and sets the user's authorization.
- The Student is given access to the website to perform his tasks.

Alternative Flow:

The Student enters invalid username and password then he will not be allowed to enter the website**.**

Post conditions: The website state is unchanged by this use case.

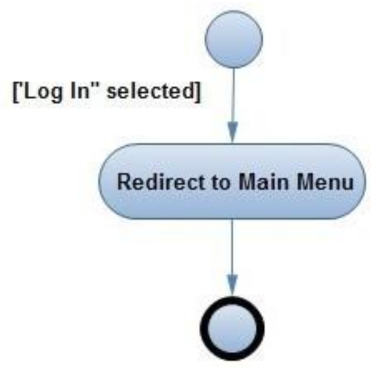


Fig 2(f): Use Case Report- Login into the system

Use Case: Edit student details

Goal in context: Edit the details of a student

Brief Description: This use case is used when the student wants to edit the personal details of the himself/herself already existing in the database.

Preconditions:

- ➢ The Student must be logged into the system in order for this use case to begin.
- ➢ The details of the student must pre-exist in the database
- ➢ The student must be enabled by administrator.

Basic Flow:

- ➢ The Student logs onto the System.
- ➢ The Student can edit following keys:-
  - First/last name
  - Gender
  - DOB
  - Contact no
  - Qualification
  - City
  - Email1
  - Email2
  - Address
  - Description

- ➢ The Website updates the database according to edited details.
- ➢ The student details are edited in the database.

Alternative Flow:

There is no alternative flow of this use case diagram.

Post conditions:

The student details get updated in the database**.**

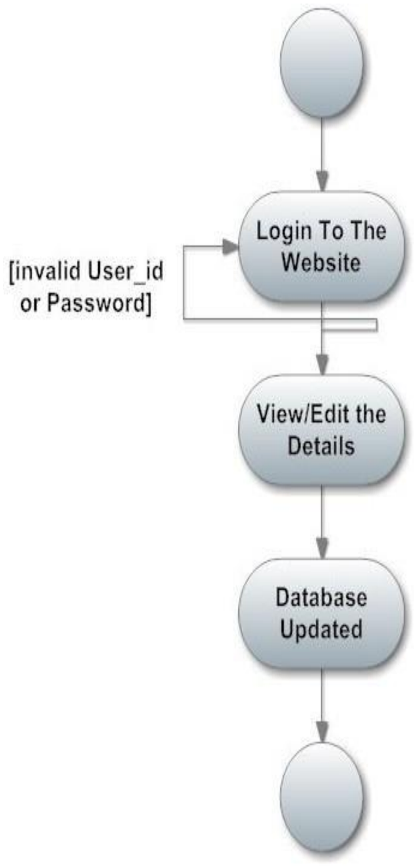


Fig 2(g): Use Case Report- Edit Student Details into Database

<u>Functional Requirements</u> :

- ➢ The Administrator will be given more powers (enable/disable/ update) than other users.
- ➢ It will be ensured that the information entered is of the correct format. For example name cannot contain numbers. In case if incorrect form of information is added, the user will be asked to fill the information again.
- ➢ The system can be accessed anytime.

<u>Non- Functional Requirement</u> :

Performance Requirements:

The proposed system that we are going to develop will be used as the Chief performance system for providing help to the organization in managing the whole database of the student studying in the organisation. Therefore, it is expected that the database would perform functionally all the requirements that are specified.

Safety Requirements:

The database may get crashed at any certain time due to virus or operating system failure. Therefore, it is required to take the database backup.

Security Requirements:

We are going to develop a secured database. There are different categories of users namely Administartor, Student who will be viewing either all or some specific information form the database.

Depending upon the category of user the access rights are decided. It means if the user is an administrator then he can be able to modify the data, append etc. All other users only have the rights to retrieve the information about database.

Conclusion:

This SRS has given all the details of the application need to be built.

4. Design phase

4.1 Introduction

Scope and purpose

The purpose of the design phase is to develop a clear understanding of what the developer want people to gain from his/her project. As you the developer work on the project, the test for every design decision should be "Does this feature fulfill the ultimate purpose of the project?"

A purpose statement affects the design process by explaining what the developer wants the project to do, rather than describing the project itself.

The Design Document will verify that the current design meets all of the explicit requirements contained in the system model as well as the implicit requirements desired by the customer.

Overall System Design Objectives

The overall system design objective is to provide an efficient, modular design that will reduce the system's complexity, facilitate change and result in an easy implementation. This will be accomplished by designing strongly cohesion system with minimal coupling. In addition, this document will provide interface design models that are consistent user friendly and will provide straight forward transition through the various system functions.

Structure of Design Document

- *System Architecture Design* – The System architecture section has detailed diagram of the system, server and client architecture.
- *Data Design* – The data Design include an ERD as well as Database design.
- *Functional Design Description* – This section has the functional partitioning from the SRS, and goes into great detail to describe each function.

4.2 System Architecture Design

System Architecture

The SIMS is a system which contains major part which includes: student Detail, Student image and resume.

The user selects one of the available options as an input to the system. According to the input by the user the system acts and the rest of the functions are performed accordingly.The administartor can operate on any student details.But the normal student or users can only access their details of all the functionalities.

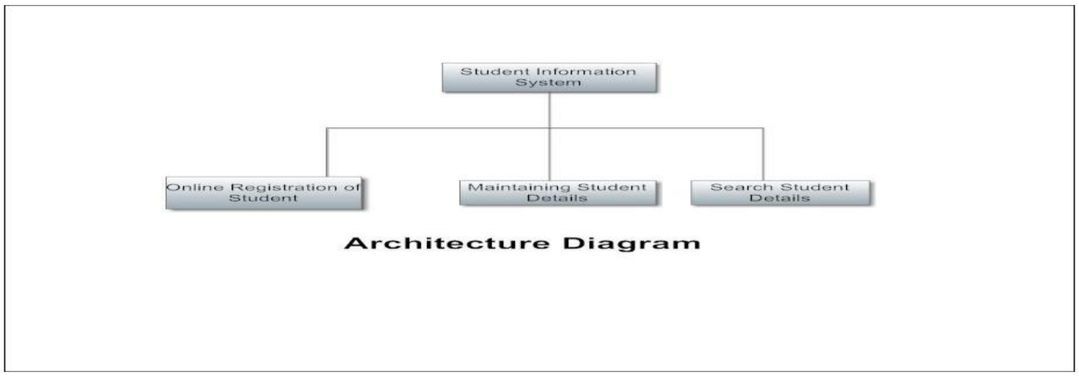

Student Information System
Online Registration of Student
Maintaining Student Details
Search Student Details
Architecture Diagram

Fig 2(h): Architecture Diagram

4.3. Data Design

4.3.1. Entity Relationship Diagram:

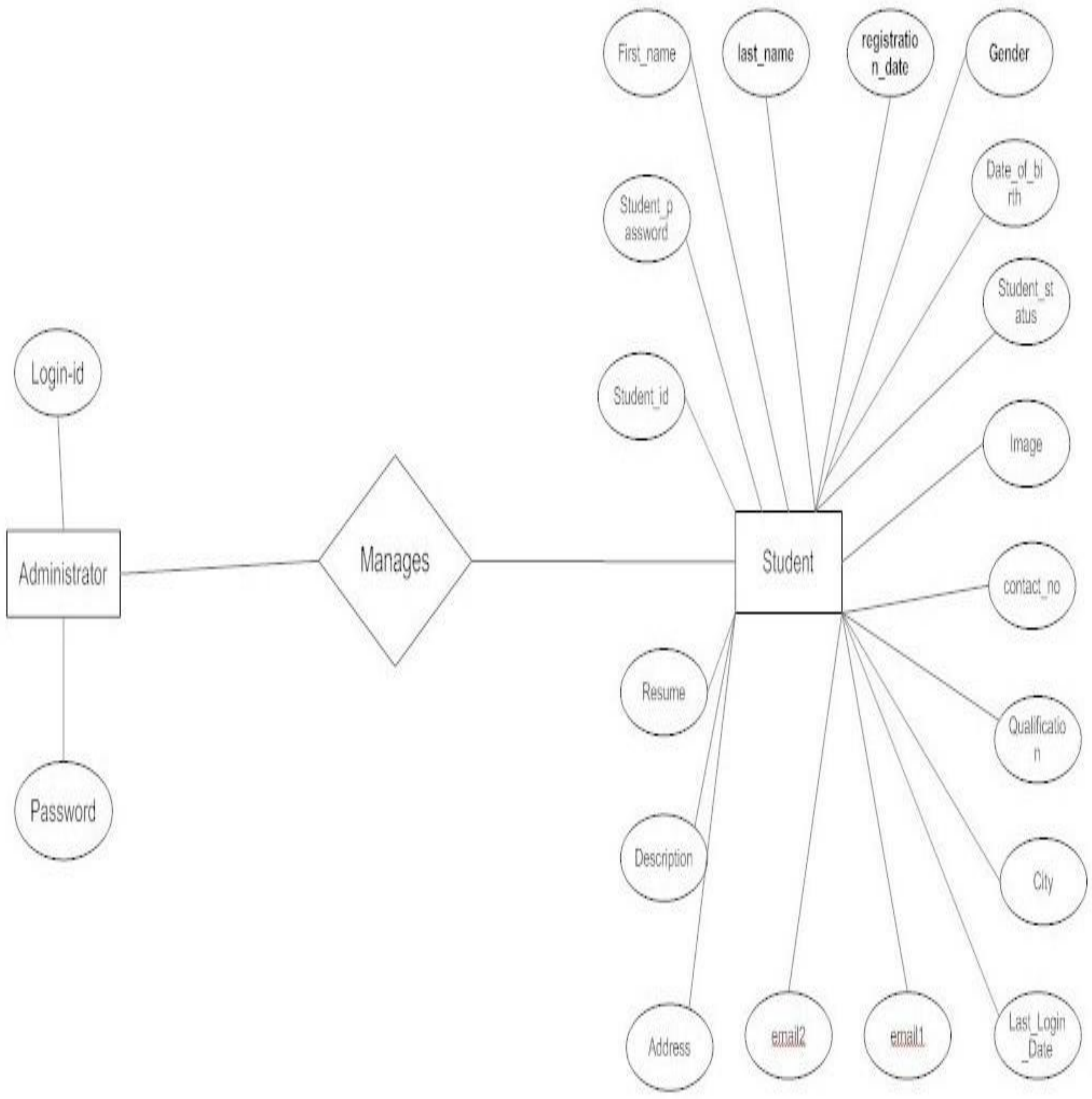


Fig 2(i): Entity Relationship Diagram

4.4. Functional Design Description

4.4.1. Data Flow Diagram :

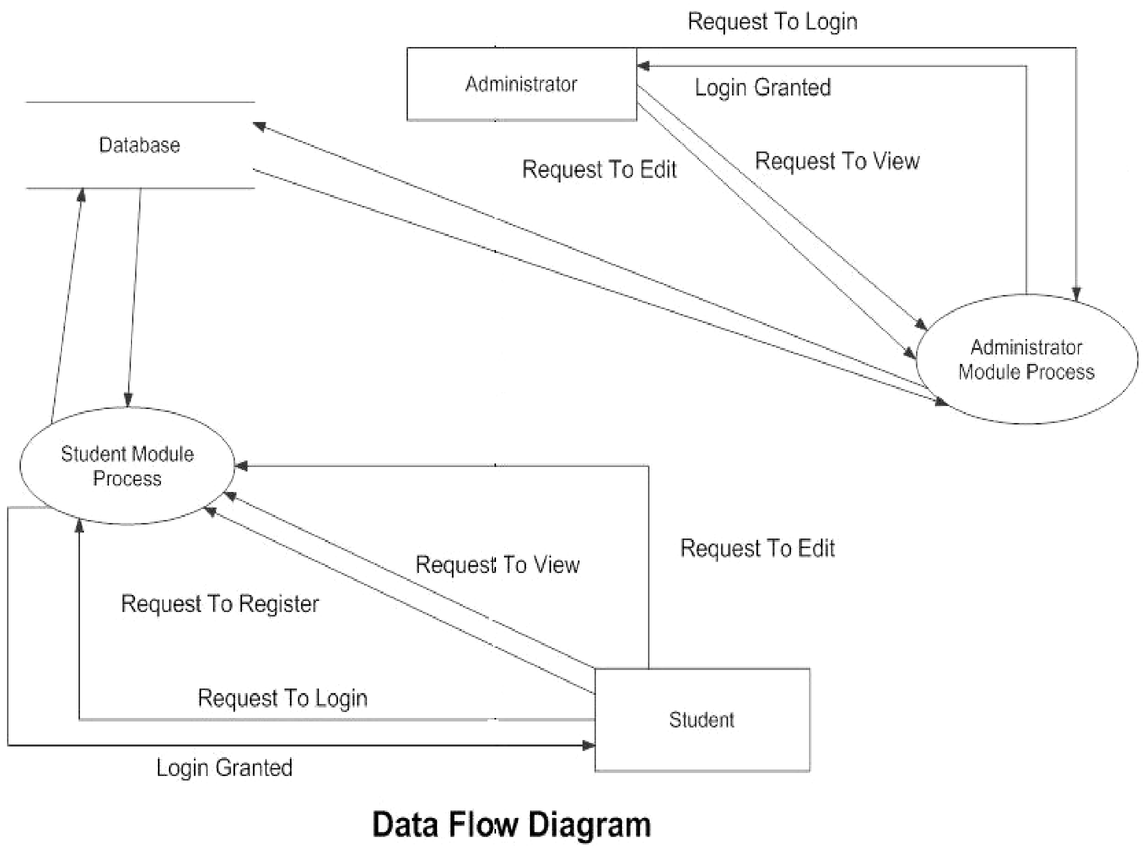


Fig 2(j): Data Flow Diagram

4.5. Decision Tree :

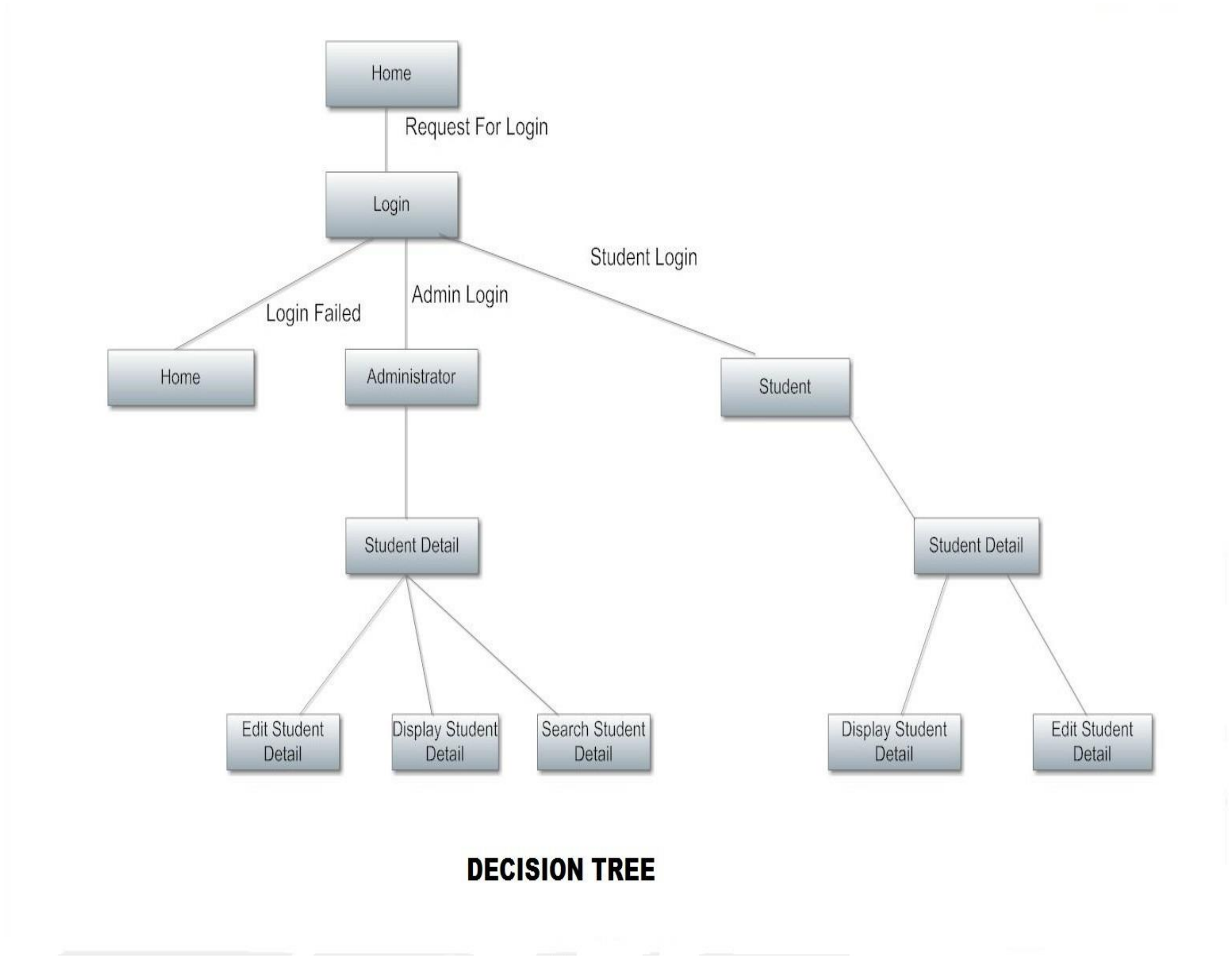


Fig 2(k): Decision Tree

4.6. Conclusion

Hence we can conclude that the design phase of the SIMS give us the information of all the processes used in the project and their relation.

3. TECHNOLOGY OVERVIEW

The technology selected for implementing Student Website is JSP/Oracle. Apache is used as the HTTP server. The development was done in a ‘windows’ environment using adobe dreamweaver CS5.

JSP

JSP stands for **Java Server Pages**. It is a server side technology and used for creating web application. JSP is used to create dynamic web content. In this JSP tags are used to insert JAVA code into HTML pages. It is an advanced version of Servlet Technology. it is a Web based technology helps us to create dynamic and platform independent web pages. In this, Java code can be inserted in HTML/ XML pages or both. JSP is first converted into servlet by JSP container before processing the client’s request.

JSP pages are more advantageous than Servlet:

- They are easy to maintain.
- No recompilation or redeployment is required.
- JSP has access to entire API of JAVA .
- JSP are extended version of Servlet.
- **Coding in JSP is easy** :- As it is just adding JAVA code to HTML/XML.
- **Reduction in the length of Code** :- In JSP we use action tags, custom tags etc.
- **Connection to Database is easier** :-It is easier to connect website to database and allows to read or write data easily to the database.
- **Make Interactive websites** :- In this we can create dynamic web pages which helps user to interact in real time environment.
- **Portable, Powerful, flexible and easy to maintain** :- as these are browser and server independent.
- **No Redeployment and No Re-Compilation** :- It is dynamic, secure and platform independent so no need to re-compilation.
- **Extension to Servlet** :- as it has all features of servlets, implicit objects and custom tags

ORACLE

ORACLE is a fourth generation relational database management system. In general, a database management system (DBMS) must be able to reliably manage a large amount of data in a multi-user environment so that many users can concurrently access the same data. All this must be accomplished while delivering high performance to the users of the database. A DBMS must also be secure from unauthorized access and provide efficient solutions for failure recovery. The ORACLE Server provides efficient and effective solutions for the major database features.

ORACLE consists of many tools that allow you to create an application with ease and flexibility. You must determine how to implement your requirements using the features available in ORACLE, along with its tools. The features and tools that you choose to use to implement your application can significantly affect the performance of your application.

Several of the more useful features available to ORACLE application developers are integrity constraints, stored procedures and packages, database triggers, cost-based optimizer, shared SQL, locking and sequences.

This documentation will lead you through the main features and tools of ORACLE. It is intended to give you a partial view of what is available to you to use within the assignments.

This documentation will cover:

- ORACLE Architecture - provides a basic understanding of the ``Big Picture'' including the concepts and terminology of the ORACLE Server.
- Starting ORACLE And Other Important Information - provides the knowledge of how to set up your account and other system environment variables. It will also provide information about how ORACLE is currently setup, which you will require, and the steps you must take to report any problems.
- SQL*Plus (Terminal Monitor) - provides a summary of the commands that you will require in order to create tables and manipulate the database.
- SQL*Loader - provides a summary of the commands that you will require to allow you to load data from a file to the database.
- SQL Commands - provides the syntax of some of the SQL Commands in ORACLE to help you get started. This section will only shed light on the Data Definition Language commands and will not contain any information on querying the database (which should be covered in class).

Use this manual to give you an introduction and reference, and peruse whatever code the T.A. gives you, but above all else experiment! No amount of documentation could hope to encapsulate all of the little ins and outs that some good old fashioned fiddling around will find. Try to remember that a few thousand pages were boiled down to this thin tome that you are holding, so if you think that this book is the be-all and end-all of ORACLE knowledge... Well, you'll find out soon enough.

Apache

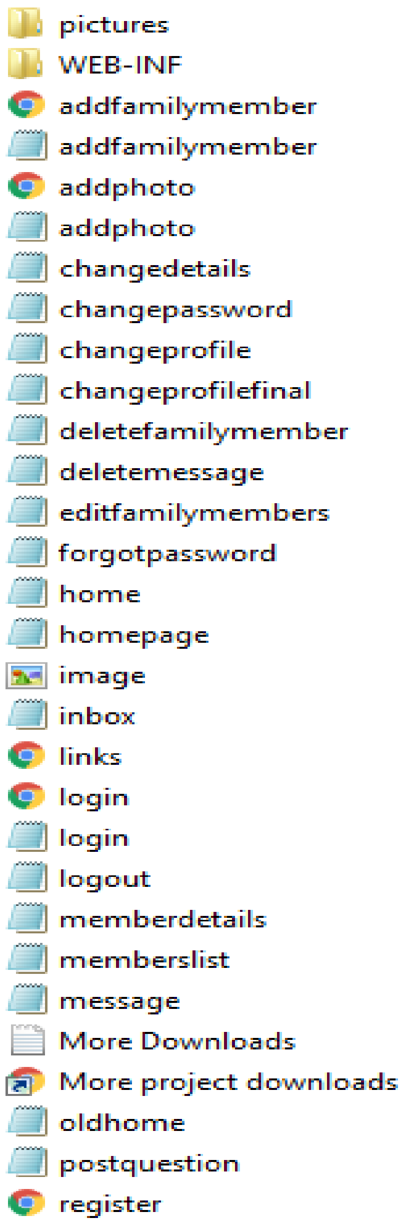
The Apache Tomcat software is an open source implementation of the Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket technologies. Apache Tomcat is usually used as a Servlet Container even though Tomcat has a fully functional HTTP Server to serve static content. In most of production, Tomcat is used in conjunction with Apache HTTP Server where Apache HTTP Server attends static content like html, images etc., and forwards the requests for dynamic content to Tomcat. This is because Apache HTTP Server supports more advanced options than that of Tomcat.

4. PROJECT DESCRIPTION

1. Introduction

Student Website can be used by education institutes to maintain the records of students easily. Achieving this objective is difficult using a manual system as the information is scattered, can be redundant and collecting relevant information may be very time consuming. All these problems are solved using this project

The directory structure of the project is as follows:



```
pictures
WEB-INF
addfamilymember
addfamilymember
addphoto
addphoto
changedetails
changepassword
changeprofile
changeprofilefinal
deletefamilymember
deletemessage
editfamilymembers
forgotpassword
home
homepage
image
inbox
links
login
login
logout
memberdetails
memberslist
message
More Downloads
More project downloads
oldhome
postquestion
register
```

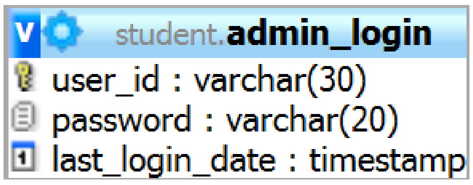
.

Description of root directory contents

- **Pictures Directory:** This directory contains the images uploaded by the students during registration process.Supported formats are the .jpg and .gif files.
- **WEB-INF Directory**: This Directory Contains resumes of students uploaded during registration process of students.Files in this folder can be of .doc,.txt or
  .pdf format.
- **Addfamilymember.jsp** : Addfamilymember page for adding information of family member of a student.
- **Addphoto.jsp** : Addphoto page for adding photo of a student.
- **Changedetails.jsp** : This page allows the user to change their details
- **Changepassword.jsp:** This page allows the users if they want to change their passwords.
- **Changeprofile.jsp** : . This page allows the users if they want to change their profile.
- **Changeprofilefinal.jsp:** This page allows the users if they want to change their profile final.
- **Deletefamilymember.jsp** : This page allows the users if they want to delete their information of familymember.
- **Deletemessage.jsp** : This page allows the users if they want to delete their messages.
- **Editfamilymember.jsp**: This page allows the users if they want to edit their family member details.
- **Forgotpassword.jsp:** . This page allows the users to generate their new password if they forgot their password.
- **Homepage.jsp** : Homepage of the website.
- **Inbox.jsp** : This page allows the users to show their information.
- **Login.jsp** : This page allows the users to login into the website.
- **Logout.jsp** : This page is for user to quit from the website.

Description of database tables

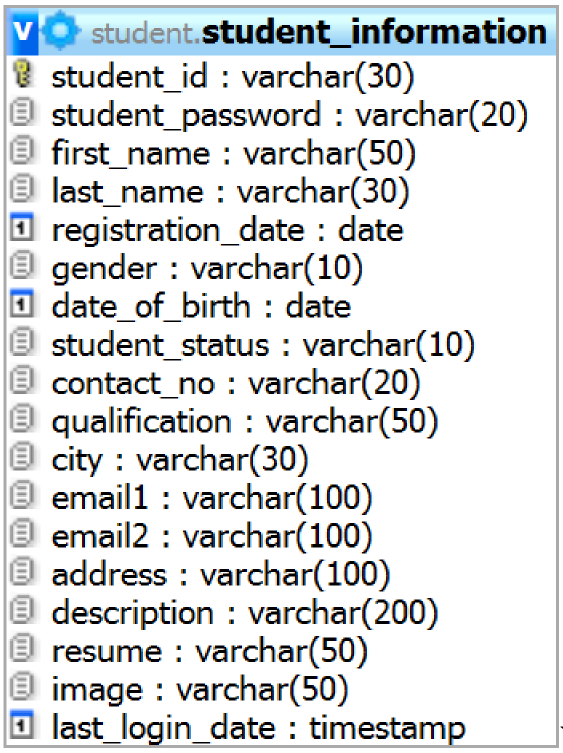
- **admin_login :**



student.admin_login
- user_id : varchar(30)
- password : varchar(20)
- last_login_date : timestamp

  - user_id : Stores user id of administrator(s).
  - password : Stores password of the administrator(s).
  - last_login_date : Stores the last login date of the administrator(s).

- **Student_information :**



student.student_information
- student_id : varchar(30)
- student_password : varchar(20)
- first_name : varchar(50)
- last_name : varchar(30)
- registration_date : date
- gender : varchar(10)
- date_of_birth : date
- student_status : varchar(10)
- contact_no : varchar(20)
- qualification : varchar(50)
- city : varchar(30)
- email1 : varchar(100)
- email2 : varchar(100)
- address : varchar(100)
- description : varchar(200)
- resume : varchar(50)
- image : varchar(50)
- last_login_date : timestamp

- student_id : Stores user id of the student(s)
- student_password : Stores password of the student(s)
- first_name : Stores first name of the student(s)
- last_name : Stores last name of the student(s)
- registration_date : Stores the registration date of the student(s).
- gender : Stores the gender of the student(s).
- date_of_birth : Stores the date of birth of the student(s).
- student_status : Stores the current status of the student account(s).
- contact_no : Stores the contact number of the student(s).
- qualification : Stores student(s) qualification.
- city : Stores the city in which the student(s) lives.
- email1 : Stores primary email of the student(s).
- email2 : Stores secondary email of the student(s).
- address : Stores the address of the student(s).
- description : Stores description of the student(s).
- resume : Stores resume of students(s).
- image : Stores image of the student(s).
- last_login_date : Strores last login date of the student(s).

Features

The Website provides following functionalities to the users :

- **Administrator** :

  - Login/Logout
  - View student information
  - Edit Student Information
  - Enable/disable student accounts
  - Search students

- ♣ **Student** :

    - o Login/Logout
    - o View profile
    - o Edit profile
    - o Change password
    - o Register new profile

Source Code:

home.jsp:

```
<frameset cols="25%,*" noresize>
  <frame src="links.html">
  <frame src="homepage.jsp" name="details">
</frameset>
```

homepage.jsp

```
<html>
<body>
<link rel="stylesheet"   href="style.css">
<jsp:useBean id="member" scope="session"  class="classmates.MemberBean" />
<%@ page import="java.util.*,java.sql.*,classmates.*"%>
<div style="background-color:navy;color:white;font:700 12pt verdana">
Welcome, <jsp:getProperty name="member" property="fullname"/>
</div>
<h2>Inbox</h2>
<table width=100% border=1>
<tr style="background-color:maroon;color:wheat">
<th>Date
<th>Subject
<th>From
</tr>
<%
Connection con = Database.getConnection();
```

```
PreparedStatement ps = con.prepareStatement("select msgid,fullname,subject,senton from
messages msg,members m  where msg.sender = m.mid and  receipient = ? order by senton
desc");
ps.setString(1,member.getMid());
ResultSet rs = ps.executeQuery();
while ( rs.next())
   {
%>
<tr>
<td><%=rs.getString("senton")%>
<td><a href=message.jsp?msgid=<%=rs.getString("msgid")%>>
<%=rs.getString("subject")%>
</a>
<td><%=rs.getString("fullname")%>
</tr>
<%
   }
   rs.close();
   ps.close();
   con.close();
%>
</table>
</body>
</html>
```

<u>login.jsp</u>

```
<html>
<link rel="stylesheet" href="style.css">
<body>
<h1 align="center">CLASSMATES.COM </h1>
<table width="100%" height="100%">
<tr>
<td width="30%" style="background-color:tan" valign="top">
```

```
<h3>Login </h3>
<form  action="login.jsp" method=post>
<b>Email Address</b>
<br>
<input type=text size=30 name=email>
<br>
<b>Password</b>
<br>
<input type=password  size=20 name=pwd>
<p>
<input type=submit value="Login">
</form>
<%
if ( request.getParameter("email") != null )
{
%>
<jsp:useBean id="member" class="classmates.MemberBean" scope="session" />

<jsp:setProperty name="member"
   property="*" />
  <%
   if  ( member.login() )
   {
    response.sendRedirect("home.jsp");
   }
   else
   {
     out.println("<p><span style=color:red>Invalid Login</span>");
    }
  }
%>
<p>
<a href="forgotpassword.jsp">Forgot Password?</a>
<p>
```

```
<a href="register.html">New User? Register!</a>
</td>
<td valign="top">
<h3>About Classmates.com </h3>
This website allows students of XYZ college to get details about their classmates.
<p>
<h4>Contact Details: </h4>
<img src="image.gif">
</td>
</tr>
</table>
</body>
</html>
```

registration.jsp

```
<html>
<jsp:useBean id="member" class="classmates.MemberBean" scope="session" />
<jsp:setProperty  name="member" property="*" />
<%
   try
   {
       member.registerUser();
       out.println("<h4>User Registration Is Sucessful. Please click <a
href=home.jsp>here</a> to continue..</h4>");
   }
   catch(Exception ex)
   {
  out.println("<h4>Error Occurred During User Creation. Error : " + ex.getMessage() +
"<P>Use BACK button to try again!</h4>");
   }
%>
```

__**changedetails.jsp**__

```
<jsp:useBean id="member" class="classmates.MemberBean" scope="session" />
<jsp:setProperty  name="member" property="*" />
<%
  if ( request.getParameter("phone") != null)
  {
    // update data in the table with the changes made in the form
    member.updateDetails();
  }
%>
<html>
<link rel="stylesheet"  href="style.css">
<%
 // get details from table - MEMBERS
  %>
 <body>
<center>
<h2>Change Details </h2>
<hr>
<form action="changedetails.jsp" method="post">
<table>
<tr>
<td>
Phone Number
<td>
<input type=text name=phone size=30 value='<%=member.getPhone()%>'>
</tr>
<tr>
<td>
Email
<td>
<input type=text name=email size=30 value='<%=member.getEmail()%>'>
</tr>
```

```
<tr>
<td>
Address
<td>
<textarea  name=address rows=5 cols=30><%=member.getAddress()%></textarea>
</tr>
<tr>
<td>
Occupation
<td>
<input type=text name=occup size=50 value='<%=member.getOccup()%>'>
</tr>
<tr>
<td>
Qualification
<td>
<input type=text  name=qual  size=30  value='<%=member.getQual()%>'>
</tr>
</table>
<p>
<input type=submit value="Change">
<input type=reset value="Clear All">
<p>
<a href="home.jsp">Go Back</a>
</form>
</center>
</body>
</html>
```

changepassword.jsp

```
<jsp:useBean id="member" class="classmates.MemberBean" scope="session" />

<link rel="stylesheet"  href="style.css">
<html>
<script language="javascript">

function check()
{
   if ( f1.newpwd.value !=  f1.confirmpwd.value )
   {
         alert("New password and confirm password are not matching.");
         f1.confirmpwd.focus()
         return false;
   }
   return true;
}
</script>
  <body>
<h2>Change Password </h2>
<form name=f1  onsubmit="return check()" action="changepassword.jsp" method="post">
<table>
<tr>
<td>
Old Password
<td>
<input type=password name=oldpwd size=15>
</tr>
<tr>
<td>
New Password
<td>
<input type=password name=newpwd  size=15>
```

```
</tr>
<tr>
<td>
Confirm New Password
<td>
<input type=password name=confirmpwd  size=15>
</tr>
</table>
<p>
<input type=submit value="Change Password">
</form>
<%
    // process input
    String oldpwd = request.getParameter("oldpwd");
   String newpwd = request.getParameter("newpwd");

   if (oldpwd == null ) return;

   if (! oldpwd.equals(member.getPwd()) )
   {
     out.println("Sorry! Invalid Password!");
     return;
   }
     if(member.updatePassword(newpwd) )
     out.println("Password Changed Successfully!");
   else
     out.println("Sorry! Some problem occurred while changing pasword!");
 %>
</body>
</html>
```

addphoto.jsp

```
<%@ page import="org.apache.commons.fileupload.*,java.util.*,java.io.*"%>
<html>
<link rel="stylesheet"  href="style.css">
<body>
<h2>Upload Member Photo</h2>
<form action="addphoto.jsp"  method="post"  enctype="multipart/form-data">
Select a .jpg file :  <input type=file name=file>
<p>
<input type=submit value="Upload">
</form>
<p>
<jsp:useBean id="member" class="classmates.MemberBean" scope="session" />
<%
      // Check that we have a file upload request
  boolean isMultipart = FileUpload.isMultipartContent(request);
  if ( !isMultipart ) return;
      // Create a new file upload handler
  DiskFileUpload upload = new DiskFileUpload();

  // parse request
  List items = upload.parseRequest(request);
   // get uploaded file
  FileItem  file = (FileItem) items.get(0);
  File outfile = new File( request.getRealPath("pictures") + "\\" +  member.getMid() +
".jpg");
  file.write(outfile);
    out.println("Photo Uploaded Successfully!");
%>
</body>
</html>
```

addfamiltmember.jsp

```
<%@ page import="java.sql.*"%>
<html>
<link rel="stylesheet"  href="style.css">
<body>
<jsp:useBean id="member" class="classmates.MemberBean" scope="session" />

<jsp:useBean id="familymember"  class="classmates.FamilyMember"  scope="page" />
<jsp:setProperty name="familymember" property="*"/>

<table border=1 width="100%">
<tr style="background-color:navy;color:white;font:700 12pt verdana">
<td>Result </td>
</tr>
<tr>
<td>
<%
  String msg = familymember.add(member.getMid());
  if ( msg == null)
     out.println("Family Member Is Added Successfully!");
  else
     out.println("Error : " + msg);
 %>
<p>
To add another member click <a href="addfamilymember.html">here</a>
<p>
To goto home page click <a href="homepage.jsp">here</a>
</td>
</tr>
</table>
</body>
</html>
```

deletefamilymember.jsp

```
<html>
<body>
<link rel="stylesheet"  href="style.css">
<jsp:useBean id="member" class="classmates.MemberBean" scope="session" />
<jsp:useBean id="family" class="classmates.FamilyMember" scope="page" />
<jsp:setProperty name="family" property="*"/>
<table border=1 cellpadding=5  width=100%>
<tr style="color:white;background-color:navy;font:700 12pt verdana">
<td>Result</td>
<tr>
<td>
<%
    String msg = family.delete(member.getMid());
    if ( msg == null)
    out.println("Family Member Details Deleted Successfully!");
    else
      out.println("Error : " + msg);
%>
  </tr>
</table>
<p>
<a href="editfamilymembers.jsp">Family Members </a>
</body>
</html>
```

message.jsp

```
<html>
<body>
<link rel="stylesheet"  href="style.css">

<jsp:useBean id="member" class="classmates.MemberBean" scope="session" />

<%@page import="java.sql.*,classmates.*"%>

<h2>Message Details</h2>

<%
   String msgid = request.getParameter("msgid");

   Connection con = Database.getConnection();
   PreparedStatement ps = con.prepareStatement("select fullname,sender,subject,body,senton
from messages ms,members m  where ms.sender = m.mid and  msgid  = ?");
   ps.setString(1,msgid);
   ResultSet rs = ps.executeQuery();
   rs.next();
%>

<table width=100% border=1>
<tr>
<td>
Sender
<td>
<a href=memberdetails.jsp?mid=<%=rs.getString("sender")%>>
<%=rs.getString("fullname")%></a>
</tr>
<tr>
<td>Subject
<td><%=rs.getString("subject")%>
```

```
</tr>
<tr>
<td>Sent On
<td><%=rs.getString("senton")%>
</tr>
<tr>
<td>Body
<td><pre><%=rs.getString("body")%></pre>
</tr>
</table>
<p>
<a href=deletemessage.jsp?msgid=<%=msgid%>>Delete</a>
&nbsp;&nbsp;
<a href=sendmessage.jsp?mid=<%=rs.getString("sender")%>>Reply</a>
&nbsp;&nbsp;
<a href=homepage.jsp>Inbox</a>
<%
   rs.close();
   ps.close();
   con.close();
%>
```

sendmessage.jsp

```
<%@ page import="java.sql.*"%>
<html>
<link rel="stylesheet"  href="style.css">
<body>
<h2>Send Message</h2>
<jsp:useBean id="member" class="classmates.MemberBean" scope="session" />
<jsp:setProperty  name="member" property="*" />
<form action="sendmessage.jsp" method=post>
<input type=hidden value=<%=request.getParameter("mid")%> name="mid" >
<table cellpadding=3>
```

```
<tr>
<td>
Subject
<td>
<input type=text name=subject size=50>
</tr>
<tr>
<td>Body
<td>
<textarea name=body  cols=50 rows=5></textarea>
</td>
</tr>
</table>
<input type=submit value="Send">
<%
  // return if it is first time
  if ( request.getParameter("subject") == null)  return;
    String mid = request.getParameter("mid");
%>
<jsp:useBean id="message"  class="classmates.MessageBean"  scope="page" />
<jsp:setProperty name="message" property="*"/>
<p>
<table border=1 cellpadding=5 width=100%>
<tr style="color:white;background-color:navy;font:700 10pt verdana">
<td>Result</td>
</tr>
<tr>
<td>
<%
  String msg  = message.send(member.getMid(),mid);
  if ( msg == null)
     out.println("Member has been sent successfully");
  else
      out.println("Error : " + msg);
```

```
 %>
</tr>
</table>
</body>
</html>
```

sendmessageform.jsp

```
<%@ page import="java.sql.*,classmates.*"%>
<html>
<link rel="stylesheet"  href="style.css">
<body>
<h2>Send Message</h2>
<jsp:useBean id="member" class="classmates.MemberBean" scope="session" />
<form action="sendmessage.jsp" method=post>
Select Member
<select name=mid>
<%
   Connection con  = Database.getConnection();
   PreparedStatement ps = con.prepareStatement("select mid,fullname || ',' || branch || ',' || year
from members where  mid <> ? order by fullname");
   ps.setString(1, member.getMid());
   ResultSet rs = ps.executeQuery();
   while ( rs.next())
   {
%>
<option  value=<%=rs.getString("mid")%>> <%=rs.getString(2)%></option>
<%
   }
   rs.close();
   ps.close();
   con.close();
%>
</select>
```

```
<p>
<input type=submit value="Continue">
</form>
</body>
</html>
```

<u>sendmessageold.jsp</u>

```
  <%@ page import="java.sql.*"%>
<html>
<link rel="stylesheet"  href="style.css">
<body>
<h2>Send Message</h2>
<jsp:useBean id="member" class="classmates.MemberBean" scope="session" />
<form action="sendmessage.jsp" method=post>
<table cellpadding=3>
<tr>
<td> To
<td>
<select name=receiver>
<%
  Connection con  = member.getConnection();
  PreparedStatement ps = con.prepareStatement("select lname,fullname from members where
lname <> ?");
  ps.setString(1, member.getLname());
  ResultSet rs = ps.executeQuery();
  while ( rs.next())
  {
%>
<option  value=<%=rs.getString("lname")%>> <%=rs.getString("fullname")%>
<%
  }
  rs.close();
  ps.close();
  con.close();
%>
</select>
</tr>
<tr>
<td>
Subject
<td>
<input type=text name=subject size=50>
</tr>
<tr>
<td>Body
<td>
<textarea name=body  cols=50 rows=5></textarea>
</td>
</tr>
</table>
<input type=submit value="Add">
<input type=reset value="Clear All">
<p>
<a href="home.jsp">Home Page</a>
<%
  // return if it is first time
  if ( request.getParameter("subject") == null)   return;
%>
<jsp:useBean id="message"  class="classmates.MessageBean"  scope="page" />
<jsp:setProperty name="message" property="*"/>
<h4>
<%
  String msg  = message.add(member);
  if ( msg == null)
     out.println("Member has been sent successfully");
  else
     out.println("Error : " + msg);
```

```
 %>
</table>
</body>
</html>
```

Inbox.jsp

```
<link rel="stylesheet"  href="style.css">
<jsp:useBean id="member" class="classmates.MemberBean" scope="session" />
<%@page import="java.sql.*"%>
<h2>InBox</h2>
<%
 Connection con = member.getConnection();
 PreparedStatement ps = con.prepareStatement("select
mid,fullname,sender,subject,body,senton from messages,members  where messages.sender =
members.lname and  receipient = ?");
 ps.setString(1,member.getLname());
 ResultSet rs = ps.executeQuery();
 while ( rs.next())
  {
%>
<table width=100%>
<tr>
<td style="color:blue">
Sender
<td><%=rs.getString("fullname")%>
</tr>
<tr>
<td style="color:blue">Subject
<td><%=rs.getString("subject")%>
</tr>
<tr>
<td style="color:blue">Sent On
<td><%=rs.getString("senton")%>
```

```
</tr>
</table>
<pre>
<%=rs.getString("body")%>
</pre>
<a href=deletemessage.jsp?mid=<%=rs.getInt("mid")%>>Delete</a>
<hr>
```

deletemessage.jsp

```
<html>
<body>
<jsp:useBean id="message" class="classmates.MessageBean" scope="page" />
<h5>
<%
  String msgid =request.getParameter("msgid");
  String msg =  message.delete(msgid);
  if ( msg == null)
    response.sendRedirect("homepage.jsp");
  else
    out.println("Error Occured During Deletion Of Message : " + msg);
%>
</h5>
</body>
</html>
```

searchclassmates.jsp

```
<jsp:useBean id="member" scope="session"   class="classmates.MemberBean"/>
<%@ page import="java.sql.*,classmates.*"%>
<html>
<link rel="stylesheet" href="style.css">
<body>
<h2>Search Classmates </h2>
<%
  String  fullname =  request.getParameter("fullname");
  if ( fullname == null) fullname= "";
  String  branch=  request.getParameter("branch");
  if (branch == null)  branch = "ALL";
  String  year =  request.getParameter("year");
  if ( year== null)  year = "";
  String  address =  request.getParameter("address");
  if ( address == null)  address = "";
%>
<form action="searchclassmates.jsp" method="post">
<table>
<tr>
<td>Name Contains
<td><input type=text size=30  value='<%=fullname%>' name=fullname>
</tr>
<tr>
<td>Branch
<td>
<select name=branch>
<option  value="ALL" <%= branch.equals("ALL")?"SELECTED":"" %> >ALL</option>
<option value="BECS" <%= branch.equals("BECS")?"SELECTED":"" %> >BE
CS</options>
<option value="BEEC" <%= branch.equals("BEEC")?"SELECTED":"" %> >BE
EC</options>
<option value="MCA" <%= branch.equals("MCA")?"SELECTED":"" %> >MCA</options>
```

```
</select>
Year of Passing
<input type=text size=10 value='<%=year%>' name=year>
</td>
</tr>
<tr>
<td>Address Contains
<td><input type=text size=30 value='<%=address%>' name=address>
</tr>
</table>
<p>
<input type=submit value=Search>
</form>
<%
  if (request.getParameter("fullname") == null)
  return;
  String  cond = " mid <> " + member.getMid();
  if  ( fullname.length() >  0 )
     cond = cond + "  and  fullname  like '%"  + fullname + "%'";
  if  ( ! branch.equals("ALL") )
     cond = cond + "  and  branch = '"  + branch+ "'";
  if  ( address.length() >  0 )
     cond = cond + "  and address like '%"  + address + "%'";
  if  ( year.length() >  0 )
     cond = cond + "  and year = "  + year;
  String query = "select  * from members where " +  cond;
%>
 <table border=1 cellpadding="5" width=100%>
<tr>
<th>Full Name
<th>Occupation
<th>Branch
<th>Year
</tr>
```

```
   <%
     Connection con = Database.getConnection();
     Statement st = con.createStatement();
     ResultSet rs = st.executeQuery(query);
     while ( rs.next())
     {
%>
 <tr>
<td>
<a href=memberdetails.jsp?mid=<%=rs.getString("mid")%>>
<%=rs.getString("fullname")%></a>
<td><%=rs.getString("occup")%>
<td><%=rs.getString("branch")%>
<td><%=rs.getString("year")%>
</tr>
   <%
    }
    rs.close();
    st.close();
   con.close();
%>
</table>
</body>
</html>
```

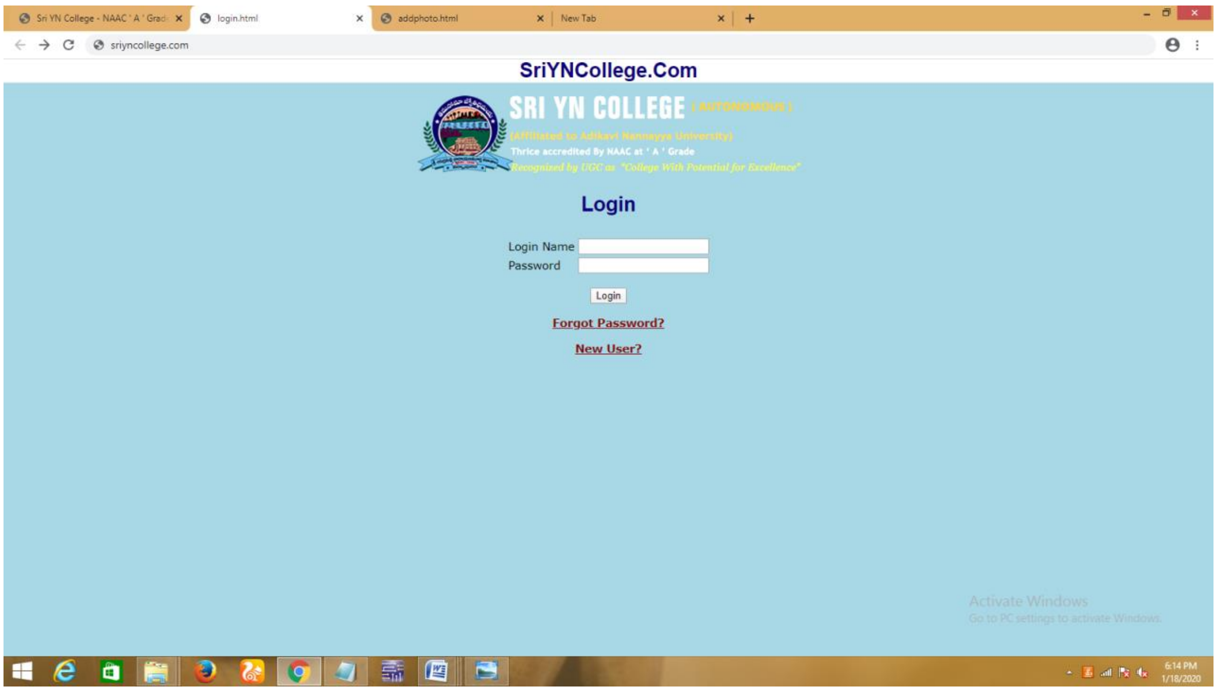
style.jsp

```
td {font:10pt verdana}
th {font:700 10pt verdana}
a  {color:black;font:700 10pt verdana}
a:hover  {color:red;font:700 10pt verdana}
h1 {color:wheat;font:700 30pt arial;letter-spacing:10pt;background-color:maroon}
h2 {color:black;font:700 18pt arial}
h3 {color:navy;font:700 16pt arial}
h4 {color:balck;font:700 12pt arial}
body {background-color:wheat;font:10pt verdana}
```

logout.jsp

```
<%
   session.invalidate();
   response.sendRedirect("login.jsp");
%>
```

5. SNAPSHOTS


SriYNCollege.Com

SRI YN COLLEGE (AUTONOMOUS)

(Affiliated to Adikavi Nannayya University)

Thrice accredited By NAAC at ' A ' Grade

Recognized by UGC as "College With Potential for Excellence"

Login

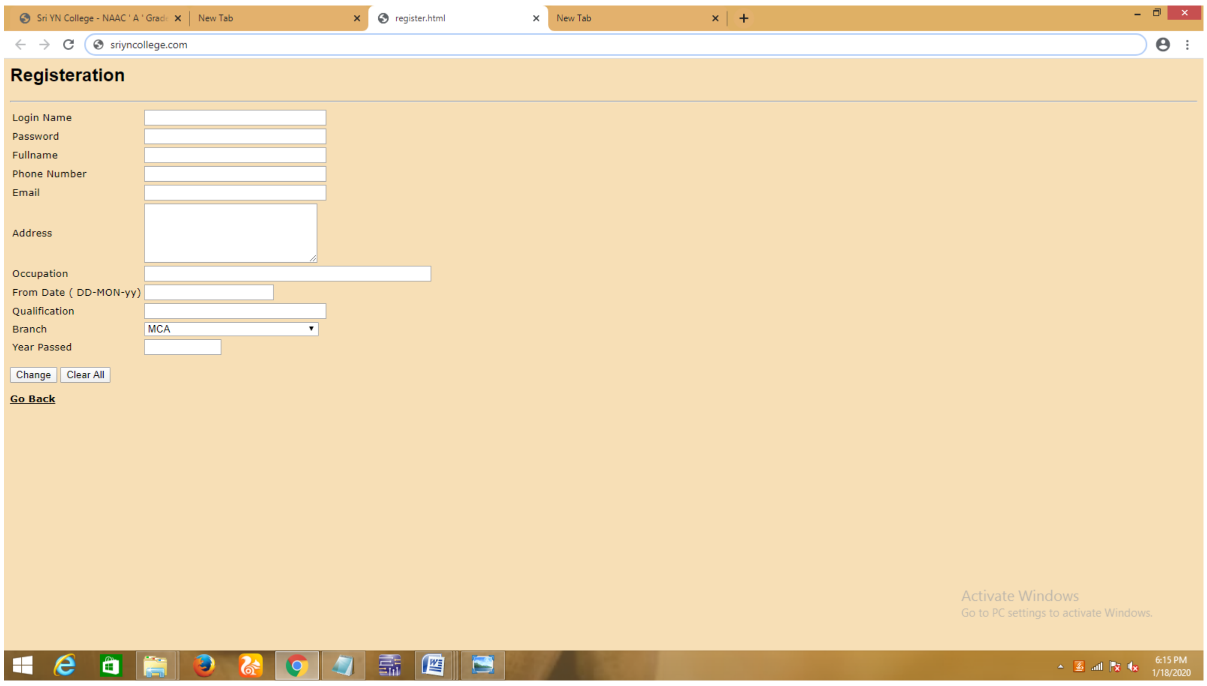
Login Name	
Password	

Login

Forgot Password?

New User?

Fig 5(a): This is the student login page

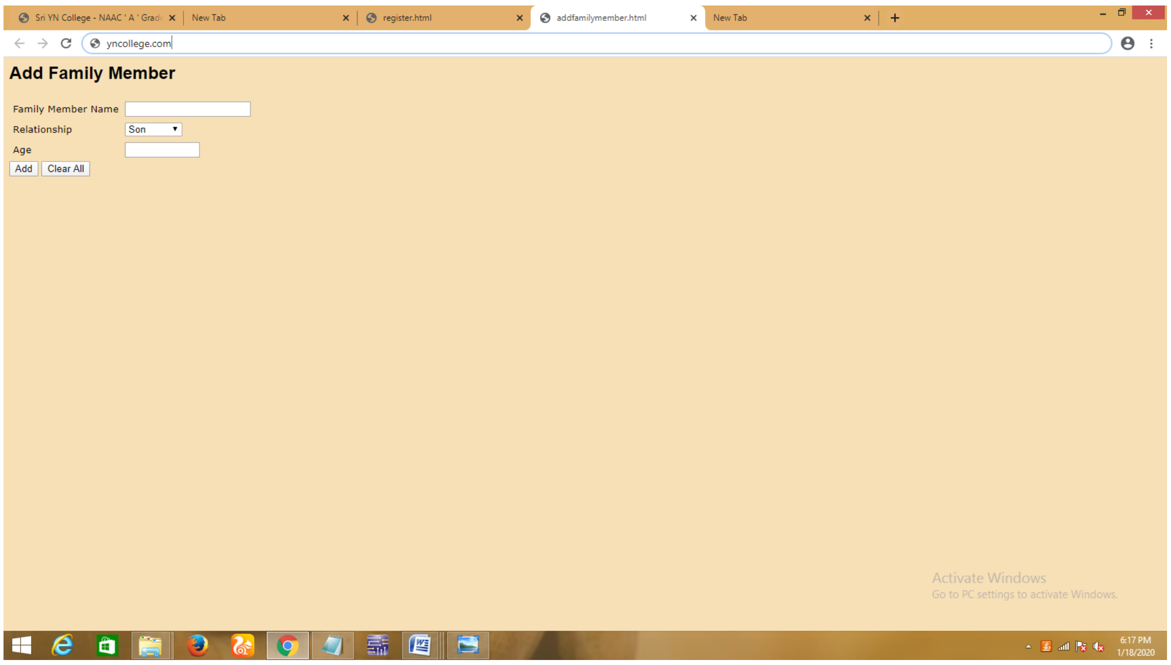

Registeration

Login Name	
Password	
Fullname	
Phone Number	
Email	
Address	
Occupation	
From Date (DD-MON-yy)	
Qualification	
Branch	MCA
Year Passed	

Change Clear All

Go Back

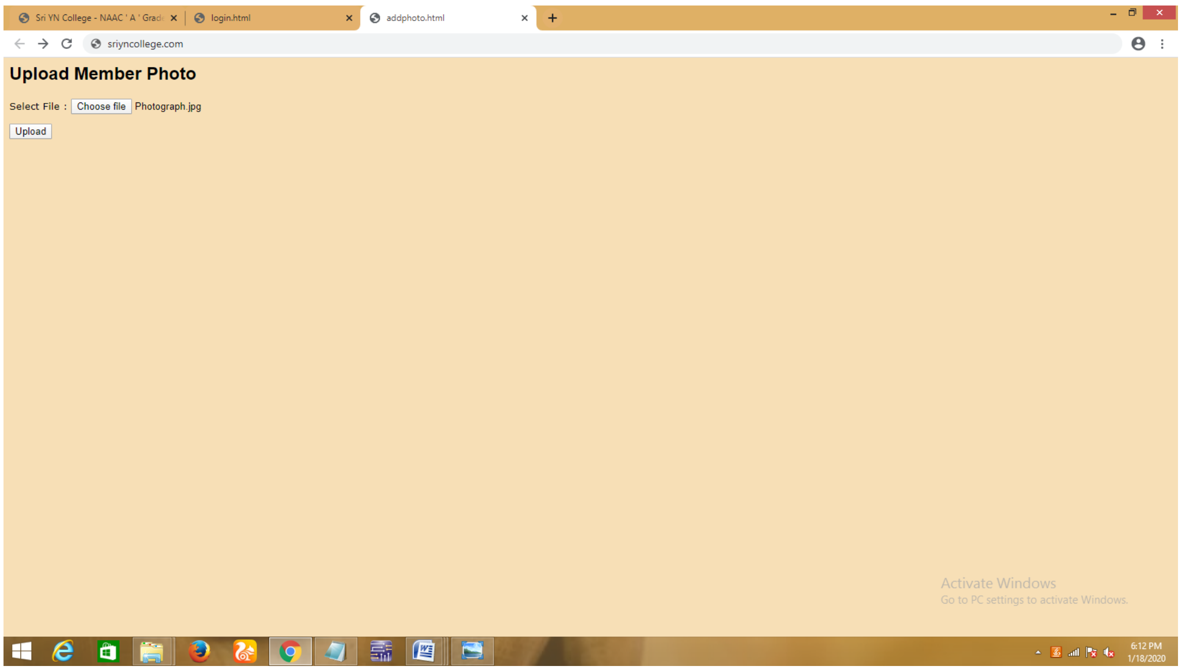
Fig 5(b): This is the registration page where students can register online


Add Family Member

Family Member Name	
Relationship	Son
Age	

Add | Clear All

Fig 5(c): This is the page to add family member details.


Upload Member Photo

Select File : Choose file Photograph.jpg

Upload

Fig 5(d): This is the page to upload member photo

6. SCOPE OF THE PROJECT

- ♣ The Students Website can be enhanced to include some other functionality like marks, attendance management.
- ♣ Talent management of students based on their performance evaluation can be added.
- ♣ Social networking can also be added wherein students can interact with each other.
- ♣ Online class functionality can be added.
- ♣ Can evolve as an online institution.
- ♣ Functionality of chat and messages can be added.
- ♣ Online exam functionality can be added.
- ♣ Online resume builder functionality can also be added.

7. CONTRIBUTION IN THE PROJECT

Students Website lead to a better organization structure since the information management of the students is well structured and also leads to better as well as efficient utilization of resources.

Student Website can be used by education institutes to maintain the records of students easily. Achieving this objective is difficult using a manual system as the information is scattered, can be redundant and collecting relevant information may be very time consuming. All these problems are solved using this project.

Our project Student Website was developed by all three of us. We, a team of three persons took a step by step approach in order to reach our goal. We applied the knowledge we gained during our training period at **Sri Y N College (A).** and developed this project **"STUDENTs WEBSITE"**.

8. CONCLUSION

We have developed this project for maintaining old students information. The system has been successfully implemented and the aim is achieved without any deviation. There is a lot of future scope of this project because of presenting information in an easy and intelligible manner. It can be used in Educational Institutes/Colleges. This project can also be developed or modified according to the rising needs and demand.

9. BIBLIOGRAPHY

- Programming JavaScript Applications book by Eric Elliott.
- JavaScript-The Definitive Guide by David Flanagan.
- HTML & CSS, and JavaScript & JQuery by Jon Duckett
- Pure JSP: Java Server Pagesby James Goodwill (Sams, 2000)
- JavaServer Pagesby Larne Pekowsky (Addison-Wesley, 2000)
- Instant JavaServer Pagesby Paul Tremblett (Osborne McGraw-Hill, 2000)
- Web Development with JavaServer Pagesby Duane K. Fields and Mark A. Kolb (Manning Publications, 2000)
- www.google.com.
- www.wikipedia.com
- www.w3schools.com
- Oracle8i: The Complete Reference (Book/CD-ROM Package)
- **Oracle9i: The Complete Reference** Written by best-selling Oracle Press authors Kevin Loney and George Koch