**Student Information System**

Software Requirement Specification

**Project Guide:** **Project Members:**

Srikanth Kavuri P. Susanth (N091130)

B. Meghana (N100648)

P. Maneesha (N100598) A.NazmaBegum(N100330) P. Annapurna (N091121)

**Table of Contents**

1. Introduction

1.1 Purpose

1.2 Scope

1.3 Definitions, Acronyms and Abbreviations

1.4 Technologies

1.5. References

1.6 Overview

1. Overall Description
   1. Product Perspective
   2. Product Functions
   3. User Characteristics
   4. Assumptions and Dependencies
2. System Requirements

3.1 Function Requirements

3.2 Non-Functional Requirements

4.UML Diagrams

4.1 Use-Case Diagram

4.1.1 Student

4.1.2 Admin

* 1. E-R Diagram
  2. DFD Diagram
     1. DFD-Level 0
     2. DFD- Level 1
  3. Sequence Diagram

5.Interfaces

5.1 User Interface

5.2 Hardware Interface

5.3 Software Interface

5.4 Communication Interface

6.Project Flow

7.Basic code of project

**1. Introduction**

1.1. Purpose:

This document describes the software requirements for a student information system. It is meant to be used to maintain and store the data regarding students and make the information useful.

1.2 Scope:

The function of the system is to facilitate student activities such as registration ,login, course registration, exam registration ,personal details and other information.

1.3 Definitions, Acronyms and Abbreviations:

JDBC: Java Database Connection

1.4 Technologies:

Eclipse , JDBC.

1.5 References:

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

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

1.6 Overview:

This system describes the general factors that involves. In order to be easily understandable, this part of SRS provides a background for the requirements. The detailed definitions can be found in Section 3 of the SRS.

**2. Overall Description**

2.1 Product Perspective:

Web Systems requires the development of a centralized system for facilitating all student administrative tasks such as registering for courses, managing grades data & transcript requests. Not only will the system be designed for students, there will be a feature to allow Records Administration to handle transcript requests. The system will also work with the existing University database

2.2 Product Functions:

The new web-based system should combine the functionality of student information controlling system. There should be a single log-on portal with which the user can access all possible services. The user would be directed to the appropriate page depending on whether the user is a student or records staff.

If the user is a student:

* They are directed to a Main page that has links to the other systems. (Course Registration, Course Time-Table, Student Accounts, and Records)
* The Course Registration page will provide course information and allow the user to register for course as well as view his time-table.
* The Course Time -Table page will provide general course information without any need of registration.
* The Student Accounts page will display a student’s personal information. It will also contain access to their account.
* The Records page will display information regarding the student’s academic progress and allow interaction with the Records Department.

If the user is an Administrator:

* The user is directed to a page that allows them to process Student Record related requests
* Student database access and control view

2.3 User Characteristics:

There are two groups of users using the system:

**Students:**  
 The students access the system to manage their courses and view their information. Due to the large size of this group, there is a wide range of technical ability. Ease of learning the system should be a priority for this user group.

**Admin:**  
 The admin access the system to view pending transcript requests and close them when they've been completed. As using the system is a specialized part of the admin occupation, efficiency of their workflow is the priority.

2.4 Assumptions and Dependencies:

We assume that everyone that uses the system has access to the portal at speeds of 56k or above.

The system is depended upon by many users therefore it should be able to deal with thousands of users logging onto the system at any one time.

**3. System Requirements**

* 1. Functional Requirements
* Secured Log in and Secured Database Management System. This was completely protected from the outside threats.
* Adequate searching mechanisms for easy and quick access to particular information.
  1. Non-Functional Requirement

3.2.1 Performance Requirements

* Speed and accuracy in retrieving data from database.
* Providing correct details.
* No redundancy in data storing.
  + 1. Easy to use interface:

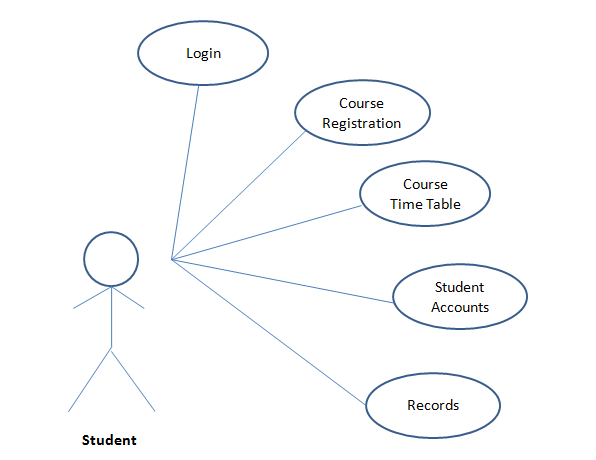
It is user friendly interface with little knowledge JSP Web portal

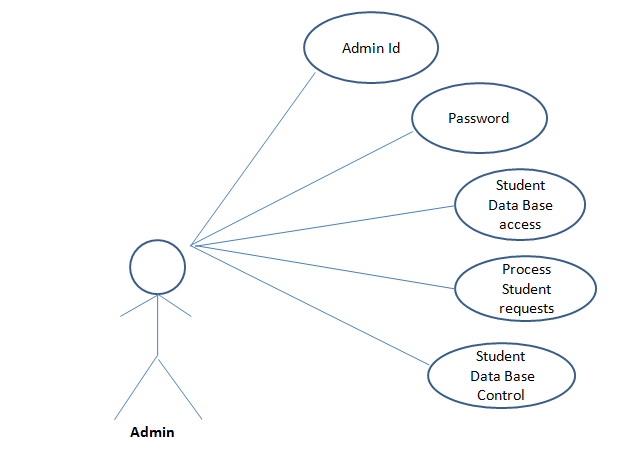
3.2.3. Efficiency:

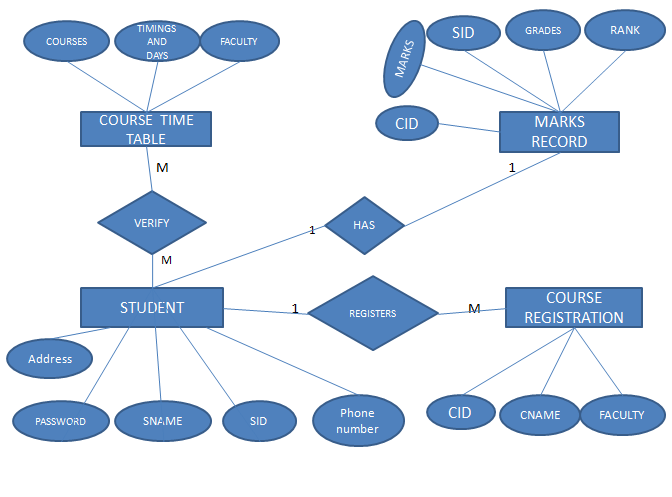
Portal is efficient to provide information of student details as much as efficient.

**4.UML Diagrams**

**4.1 Use-Case Diagram:**

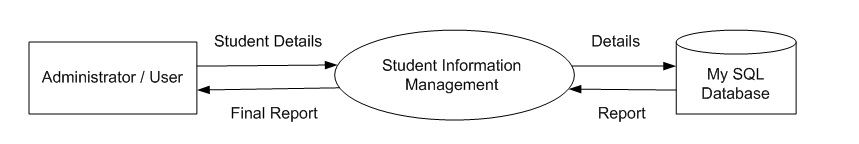
****

****

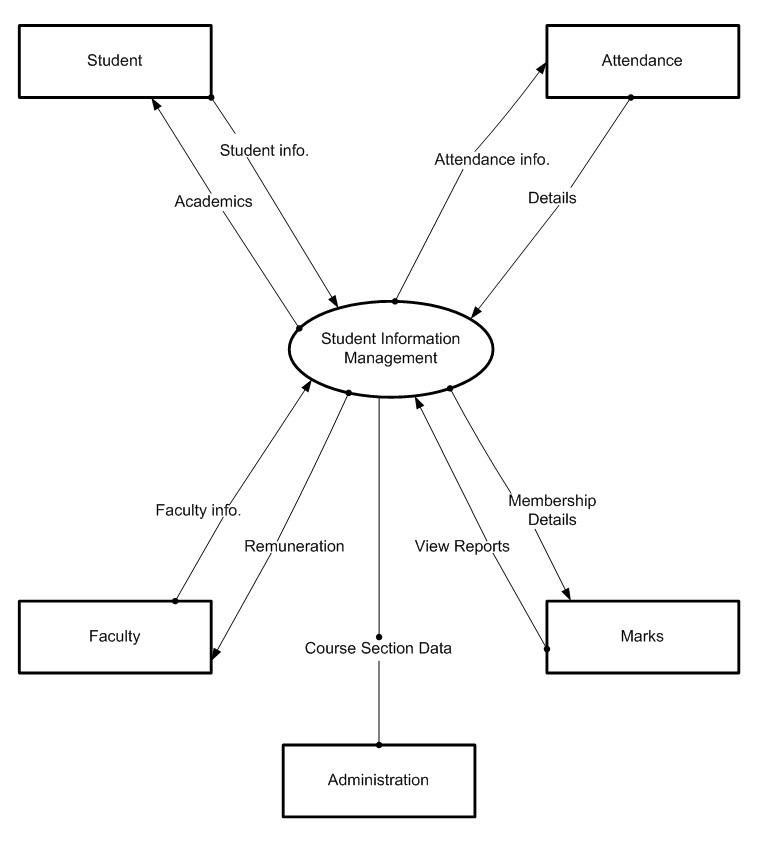
**4.2 E-R Diagram **

**4.3 DFD Diagram:**

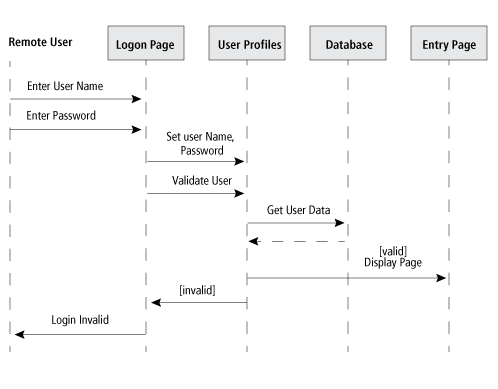
**DFD Level-0**

****

**DFD Level-1**

****

**4.4 Sequence Diagram :**

****

**5.Interfaces**

5.1 User Interface:

User Interface is an applet application which the user can interact and store their details.

5.2 Hardware Interfaces

Dual core processor  
2GB- Ram  
500 GB- Hard Disk  
Keyboard   
Mouse

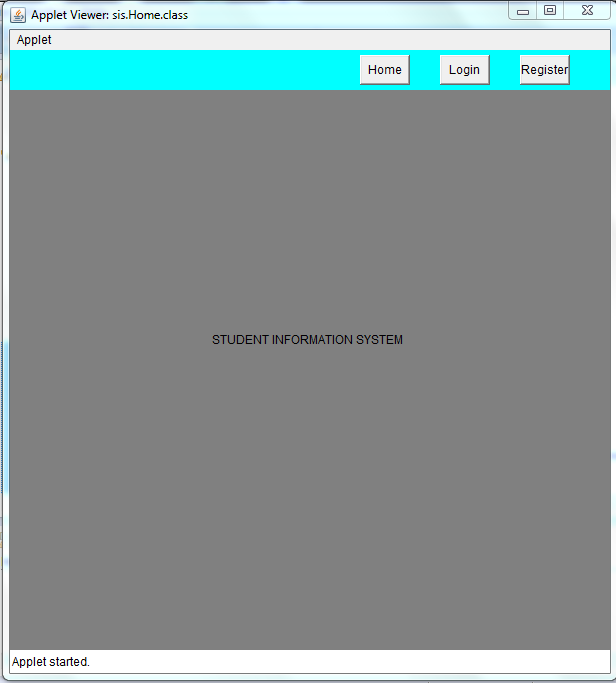
* 1. Software Interfaces

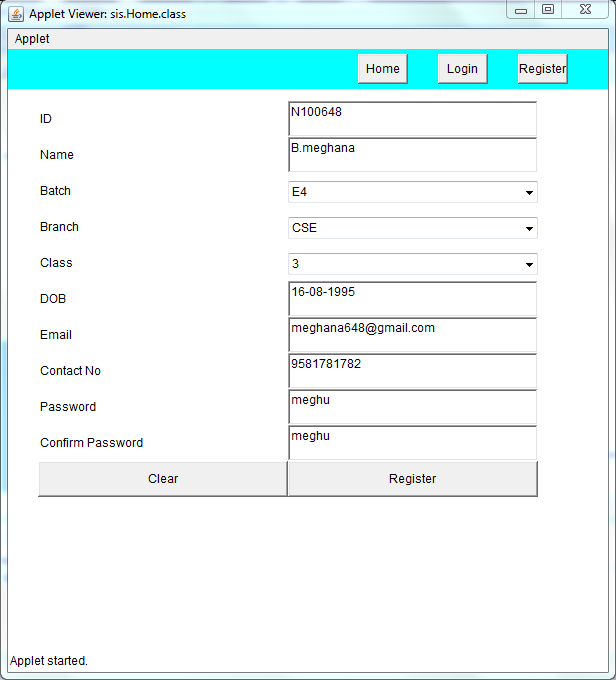
The system interface is with Windows O.S, JDBC, Eclipse (kepler), JRE-8, JDK-8, XAMPP Server.

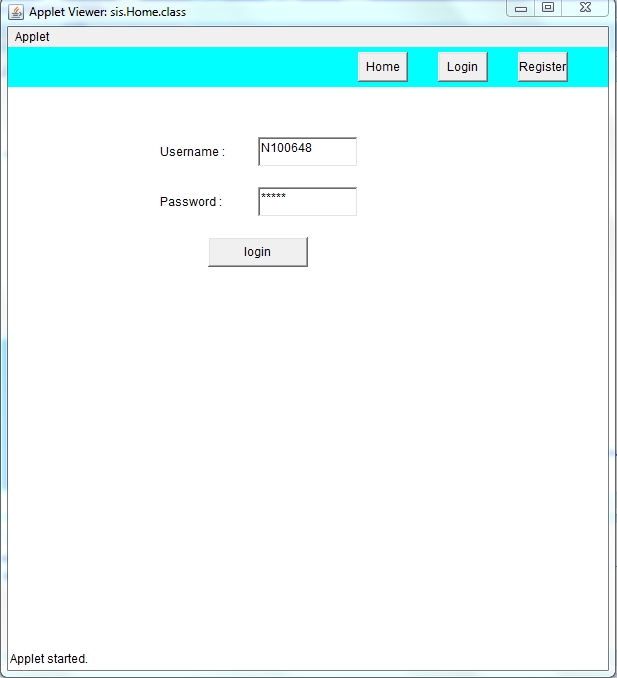
* 1. Communication Interfaces

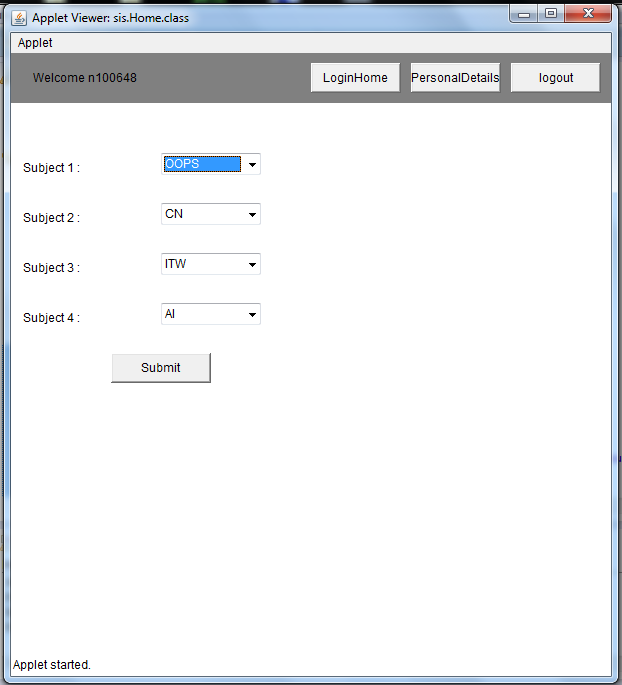
We need the central server to access the student information and registrations and login.

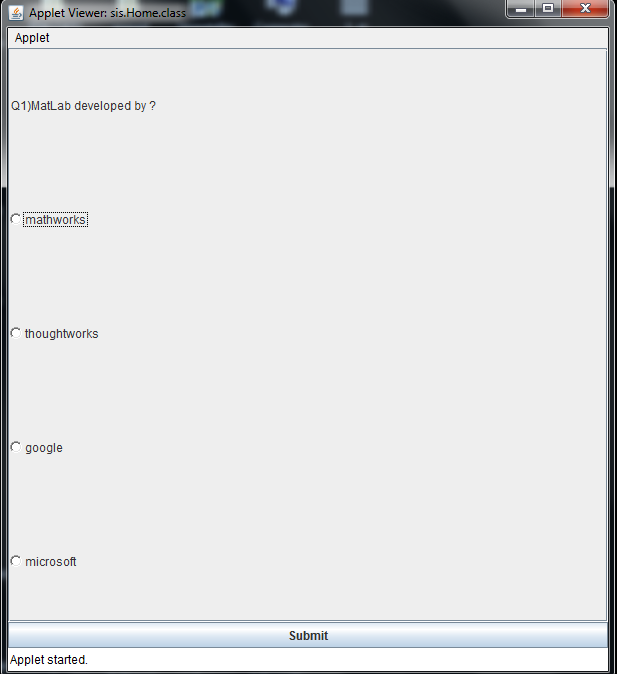
**6.Project Flow:**

****

****

****

****

****

**7.Project Basic Code:**

package sis;

import java.applet.\*;

import java.awt.\*;

import java.awt.event.\*;

public class Home extends Applet{

String userId = null;

Panel p1,p2;

Label title;

Button home,login,register;

public void init(){

setSize(600,600);

setLayout(null);

addPanel1();

addPanel2();

}

public Home getInstance(){

return this;

}

public void addPanel1(){

home = new Button("Home");

home.addActionListener(new ActionListener(){

@Override

public void actionPerformed(ActionEvent arg0) {

// TODO Auto-generated method stub

remove(p2);

p2 = new Panel();

p2.setLayout(null);

p2.add(title);

p2.setBounds(0,40,600,560);

add(p2);

revalidate();

}

});

login = new Button("Login");

login.addActionListener(new ActionListener(){

@Override

public void actionPerformed(ActionEvent arg0) {

// TODO Auto-generated method stub

remove(p2);

p2 = new LoginScreen(getInstance());

p2.setBounds(0,40,600,560);

add(p2);

revalidate();

}

});

register = new Button("Register");

register.addActionListener(new ActionListener(){

@Override

public void actionPerformed(ActionEvent arg0) {

// TODO Auto-generated method stub

remove(p2);

p2 = new Register(getInstance());

p2.setBounds(30,50,500,400);

add(p2);

revalidate();

}

});

home.setBounds(350, 5, 50, 30);

login.setBounds(430, 5, 50, 30);

register.setBounds(510, 5, 50, 30);

p1 = new Panel();

p1.setLayout(null);

p1.setBackground(Color.cyan);

p1.add(home);

p1.add(login);

p1.add(register);

p1.setBounds(0, 0, 600, 40);

add(p1);

}

public void addPanel2(){

title = new Label("STUDENT INFORMATION SYSTEM");

title.setBounds(200, 200, 1000, 100);

p2 = new Panel();

p2.setBackground(Color.gray);

p2.setLayout(null);

p2.add(title);

p2.setBounds(0,40,600,560);

add(p2);

}

}