MINOR PROJECT ON:

STUDENT INFORMATION SYSTEM

Minor Project Submitted in Partial Fulfillment of the Requirements for the degree of Bachelor of Computer Application

Prepared By

Name: MANISH PARUI
Roll No: 30901216085

Under the Guidance of

Mr. Dhruba Ray

TECHNO INDIA SALT LAKE
EM 4/1, Sector V
Kolkata-700091
2016-2019

Techno India, Salt Lake
Maulana Abul Kalam Azad University of Technology
(Formerly WBUT)

FACULTY OF BCA DEPARTMENT

Certificate of Recommendation

This is to certify that MANISH PARUI has completed his project work titled "Minor project on: STUDENT INFORMATION SYSTEM", under the direct supervision and guidance of Mr. DHRUBA RAY. We are satisfied with his work, which is being presented for the partial fulfillment of the degree of Bachelor of Computer Application (BCA), West Bengal University of Technology (WBUT), Kolkata– 700032.

Date: _____________ ________________________

DHRUBA RAY
Teacher in charge of project

____

MONALISA BANERJEE
HOD BCA Department

TECHNO INDIA SALTLAKE

Maulana Abul Kalam Azad University of Technology Formerly WBUT

FACULTY OF BCA DEPARTMENT

Certificate of Approval

The foregoing Minor project is hereby approved as a creditable study of Bachelor of Computer Application (BCA) and presented in a manner satisfactory to warrant its acceptance as a pre-requisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or any statement made, opinion expressed or conclusion therein but approve this Minor project only for the purpose for which it is submitted.

____

____

____

Signature of the Examiners

Final Examination for
Evaluation of the Project

Only in case the Minor project is approved.

PREFACE

Goal:

The main goal of the software is to help the faculty to access the detailed academic information of a student.

Organization:

Chapter 1:

SCOPE OF THE PROJECT -

In this chapter the introduction of the project is defined. An attempt has been made through this project to do all work ease & fast. It provides add, update, search & delete facilities to accomplish the desired objectives.

Chapter 2:

CONCEPT OF PROBLEM ANALYSIS -

In this chapter we have discussed about the COCOMO Model and calculated the LOC, KLOC and Development Time.

Chapter 3:

THEORITICAL BACKGROUND -

In this chapter the main background of the required softwares Java, MySql, Swing is described.

Chapter 4:

SOFTWARE REQUIREMENT SPECIFICATION -

1. Introduction- In this chapter we have discussed so far about the main requirements for the project. In this section we have discussed about the project details; its scope; definitions, acronyms and abbreviations used; the references from where we have got the details to use and the main responsibility of the developer in making this project.
2. General Descriptions- In this section the main overview of the project is given along with the user characteristics and general constraints and assumptions used.

3. Functional Requirements- This section deals with the descriptions of input and outputs in the project of what input will give which output to the user. It defines all the main functional requirements that have been used in making the project interface wise.
4. External Interface Requirements- This section gives us the details of how the interface will be used by the users. The error messages that will be shown are described here.
5. Design Constraints- This section gives us details of the hardware and software used to make the project.

Chapter 5:

DESIGN –

1. Data Design- This gives us the detailed diagram of Entity Relational Diagram of the project so made.
2. Interface Design- It holds the screenshot of all the interfaces.
3. Procedural Design- It gives us the details of the class names and its purpose, member function- name of function, argument list, return type, purpose and function algorithm.

Chapter 6:

CODING STANDARD FOLLOWED and ASSUMPTIONS –

This gives the details of the standard format of coding and assumptions that are followed.

Chapter 7:

TESTING –

This chapter gives us the detailed description of all the validation testings done. It gives us every testing details module wise. It specifies when we give any input what output is shown to us and whether the result passes the requirements.

Chapter 8:

FUTURE SCOPE OF THE PROJECT –

This topic gives us details about all the scopes in which we can work in future using this project.

Chapter 9:

CONCLUSION –

This topic gives us the brief description of the project’s documentation.

Chapter 10:

REFERENCE AND BIBLIOGRAPHY –

This gives us the reference and links from where we have used TO COMPLETE THE PROJECT OR THE DOCUMENTATION PART.

Chapter 11:

APPENDIX –

This section carries the coding of different module of the student information system application.

ACKNOWLEDGEMENT

It gives me a great sense of pleasure to present the report of BCA Project undertaken during BCA Final Year. We take this opportunity to express our sincere gratitude to all those who helped us in various capacities in undertaking this project and devising the report. We are privileged to express our sense of gratitude to our respected teacher Mr. Dhruba Roy whose unparalleled knowledge, moral fiber and judgement along with his know-how was an immense support in completing the project. We are also grateful to Mrs. Monalisa Banerjee the Head of our Department, Bachelor of Computer Application for the brainwave and encouragement given. We take this opportunity also to thank our friends and contemporaries for their co-operations and compliance.

MANISH PARUI

INDEX

1. SCOPE OF THE PROJECT

The present project has been developed to meet the aspirations indicated in the modern age. An attempt has been made through this project to do all work ease & fast. It provides current add, update, search & delete all facilities to accomplish the desired objectives. The facility includes in this project and the suggested activities have been organized to impart knowledge & develop skill & attitude in the College official works of student's details.

A student information system (SIS), school administration software or student administration system is a management information system for education establishments to manage student data. SIS provide capabilities for registering students in courses; documenting grading transcripts, results of student tests and other assessment scores; building student schedules; tracking student attendance; and managing many other student-related data needs in a college.

The application is reduced as much as possible to avoid error while entering the data. It also provides error message while entering invalid data. No formal knowledge is needed for the user to use this system. Thus by this all it proves that it is user-friendly. Student information system is a direct interface between college & student or their parents. Here Students or their Guardians can check the performance of the student using the provided login id and password to them. We can check session Marks obtained by student in the current session.

Every institution whether big or small has challenges to overcome and managing the information of student profile, general information, courses, exams, marks. Because every student information system has different student details. This is designed to assist in strategic planning and will help to ensure that the institution is equipped with the right level of information and details for the future goals. This has been developed to override the problems prevailing in the practicing manual system.

It may help collecting perfect management in details. In a very short time, the collection will be obvious, simple and sensible. It will help a person to know the management of passed year perfectly and vividly. It also helps in current all works relative to a school or college. It will be also reduced the cost of collecting the management & collection procedure will go on smoothly.

2. CONCEPT AND PROBLEM ANALYSIS

2.1 Cost Analysis using COCOMO Model:

We consider following cost drivers:

Cost drivers	Ratting
Required software reliability	1
Complexity of the product	1
Required turnabout time	1
Analyst capability	1
Applications experience	1
Programming language experience	1
Use of software tools	1

Total LOC = 1500
Then, KLOC = 1.5

EAF = 1*1*1*1*1*1*1 = 1

Development Effort = $3.2 * (KLOC)^{1.05} * EAF$
= $3.2 * 1.5^{1.05} * 1$
= 4.9 PM (approx.)

Development time = $2.5 * (Effort)^{0.38}$
= 4.6 months (approx.).
= 4 months 18 days (approx.).

Note: we consider our project as organic type. And we follow Intermediate COCOMO estimation method to calculate effort and development time.

3. THEORETICAL BACKGROUND

Java

Java is a general-purpose computer-programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture. As of now, Java is one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers. Java was originally developed by James Gosling at Sun Microsystems and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.

The original and reference implementation Java compilers, virtual machines, and class libraries were originally released by Sun under proprietary licenses. As of May 2007, in compliance with the specifications of the Java Community Process, Sun relicensed most of its Java technologies under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java (bytecode compiler), GNU Classpath (standard libraries), and IcedTea-Web (browser plugin for applets).

The latest version is Java 11, released on September 25, 2018, which follows Java 10 after only six months in line with the new release schedule. Java 8 is still supported but there will be no more security updates for Java 9. Versions earlier than Java 8 are supported by companies on a commercial basis.

Swing (Java)

Swing is a GUI widget toolkit for Java. It is part of Oracle's Java Foundation Classes (JFC) – an API for providing a graphical user interface (GUI) for Java programs.

Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit (AWT). Swing provides a look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

Unlike AWT components, Swing components are not implemented by platform-specific code. Instead, they are written entirely in Java and therefore are platform-independent. The term "lightweight" is used to describe such an element.

MySQL

MySQL is an open source relational database management system (RDBMS). Its name is a combination of "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation. For proprietary use, several paid editions are available, and offer additional functionality.

MySQL is written in C and C++. Its SQL parser is written in yacc, but it uses a home-brewed lexical analyzer. MySQL works on many system platforms, including AIX, BSDi, FreeBSD, HP-UX, eComStation, i5/OS, IRIX, Linux, macOS, Microsoft Windows, NetBSD, Novell NetWare, OpenBSD, OpenSolaris, OS/2 Warp, QNX, Oracle Solaris, Symbian, SunOS, SCO OpenServer, SCO UnixWare, Sanos and Tru64. A port of MySQL to OpenVMS also exists.

The MySQL server software itself and the client libraries use dual-licensing distribution. They are offered under GPL version 2, beginning from 28 June 2000 (which in 2009 has been extended with a FLOSS License Exception) or to use a proprietary license.

Support can be obtained from the official manual. Free support additionally is available in different IRC channels and forums. Oracle offers paid support via its MySQL Enterprise products. They differ in the scope of services and in price. Additionally, a number of third party organisations exist to provide support and services, including MariaDB and Percona.

MySQL has received positive reviews, and reviewers noticed it "performs extremely well in the average case" and that the "developer interfaces are there, and the documentation (not to mention feedback in the real world via Web sites and the like) is very, very good". It has also been tested to be a "fast, stable and true multi-user, multi-threaded sql database server".

4. SOFTWARE REQUIREMENTS SPECIFICATIONS

4.1 INTRODUCTION

A student information system (SIS), school administration software or student administration system is a management information system for education establishments to manage student data. SIS provide capabilities for registering students in courses; documenting grading transcripts, results of student tests and other assessment scores and managing many other student-related data needs in a college.
The application is reduced as much as possible to avoid error while entering the data. It also provides error message while entering invalid data. No formal knowledge is needed for the user to use this system. Thus by this all it proves that it is user-friendly. Student information system is a direct interface between college & student or their parents. Here Students or their Guardians can check the performance of the student using the provided login id and password to them. We can check session Marks obtained by student in the current session.

Every institution whether big or small has challenges to overcome and managing the information of student profile, general information, courses, exams, marks. Because every student information system has different student details. This is designed to assist in strategic planning and will help to ensure that the institution is equipped with the right level of information and details for the future goals. This has been developed to override the problems prevailing in the practicing manual system.

4.1.1 Purpose

The main purpose of the project on student information system is to manage the details of student logins, profiles, general information, courses, exams, marks. The project is totally built at administrative end so; the administrator gets more importance than student. The purpose of the project is to build and application program to reduce the manual work for managing the student profiles, general information and scorecard. It also provides features to add, update, delete and search a particular information based on the student id provided.

4.1.2 Scope

The present project has been developed to meet the aspirations indicated in the modern age. An attempt has been made through this project to do all work ease & fast. It provides add, update, search & delete facilities to accomplish the desired objectives. The facility includes in this project and the suggested activities have been organized to impart knowledge & develop skill & attitude in the College official works of student's details.

It may help collecting perfect management in details. In a very short time, the collection will be obvious, simple and sensible. It will help a person to know the management of passed year perfectly and vividly. It also helps in current all works relative to a school or college. It will be also reduced the cost of collecting the management & collection procedure will go on smoothly.

4.1.3 Definitions, Acronyms, Abbreviations

- SIS: Student Information System
- SERVER: refers to the Host machine
- USER: Refers to the user of SIS
- SQL: Structure Query Language; used to retrieve information, database
- BOOLEAN: A true/false notation
- UNIQUE key: Use to deferential entries in a data base.
- LAYER: Represent a section of the project
- DATA BASE: The section of the assignment referring to where all data is recorded.

4.1.4 References

- https://en.wikipedia.org/wiki/Student_information_system
- https://projectabstracts.com/1664/student-information-system.html
- https://www.scribd.com/document/334841577/Synopsis-of-Student-Information-System
- http://www.microtek.ac.in/UploadedFiles/Downloads/SYNOPSIS%20Example.pdf
- https://employment.blurtit.com/3206502/what-are-the-scope-and-limitation-in-student-information-system
- http://www.authorstream.com/Presentation/shashijais789-1244141-student-information-system/

4.1.5 Developer's Responsibility Overview

The roles and responsibilities of the developer of this project is very vast. Here are common examples that are to be followed:

- Designing, implementing, and maintaining the application as it is required for mission-critical system.
- Delivering high availability and performance
- Contributing in all phases of the development lifecycle
- Writing well-designed, efficient, and testable code
- Conducting software analysis, programming, testing, and debugging
- Managing Java application development
- Ensuring designs comply with specifications
- Preparing and producing releases of software components
- Transforming requirements into stipulations
- Maintaining user friendly operations with proper direction of use.
- Support continuous improvement
  - Investigating alternatives and technologies
  - Presenting for architectural review

4.2 General Descriptions

4.2.1 Product Overview

Product function will include the following functional areas:

- Admin log into the application with user name and password.
- If the administrator enters invalid name or password, then they will not be allowed to do any operations.
- In case of normal user, they can also log into the application with login id and password.
- If the user enters invalid name or password, then they will not be able to update their information.
- User can update student information in student details and admin can update both in student details and user details.
- The log in id must be unique.

4.2.2 User Characteristics

- In case of new user, they must go through the sign up button to create their id. They need to give their information like: **-** login id, password, first name, last name, sex, birth date, contact no., email id, address and department.
- In the student mode it is needed to login with the provided login id and password to add, delete or update any information like Student Id, First Name, Sex, Last Name, Birth Date, Contact no., Email Id, Address, University, Department, Reg. No., College, Section, Roll No. in the particular sections.
- If the user wants to delete any details from the form they will need to go through an authentication.
- Admin and user both can access the score card.

4.2.3 General Constraints and Assumptions

We need to submit this project on or before 26 November, 2018.
If we could get more time, then we would have definitely improved our software by adding more facilities.

4.3 Functional Requirements

4.3.1 General Description of Input and Output

Mainly the software is being used by the faculty.

In the administrator mode- When we create new user account profile by using provided login id and password and giving the details we save them. After that we need to refresh the database and then when we select any particular student details, the details provided is showed in the column provided beside.

In the student mode- When we give any input of information of student in the database and save it. After that we need to refresh the database and then when we select any particular student details, the details provided is showed in the column provided beside.
When we select a student's name from the provided database we can also go through his/her score card provided.

4.3.2 Functional Requirements

The main functional requirment is to maintain student information. There are two different modes for using this software:

Admin mode:

- Make sure that admin and user can view the details of all the students.
- Admin can delete the record of the student.
- Make sure that admin can accept /decline user registration request.
- The admin can add, update, search and delete the user information records into database.

User mode:

- Make sure that they can view his/her information.
- They can view and search the details of student information.
- They can add, update and delete the provided details.
- They can get the detailed view of the score card of the students.

4.4 External Interface Requirements

4.4.1 User Interface

Interface 1: - In the login interface the admin or any other user will provide the login id and password to login.

Interface 2: - In case a new user wants to login then they will go through the sign up button and provide id and password and the following interface will be opened.

Interface 3: - When we click on the “Load Database” button we get the saved database. After that if we want to see the details of any student we will need to select a student’s name the form beside gives the details of the selected. If we want to provide new details, we will need to click on “New” button and then provide the needed details according to the given instructions. After giving the details the “New” button will change to “Update button”. By clicking the update button the database gets saved. After this if we want to see the details in the form then we will click the “Refresh Database” button and the corresponding details get viewed. If we want to edit any details of a particular we will click on the “Edit” button and change as per need then the same button will get changed to “Update” and the database will get saved. If we want to delete any student’s record, we will click on the “Delete” button and then we will face an authentication in which we will have to provide the password. We can fetch any student’s information by providing the student id in the search button. When we click on the “Logout” button we will be logged out from the current logged in user.

Interface 4: - To fetch the score card of a student we can go through the “Scorecard” button and get the details. The scorecard interface will get opened.

Interface 5: - Only admin can give the details of the users by switching in the administrator mode by logging in with their own login id and password and then clicking on the “Administrator mode”. And the following interface will be opened where we will input the details of the user. Here also the work will be done same as that of student details interface.

Interface 6: - When we click on the “Change User” button the Login page will be opened but until we login to a new user we will not be logged out of the present.

Interface 7: - If in the student mode any user goes to the administrator mode and want to give any entry of the user the buttons like new, edit and delete will be disabled.

4.4.2 Error Messages

Student Information System - Login Screen

1. If we enter wrong id or password at the time of log in, then there will be shown an error message as: “Wrong Id or Password”.

Student Information System - Sign Up

1. In the Signup form if we enter wrong id then the error message shown will be as: “Unable to create password please contact admin”.

Student Information System - Student Mode

1. If we don’t provide any of the mandatory fields or void any rule (e.g. provide more than 15-character long id) in the student details during creation of any new profile, then the profile will not be created and an error message will be shown as: “Unable to create profile”.
2. If we void any rule during the updation of existing student profile, then the profile will not be updated and an error message will be shown as: “Unable to update”.
3. If we want to delete any information there comes an authentication form where we need to give the respective password if we provide wrong password, then there comes an error message that is “Wrong Password”

Student Information System - Administrator Mode

1. If we don’t provide any of the mandatory fields or void any rule (e.g. provide more than 20-character long login id) in the user details during creation of any new profile, then the profile will not be created and an error message will be shown as: “Unable to create profile”.
2. If we void any rule during the updation of existing user profile, then the profile will not be updated and an error message will be shown as: “Unable to update”.
3. If we want to delete any information there comes an authentication form where we need to give the respective password if we provide wrong password, then there comes an error message that is “Wrong Password”

Student Information System - Change User

1. In the change user form, if we provide wrong password then there will be shown an error message that is “Wrong Password”.

4.5 Design Constraints

4.5.1 Hardware Requirements

- CPU 1.5GHz Dual core or higher
- RAM 4GB or higher
- Disk 10GB or higher
- Display 1280*768
- Graphics Accelerators nVidia or ATI with Support of OpenGL 1.5 or higher

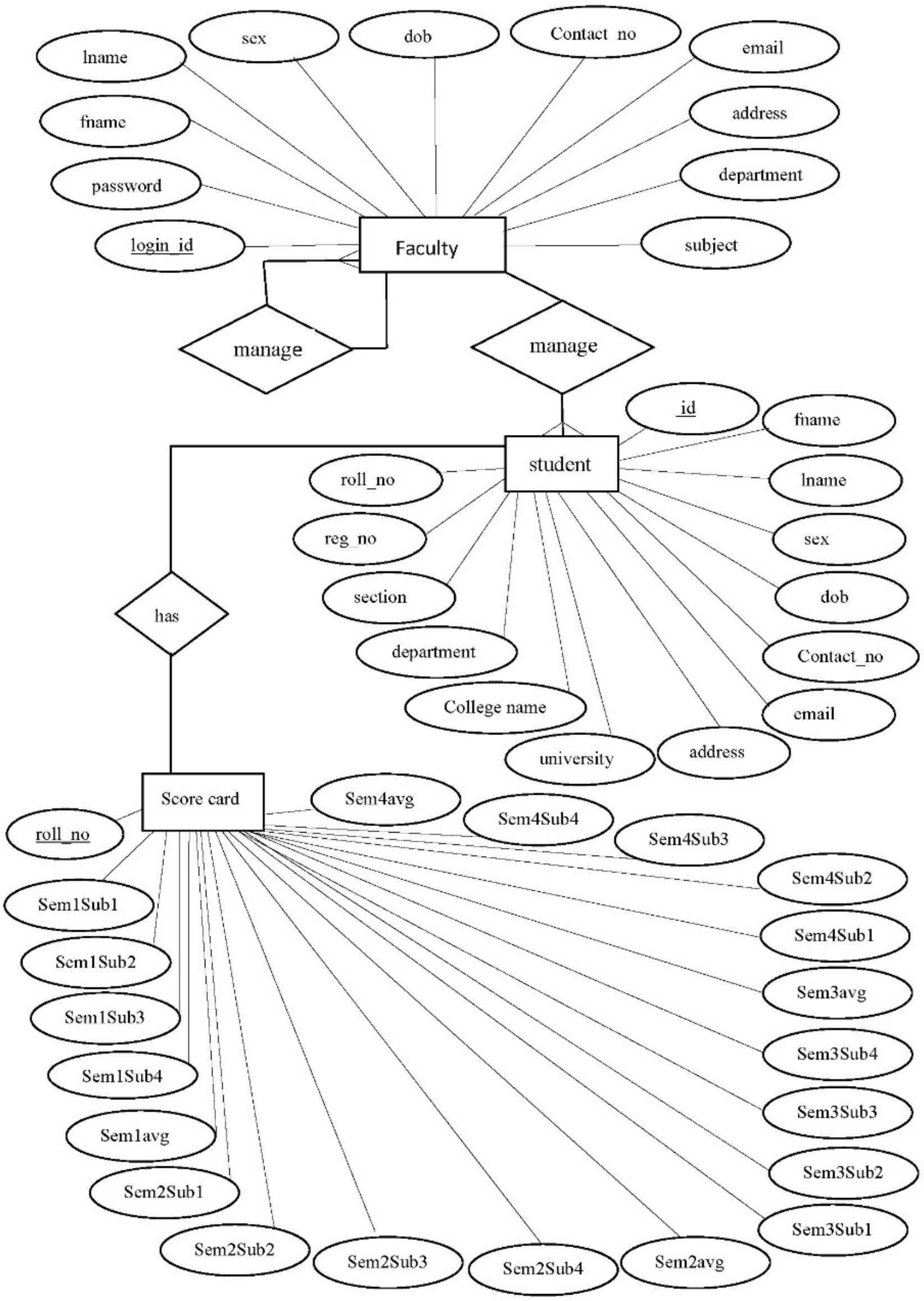
4.5.2 Software Requirements

- JDK1.8 or higher
- MySQL

5. DESIGN

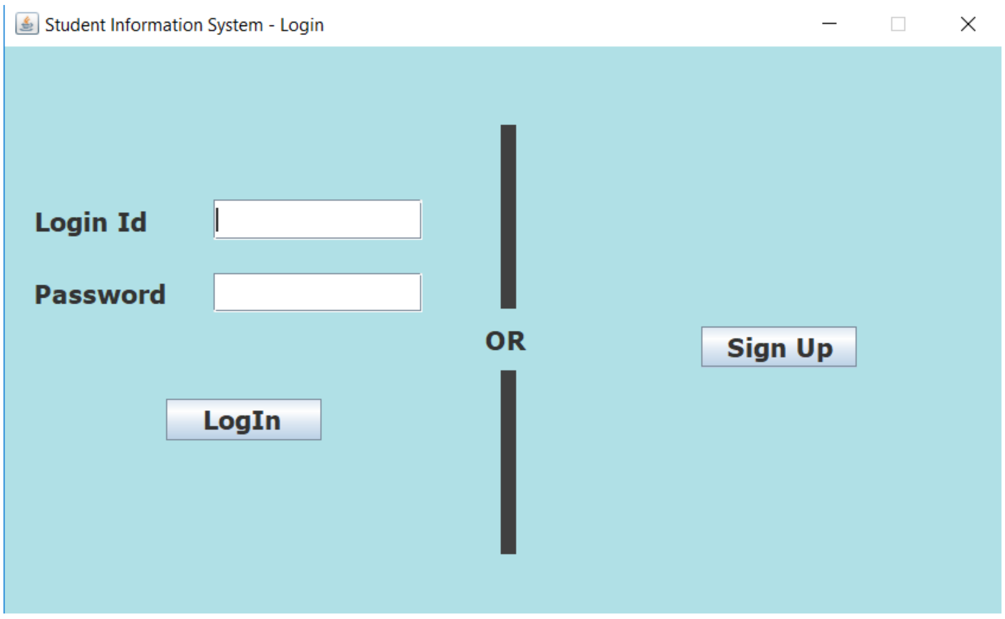
5.1 Data Design

5.1.1 Entity Relationship Diagram

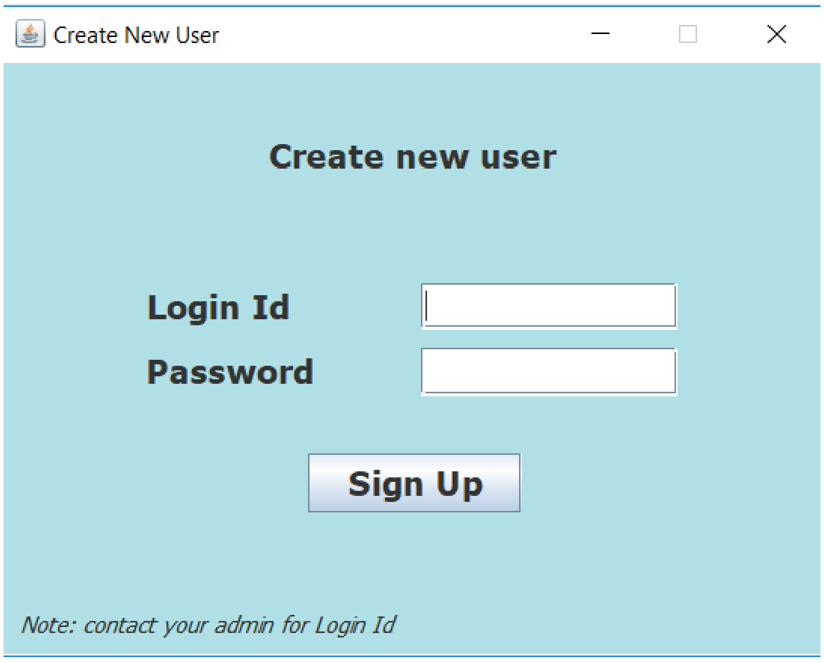


5.2 Interface Design

Interface 1-

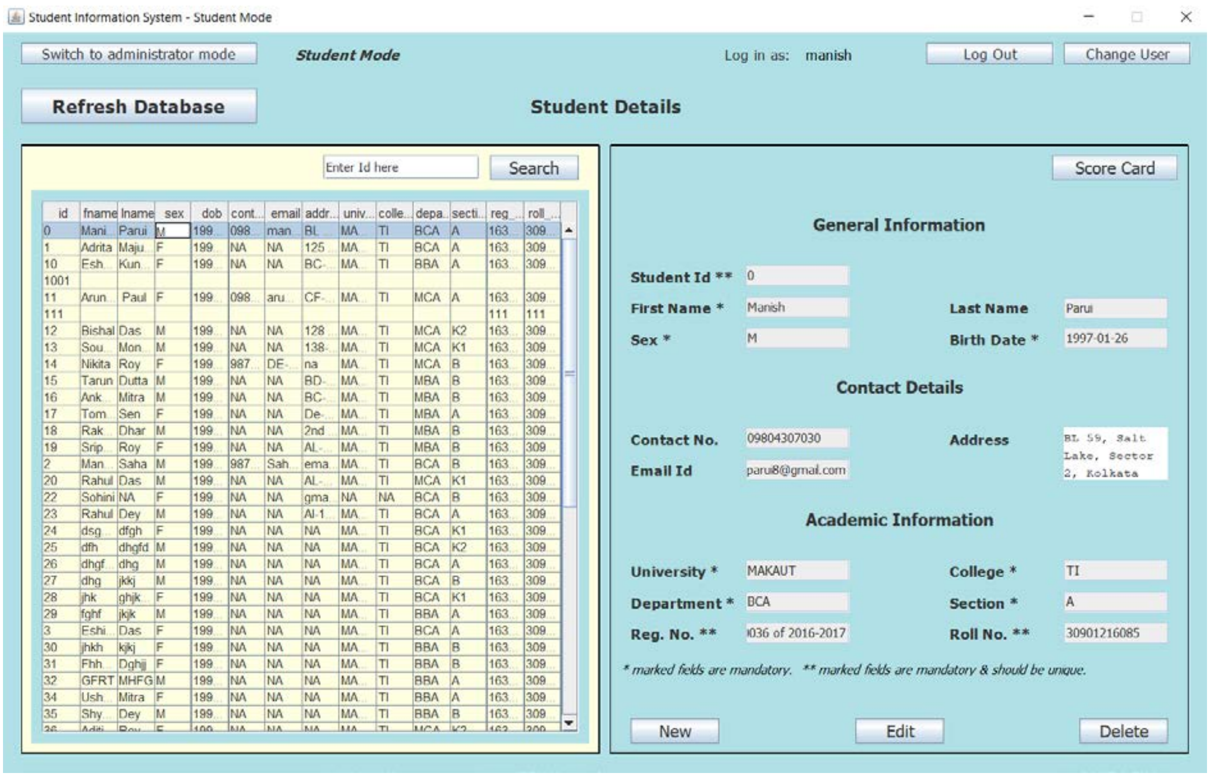


Interface 2-



Note: contact your admin for Login Id

Interface 3-



Log in as: manish

Student Details

id	fname	lname	sex	dob	cont…	email	addr…	univ…	colle…	depa…	secti…	reg_…	roll_…
0	Mani…	Parui	M	199…	098…	man…	BL…	MA…	TI	BCA	A	163…	309…
1	Adrita	Maju…	F	199…	NA	NA	125…	MA…	TI	BCA	A	163…	309…
10	Esh…	Kun…	F	199…	NA	NA	BC-…	MA…	TI	BBA	A	163…	309…
1001													
11	Arun…	Paul	F	199…	098…	aru…	CF-…	MA…	TI	MCA	A	163…	309…
111												111	111
12	Bishal	Das	M	199…	NA	NA	128…	MA…	TI	MCA	K2	163…	309…
13	Sou…	Mon…	M	199…	NA	NA	138-…	MA…	TI	MCA	K1	163…	309…
14	Nikita	Roy	F	199…	987…	DE-…	na	MA…	TI	MCA	B	163…	309…
15	Tarun	Dutta	M	199…	NA	NA	BD-…	MA…	TI	MBA	B	163…	309…
16	Ank…	Mitra	M	199…	NA	NA	BC-…	MA…	TI	MBA	B	163…	309…
17	Tom…	Sen	F	199…	NA	NA	De-…	MA…	TI	MBA	A	163…	309…
18	Rak…	Dhar	M	199…	NA	NA	2nd…	MA…	TI	MBA	B	163…	309…
19	Srip…	Roy	F	199…	NA	NA	AL-…	MA…	TI	MBA	B	163…	309…
2	Man…	Saha	M	199…	987…	Sah…	ema…	MA…	TI	BCA	B	163…	309…
20	Rahul	Das	M	199…	NA	NA	AL-…	MA…	TI	MCA	K1	163…	309…
22	Sohini	NA	F	199…	NA	NA	gma…	NA	NA	BCA	B	163…	309…
23	Rahul	Dey	M	199…	NA	NA	AI-1…	MA…	TI	BCA	A	163…	309…
24	dsg…	dfgh	F	199…	NA	NA	NA	MA…	TI	BCA	K1	163…	309…
25	dfh	dhgfd	M	199…	NA	NA	NA	MA…	TI	BCA	K2	163…	309…
26	dhgf…	dhg	M	199…	NA	NA	NA	MA…	TI	BCA	A	163…	309…
27	dhg	jkkj	M	199…	NA	NA	NA	MA…	TI	BCA	B	163…	309…
28	jhk	ghjk…	F	199…	NA	NA	NA	MA…	TI	BCA	K1	163…	309…
29	fghf	jkjk	M	199…	NA	NA	NA	MA…	TI	BBA	A	163…	309…
3	Eshi…	Das	F	199…	NA	NA	NA	MA…	TI	BCA	A	163…	309…
30	jhkh	kjkj	F	199…	NA	NA	NA	MA…	TI	BBA	B	163…	309…
31	Fhh…	Dghjj	F	199…	NA	NA	NA	MA…	TI	BBA	B	163…	309…
32	GFRT	MHFG	M	199…	NA	NA	NA	MA…	TI	BBA	A	163…	309…
34	Ush…	Mitra	F	199…	NA	NA	NA	MA…	TI	BBA	A	163…	309…
35	Shy…	Dey	M	199…	NA	NA	NA	MA…	TI	BBA	B	163…	309…
36	Aditi	Roy	F	199…	NA	NA	NA	MA…	TI	MCA	K2	163…	309…

General Information

Student Id **	0		
First Name *	Manish	Last Name	Parui
Sex *	M	Birth Date *	1997-01-26

Contact Details

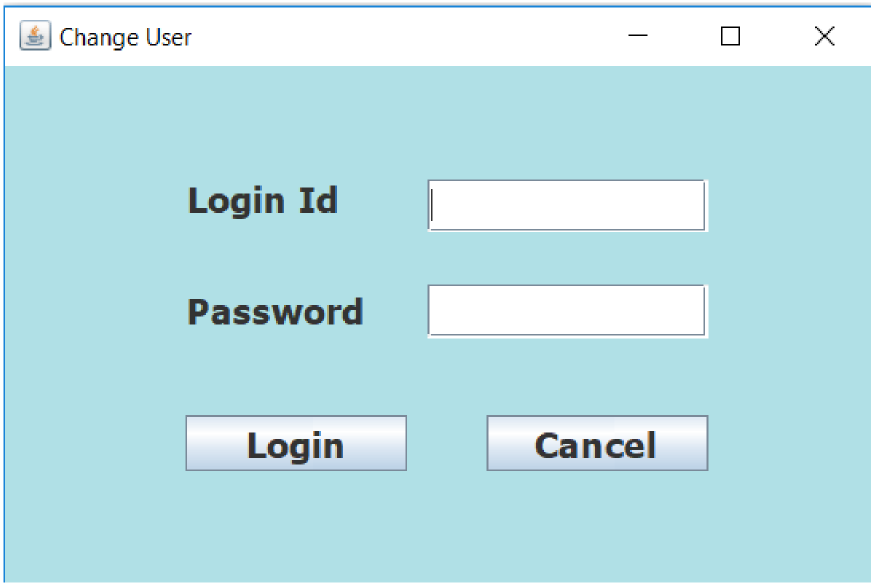
Contact No.	09804307030	Address	BL 59, Salt Lake, Sector 2, Kolkata
Email Id	parui8@gmail.com		

Academic Information

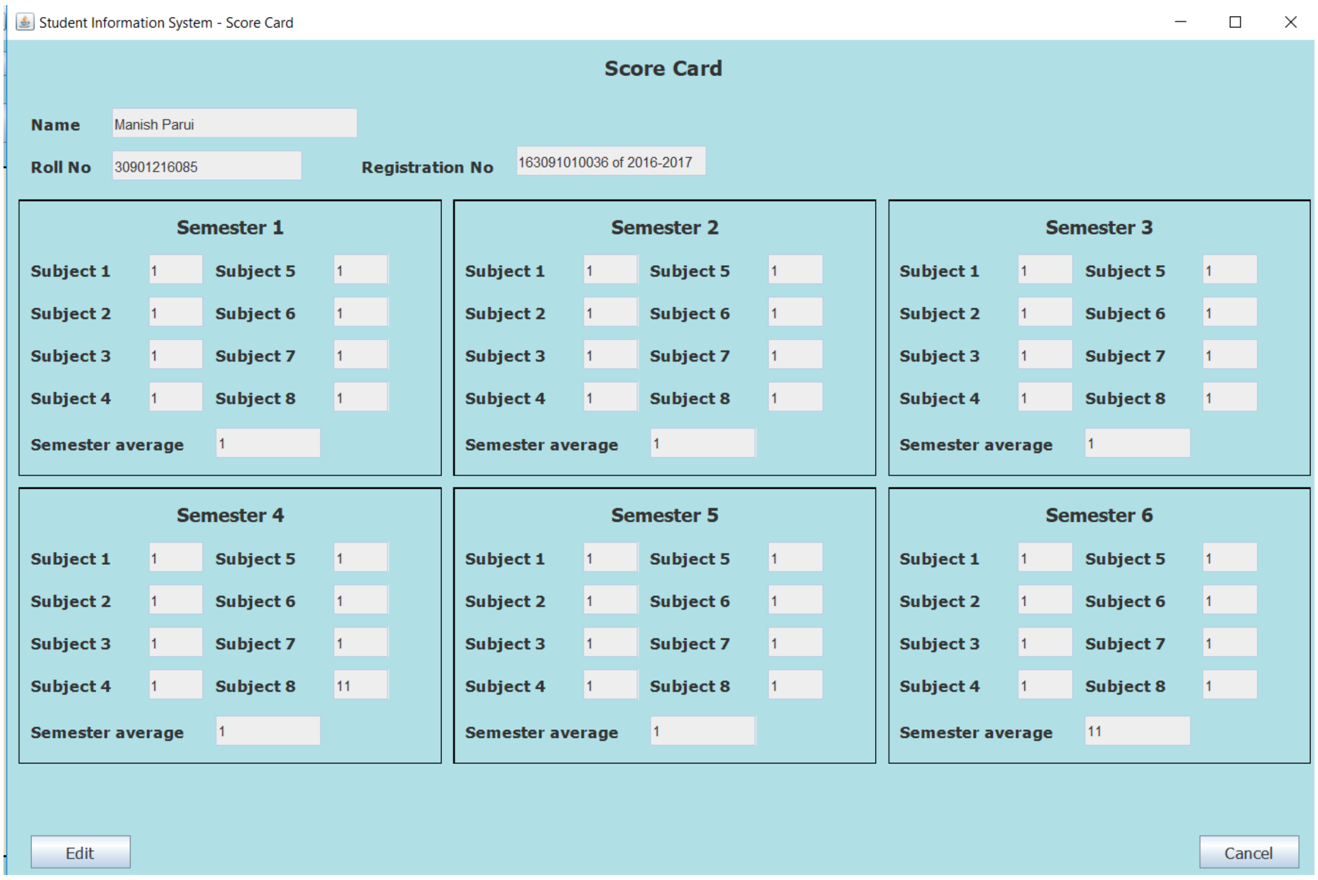
University *	MAKAUT	College *	TI
Department *	BCA	Section *	A
Reg. No. **	036 of 2016-2017	Roll No. **	30901216085

** marked fields are mandatory. ** marked fields are mandatory & should be unique.*

Interface 4-



Interface 5-



Score Card

Name: Manish Parui

Roll No: 30901216085

Registration No: 163091010036 of 2016-2017

Semester 1

Subject 1	1	Subject 5	1
Subject 2	1	Subject 6	1
Subject 3	1	Subject 7	1
Subject 4	1	Subject 8	1
Semester average	1		

Semester 2

Subject 1	1	Subject 5	1
Subject 2	1	Subject 6	1
Subject 3	1	Subject 7	1
Subject 4	1	Subject 8	1
Semester average	1		

Semester 3

Subject 1	1	Subject 5	1
Subject 2	1	Subject 6	1
Subject 3	1	Subject 7	1
Subject 4	1	Subject 8	1
Semester average	1		

Semester 4

Subject 1	1	Subject 5	1
Subject 2	1	Subject 6	1
Subject 3	1	Subject 7	1
Subject 4	1	Subject 8	11
Semester average	1		

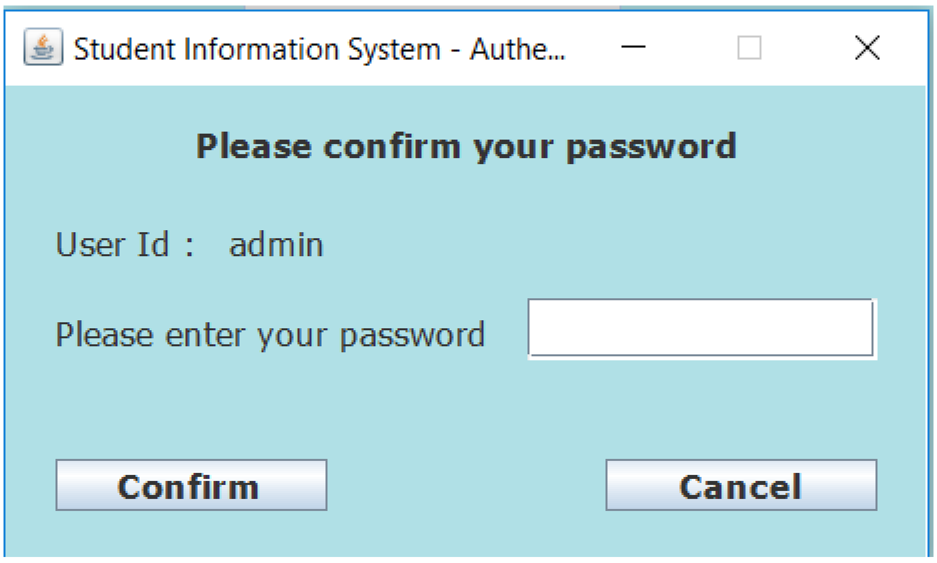
Semester 5

Subject 1	1	Subject 5	1
Subject 2	1	Subject 6	1
Subject 3	1	Subject 7	1
Subject 4	1	Subject 8	1
Semester average	1		

Semester 6

Subject 1	1	Subject 5	1
Subject 2	1	Subject 6	1
Subject 3	1	Subject 7	1
Subject 4	1	Subject 8	1
Semester average	11		

Interface 6-



Please confirm your password

User Id : admin

Please enter your password

5.3 Procedural Design

Class name: Loginfrm
Purpose: Let user log in to application.
Member functions:

1. Name of function: main
   Argument list: String[] args
   Return type: void
   Purpose: Launch the application
   Algorithm: NA

2. Name of function: actionPerformed
   Argument list: ActionEvent e
   Return type: void
   Purpose: Enable user to log in.
   Algorithm: NA

Class name: Createusrfrm
Purpose: Create new user profile.
Member functions:

1. Name of function: main
   Argument list: String[] args
   Return type: void
   Purpose: launch the application
   Algorithm: NA

2. Name of function: actionPerformed
   Argument list: ActionEvent e
   Return type: void
   Purpose: Enable user to sign up.
   Algorithm: NA

Class name: Mainfrm
Purpose: To show, search, insert, update or delete student details.
Member functions:

1. Name of function: main
   Argument list: String[] args
   Return type: void
   Purpose: Launch the application
   Algorithm: NA

2. Name of function: actionPerformed
   Argument list: ActionEvent e

Return type: void
Purpose: Search profile from database.
Algorithm: NA

3. Name of function: mouseClicked
Argument list: MouseEvent e
Return type: void
Purpose: set search field text to null
Algorithm: NA

4. Name of function: mouseClicked
Argument list: MouseEvent e
Return type: void
Purpose: Select profile from database and show it to respected fields.
Algorithm: NA

5. Name of function: actionPerformed
Argument list: ActionEvent e
Return type: void
Purpose: Edit selected profile from database
Algorithm: NA

6. Name of function: actionPerformed
Argument list: ActionEvent e
Return type: void
Purpose: Delete selected profile from database
Algorithm: NA

7. Name of function: actionPerformed
Argument list: ActionEvent e
Return type: void
Purpose: Insert new profile to database.
Algorithm: NA

8. Name of function: actionPerformed
Argument list: ActionEvent e
Return type: void
Purpose: Switch to Administrator mode
Algorithm: NA

9. Name of function: actionPerformed
Argument list: ActionEvent e
Return type: void
Purpose: Change current user
Algorithm: NA

10. Name of function: actionPerformed
    Argument list: ActionEvent e
    Return type: void
    Purpose: Log Out
    Algorithm: NA

11. Name of function: actionPerformed
    Argument list: ActionEvent e
    Return type: void
    Purpose: Load the database
    Algorithm: NA

Class name: Admainfrm
Purpose: To show, search, insert, update or delete user details.
Member functions:

1. Name of function: main
   Argument list: String[] args
   Return type: void
   Purpose: Launch the application
   Algorithm: NA

2. Name of function: actionPerformed
   Argument list: ActionEvent e
   Return type: void
   Purpose: Search profile from database.
   Algorithm: NA

3. Name of function: mouseClicked
   Argument list: MouseEvent e
   Return type: void
   Purpose: Set search field text to null
   Algorithm: NA

4. Name of function: mouseClicked
   Argument list: MouseEvent e
   Return type: void
   Purpose: Select profile from database and show it to respected fields.
   Algorithm: NA

5. Name of function: actionPerformed
   Argument list: ActionEvent e
   Return type: void
   Purpose: Edit selected profile from database
   Algorithm: NA

6. Name of function: actionPerformed
   Argument list: ActionEvent e
   Return type: void
   Purpose: Delete selected profile from database
   Algorithm: NA

7. Name of function: actionPerformed
   Argument list: ActionEvent e
   Return type: void
   Purpose: Insert new profile to database.
   Algorithm: NA

8. Name of function: actionPerformed
   Argument list: ActionEvent e
   Return type: void
   Purpose: Switch to Administrator mode
   Algorithm: NA

9. Name of function: actionPerformed
   Argument list: ActionEvent e
   Return type: void
   Purpose: Change current user
   Algorithm: NA

10. Name of function: actionPerformed
    Argument list: ActionEvent e
    Return type: void
    Purpose: Log Out
    Algorithm: NA

11. Name of function: actionPerformed
    Argument list: ActionEvent e
    Return type: void
    Purpose: Load the database
    Algorithm: NA

Class name: Chusrfrm
Purpose: Change user.
Member functions:

1. Name of function: main
   Argument list: String[] args
   Return type: void
   Purpose: Launch the application
   Function algorithm: Create new frame and make that frame visible.

2. Name of function: actionPerformed
   Argument list: ActionEvent e
   Return type: void
   Purpose: Enable user to log in as other user.
   Algorithm: NA

Class name: Scorecardfrm
Purpose:
Member functions:

1. Name of function: main
   Argument list: String[] args
   Return type: void
   Purpose: Launch the application
   Algorithm: NA

2. Name of function: windowOpened
   Argument list: WindowEvent e
   Return type: void
   Purpose: Load student score card.
   Algorithm: NA

3. Name of function: actionPerformed
   Argument list: ActionEvent e
   Return type: void
   Purpose: Edit student score card
   Algorithm: NA

4. Name of function: actionPerformed
   Argument list: ActionEvent e
   Return type: void
   Purpose: Close score card.
   Algorithm: NA

Class name: Auth
Purpose: Authentication
Member functions:

1. Name of function: main
   Argument list: String[] args
   Return type: void
   Purpose: Launch the application
   Algorithm: NA

2. Name of function: actionPerformed
   Argument list: ActionEvent e
   Return type: void
   Purpose: Authenticate user password.
   Algorithm: NA

3. Name of function: actionPerformed
   Argument list: ActionEvent e
   Return type: void
   Purpose: Cancel authentication
   Algorithm: NA

Class name: DbConnector
Purpose: Connect the database with the application.
Member functions:

1. Name of function: Connector
   Argument list: Blank
   Return type: Connection
   Purpose: Connect the database with java application
   Algorithm: NA

6. DESIGN CODING STANDARD FOLLOWED AND ASSUMPTIONS

CODING STANDARD

The main goal of the recommendation is to improve readability and thereby the understanding and the maintainability and the general quality of the code. It is impossible to cover all the specific cases in a general guide and the programmer should be flexible.

NAMING CONVENTION

Names representing packages should be in all lower case
mypackage, com.company.application.ui
Package naming convention used by Sun for the Java core packages. The initial package name representing the domain name must be in lower case.

Names representing types must be nouns and written in mixed case starting with upper case
Line, AudioSystem Common practice in the Java development community and also the type naming convention used by Sun for the Java core packages.

Variable names must be in mixed case starting with lower case
Makes variables easy to distinguish from types, and effectively resolves potential naming collision as in the declaration Line line.

Names representing constants (final variables) must be all uppercase using underscore to separate words
e.g. MAX_ITERATIONS, COLOR_RED

Names representing methods must be verbs and written in mixed case starting with lower case
getName(), computeTotalWidth()
Common practice in the Java development community and also the naming convention used by Sun for the Java core packages. This is identical to variable names, but methods in Java are already distinguishable from variables by their specific form.

Private class variables should have underscore suffix

```
class Person {
private String name_; ...
}
```

Apart from its name and its type, the scope of a variable is its most important feature. Indicating class scope by using underscore makes it easy to distinguish class variables from local scratch variables. This is important because class variables are considered to have higher significance than method variables, and should be treated with special care by the programmer.

A side effect of the underscore naming convention is that it nicely resolves the problem of finding reasonable variable names for setter methods:

```
void setName(String name){
name_ = name;
}
```

An issue is whether the underscore should be added as a prefix or as a suffix. Both practices are commonly used, but the latter is recommended because it seems to best preserve the readability of the name.

Generic variables should have the same name as their type

```
void setTopic(Topic topic) // NOT: void setTopic(Topic value)
// NOT: void setTopic(Topic aTopic)
// NOT: void setTopic(Topic t)
void connect(Database database) // NOT: void connect(Database db)
// NOT: void connect(Database oracleDB)
```

It reduces complexity by reducing the number of terms and names used. Also makes it easy to deduce the type given a variable name only.

Variables with a large scope should have long names; variables with a small scope can have short names

Scratch variables used for temporary storage or indices are best kept short. A programmer reading such variables should be able to assume that its value is not used outside a few lines of code. Common scratch variables for integers are i, j, k, m, n and for characters’ c and d.

The terms get/set must be used where an attribute is accessed directly

```
employee.getName();
employee.setName(name);
matrix.getElement(2, 4);
matrix.setElement(2, 4, value);
```

Common practice in the Java community and the convention used by Sun for the Java core packages.

The is prefix should be used for boolean variables and methods
isSet, isVisible, isFinished, isFound, isOpen
This is the naming convention for boolean methods and variables used by Sun for the Java core packages. Using the is prefix solves a common problem of choosing bad Boolean names like status or flag. isStatus or isFlag simply doesn't fit, and the programmer is forced to chose more meaningful names.

The term find can be used in methods where something is looked up
vertex.findNearestVertex();
matrix.findSmallestElement();
node.findShortestPath(Node destinationNode);
Give the reader the immediate clue that this is a simple look up method with a minimum of computations involved. Consistent use of the term enhances readability. The n prefix should be used for variables representing a number of objects. E.g. nPoints, nLines etc.

The Iterator variables should be called i, j, k etc.
for (Iterator i = points.iterator(); i.hasNext();) { : }
for (int i = 0; i < nTables; i++) { : }
The notation is taken from mathematics where it is an established convention for indicating iterators. Variables named j, k etc. should be used for nested loops only.

Negated boolean variable names must be avoided
bool isError; // NOT: isNoError bool isFound;
// NOT: isNotFound
The problem arise when the logical not operator is used and double negative arises. It is not immediately apparent what !isNotError means.

User Defined Exception classes should be suffixed with Exception
class AccessException extends Exception { : }
Exception classes are really not part of the main design of the program, and naming them like this makes them stand out relative to the other classes. This standard is followed by Sun in the basic Java library.

Functions (methods returning an object) should be named after what they return and procedures (void methods) after what they do
Increase readability. Makes it clear what the unit should do and especially all the things it is not supposed to do. This again makes it easier to keep the code clean of side effects.

Java source files should have the extension .java
Special characters like TAB and page break must be avoided
These characters are bound to cause problem for editors, printers, terminal emulators or debuggers when used in a multi-programmer, multi-platform environment.

Variables should be initialized where they are declared and they should be declared in the smallest scope possible

This ensures that variables are valid at any time. Sometimes it is impossible to initialize a variable to a valid value where it is declared. In these cases it should be left uninitialized rather than initialized to some phony value.

Variables must never have dual meaning

Enhances readability by ensuring all concepts are represented uniquely. Reduce chance of error by side effects.

Floating point constants should always be written with a digit before the decimal point

double total = 0.5; // NOT: double total = .5;
The 0.5 is a lot more readable than .5; there is no way it can be mixed with the integer.

Arrays should be declared with their brackets next to the type

double[] vertex; // NOT: double vertex[];
int[] count; // NOT: int count[];
public static void main(String[] arguments)
public double[] computeVertex()
The reason for is twofold. First, the array-ness is a feature of the class, not the variable. Second, when returning an array from a method, it is not possible to have the brackets with other than the type (as shown in the last example).

7. TESTING

Test Report

Module	Input	Output	Test Condition	Result
Login	Enter Login Id and Password	Logging into Student Mode if Id or Password is correct or showing an error message: "Wrong Id or Password"	An error message should be displayed if Id or password will wrong: "Wrong Id or Password" Else Log into Student Mode.	Passed
Create New User	Enter login Id and new password	Back to Login module and a message is appearing: "Password create, please login with your Id and Password." If given Id is correct. Else An error message is displaying: "Unable to create password, please contact your admin."	Back to Login module and a message will appear: "Password create, please login with your Id and Password." If given Id is correct. Else An error message should be displayed: "Unable to create password, please contact your admin."	Passed
Student Mode	1. Enter student Id number for searching. 2. Enter student details.	1. Table is showing respective student information according to their Id. 2. Student profile is created if clicked on New	1. Table should display respective student details. 2. Student profile should be created or updated accordingly.	1. Passed 2. Passed

		button or Update Profile if clicked on Update button.		
Administrative Mode	1. Enter Id number for searching. 2. Enter faculty details.	1. Table is showing respective faculty information according to their Id. 2. Faculty profile is created if clicked on New button or Update Profile if clicked on Update button.	1. Table should display respective faculty details. 2. Faculty profile should be created or updated accordingly.	1. Passed 2. Passed
Change User	Enter Login Id and Password	Logging into Student Mode if Id or Password is correct or Show an error message: “Wrong Id or Password”	An error message should be displayed if Id or password will wrong: “Wrong Id or Password” Else Log into respective mode.	Passed
Authentication	Enter Password	Delete selected profile if password is correct. Else showing an error message: “Wrong password.”	Delete selected profile if password is correct. Else should show an error message: “Wrong password.”	Passed
Score Card	Enter score card details.	Score card is updated.	Score card should updated.	Passed

8. FUTURE SCOPE OF THE PROJECT

The following is just a sample of future opportunities that could be implemented-

1. One can upgrade this software to store subject video lectures under different professor names and also maintain the previous year’s question papers in it.

2. The software can be used to take day to day attendance and automatically send an SMS to the students and their parent.

3. One can deploy this software into mobile android application and be used insmaller devices like mobile phones, tablets and notepads.

4. In future the software can be combined with the university internal and external web sites. So that all courses in the university will have single application.

5. Students can directly fill a resume forum system will use artificial intelligence and sent the resume to companies as per student requirement and eligibility criteria

6. The present system may be further upgraded in future even maintain the activities in library like book issue date, submission date and late fines for each student individually.

7. By using artificial intelligence the web portal may track the usage of Wi-Fi based on the students registered device.

9. CONCLUSION

The application was designed for a very basic purpose for maintaining student data like – semester marks, personal data and faculty data in a dynamic manner. The application was programmed using simple and livid codes. The use of highly user friendly software like MySQL Workbench, SQL developer for accessing the database and eclipse 2018-2019 integrated development environment helps in designing the application with ease and in desired manner. The application is embedded with a local server through the software MySQL Server with server port 3306; this enables the application to work on local network. The application is provided with an administrative mode and student mode for updating the student and faculty information time to time. The application eliminates the paper work which could lead to loss of data and data redundancy. The application enables its users to access, manage and update student data effectively and efficiently. It also allows the possibility of queries to obtain information from student & faculty details. Due to the many users viewing and modifying student data in the department, it is an ideal use for such a system.

10. REFERENCES and BIBLOGRAPHY

- https://en.wikipedia.org/wiki/Student_information_system
- https://projectabstracts.com/1664/student-information-system.html
- https://www.scribd.com/document/334841577/Synopsis-of-Student-Information-System
- http://www.microtek.ac.in/UploadedFiles/Downloads/SYNOPSIS%20Example.pdf
- https://employment.blurtit.com/3206502/what-are-the-scope-and-limitation-in-student-information-system
- http://www.authorstream.com/Presentation/shashijais789-1244141-student-information-system/

11. APPENDIX

Student Information System - Student Mode Module

```
import java.awt.BorderLayout;
import javax.swing.JFrame;
import javax.swing.JPanel;
import net.proteanit.sql.DbUtils;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
public class Mainfrm extends JFrame {
	private JPanel contentPane;
	private JTextField id;
	private JTextField ln;
	private JTextField dob;
	private JTextField fn;
	private JTextField sex;
	private JTextField university;
	private JTextField roll;
	private JButton btnNew;
	private JButton btnDelete;
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					Mainfrm frame = new Mainfrm();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}
	Connection con = null;
	private JTextField SearchKey;
	public Mainfrm() {
		con = DbConnector.Connector();
		Glvar.openfrm = "Mainfrm";
		setResizable(false);
		setTitle("Student Information System - Student Mode");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(173, 30, 1178, 750);
		contentPane = new JPanel();
		contentPane.setBackground(new Color(176, 224, 230));
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
		setContentPane(contentPane);
		contentPane.setLayout(null);
		JPanel panel_2 = new JPanel();
		panel_2.setBackground(SystemColor.info);
		panel_2.setBounds(10, 10, 536, 24);
		panel.add(panel_2);
		panel_2.setLayout(null);
		JButton btnSearch = new JButton("Search");
		btnSearch.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				btnDelete.setEnabled(false);
				btnEdit.setEnabled(false);
				btnScoreCard.setEnabled(false);
				String sql = "select * from student where id = ?";
				String searchkey = SearchKey.getText();
				try {
					PreparedStatement ps = con.prepareStatement(sql);
					ps.setString(1, searchkey);
					ResultSet rs = ps.executeQuery();
					table.setModel(DbUtils.resultSetToTableModel(rs));
				} catch (SQLException e1) {
					e1.printStackTrace();
					JOptionPane.showMessageDialog(null,
e1.getMessage());
				}
			}
		});
		btnSearch.setFont(new Font("Tahoma", Font.PLAIN, 16));
		btnSearch.setBounds(441, 0, 85, 24);
		panel_2.add(btnSearch);
		SearchKey = new JTextField();
		SearchKey.addMouseListener(new MouseAdapter() {
			@Override
			public void mouseClicked(MouseEvent e) {
				SearchKey.setText(null);
			}
		});
		SearchKey.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				btnDelete.setEnabled(false);
				btnEdit.setEnabled(false);
				btnScoreCard.setEnabled(false);
				String sql = "select * from student where id = ?";
				String searchkey = SearchKey.getText();
				try {
					PreparedStatement ps = con.prepareStatement(sql);
					ps.setString(1, searchkey);
					ResultSet rs = ps.executeQuery();
					table.setModel(DbUtils.resultSetToTableModel(rs));
				} catch (SQLException e1) {
					e1.printStackTrace();
					JOptionPane.showMessageDialog(null,
e1.getMessage());
				}
			}
		});
		SearchKey.setText("Enter Id here");
		SearchKey.setFont(new Font("Tahoma", Font.PLAIN, 12));
		SearchKey.setColumns(10);
		SearchKey.setBounds(281, 0, 150, 24);
		panel_2.add(SearchKey);
		JScrollPane scrollPane = new JScrollPane();
		scrollPane.setBounds(10, 10, 516, 511);
		panel_8.add(scrollPane);
		table = new JTable();
		table.addMouseListener(new MouseAdapter() {
			@Override
			public void mouseClicked(MouseEvent e) {
				btnDelete.setEnabled(true);
				btnEdit.setEnabled(true);
				btnScoreCard.setEnabled(true);

				try {
					int row = table.getSelectedRow();
					Glvar.stdn_id = (table.getModel().getValueAt(row,
0)).toString();

					String sql = "SELECT * FROM sis.student where id = ?";
					PreparedStatement ps = con.prepareStatement(sql);
					ps.setString(1, Glvar.stdn_id);
					ResultSet rs = ps.executeQuery();

					while (rs.next()) {
						id.setText(rs.getString("id"));
						fn.setText(rs.getString("fname"));
						ln.setText(rs.getString("lname"));
						sex.setText(rs.getString("sex"));
						dob.setText(rs.getString("dob"));
						roll.setText(rs.getString("roll_no"));
						Glvar.stdn_roll = rs.getString("roll_no");
					}
				} catch (SQLException e1) {
					e1.printStackTrace();
```

```
							}
						}
					});
					scrollPane.setViewportView(table);
					table.setBackground(SystemColor.info);
					JPanel panel_3 = new JPanel();
					panel_3.setBackground(new Color(176, 224, 230));
					panel_3.setLayout(null);
					panel_3.setBounds(10, 551, 536, 24);
					panel_1.add(panel_3);
					btnEdit = new JButton("Edit");
					btnEdit.setEnabled(false);
					btnEdit.addActionListener(new ActionListener() {
						public void actionPerformed(ActionEvent e) {
							boolean bt = true;
							boolean bf = false;
							String txt = btnEdit.getText();
							String t = null;

							String iddb = id.getText();
							String fndb = fn.getText();
							String lndb = ln.getText();
							String sexdb = sex.getText();
							String dobdb = dob.getText();
							String rolldb = roll.getText();
							if (txt.equals("Edit")) {
								btnEdit.setText("Update");
								id.setEditable(bt);
								fn.setEditable(bt);
								ln.setEditable(bt);
								sex.setEditable(bt);
								dob.setEditable(bt);
								roll.setEditable(bt);
							}
							else if (txt.equals("Update")) {
								try {
									String sql = "UPDATE `sis`.`student` SET `id` = ?,
`fname` = ?, `lname` = ?, `sex` = ?, `dob` = ?, `contact_no` = ?, `email` = ?, `address` = ?,
`university` = ?, `college` = ?, `department` = ?, `section` = ?, `reg_no` = ?, `roll_no` = ?
WHERE (`id` = ?)";
									PreparedStatement ps =
con.prepareStatement(sql);
									ps.setString(1, iddb);
									ps.setString(2, fndb);
									ps.setString(3, lndb);
									ps.setString(4, sexdb);
									ps.setString(5, dobdb);
						ps.setString(14, rolldb);
						ps.setString(15, Glvar.stdn_id);
						int res = ps.executeUpdate();
						if (res > 0)
							JOptionPane.showMessageDialog(null,
"Profile Updated. Please refresh database.");
						else
							JOptionPane.showMessageDialog(null,
"Unable to update");
					} catch (SQLException e1) {
						e1.printStackTrace();
						JOptionPane.showMessageDialog(null,
e1.getMessage());
					}
					btnEdit.setText("Edit");
					id.setEditable(bf);
					id.setText(t);
					fn.setEditable(bf);
					fn.setText(t);
					ln.setEditable(bf);
					ln.setText(t);
					sex.setEditable(bf);
					sex.setText(t);
					dob.setEditable(bf);
					dob.setText(t);
					roll.setText(t);
				}
			}
		});
		btnEdit.setFont(new Font("Tahoma", Font.PLAIN, 16));
		btnEdit.setBounds(226, 0, 85, 24);
		panel_3.add(btnEdit);
		btnDelete = new JButton("Delete");
		btnDelete.setEnabled(false);
		btnDelete.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				Auth authobj = new Auth();
				authobj.setVisible(true);
				boolean bf = false;
				String t = null;
				id.setEditable(bf);
				id.setText(t);
				fn.setEditable(bf);
				fn.setText(t);
				ln.setEditable(bf);
				ln.setText(t);
				sex.setEditable(bf);
```

```
                    sex.setText(t);
                    dob.setEditable(bf);
                    dob.setText(t);
                    roll.setText(t);
                }
            });
            btnDelete.setFont(new Font("Tahoma", Font.PLAIN, 16));
            btnDelete.setBounds(441, 0, 85, 24);
            panel_3.add(btnDelete);
            btnNew = new JButton("New");
            btnNew.setBounds(10, 0, 85, 24);
            panel_3.add(btnNew);
            btnNew.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    boolean bt = true;
                    boolean bf = false;
                    String t = null;
                    String txt = btnNew.getText();
                    String iddb = id.getText();
                    String fndb = fn.getText();
                    String lndb = ln.getText();
                    String sexdb = sex.getText();
                    String dobdb = dob.getText();
                    String rolldb = roll.getText();
                    if (txt.equals("New")) {
                        btnNew.setText("Create");
                        id.setEditable(bt);
                        id.setText(t);
                        fn.setEditable(bt);
                        fn.setText(t);
                        ln.setEditable(bt);
                        ln.setText(t);
                        sex.setEditable(bt);
                        sex.setText(t);
                        dob.setEditable(bt);
                        dob.setText(t);
                        roll.setEditable(bt);
                        roll.setText(t);
                    }
                    else if(txt.equals("Create")) {
                        try {
                            String sql = "INSERT INTO `sis`.`student` (`id`,
`fname`, `lname`, `sex`, `dob`, `contact_no`, `email`, `address`, `university`, `college`,
`department`, `section`, `reg_no`, `roll_no`) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
                            PreparedStatement ps =
con.prepareStatement(sql);
                            ps.setString(1, iddb);
```

```
							ps.setString(2, fndb);
							ps.setString(3, lndb);
							ps.setString(4, sexdb);
							ps.setString(5, dobdb);
							ps.setString(14, rolldb);
							int res = ps.executeUpdate();
							if (res > 0)
								JOptionPane.showMessageDialog(null,
"Profile created. Please refresh database.");
							else
								JOptionPane.showMessageDialog(null,
"Profile not created");
							String sql1 = "INSERT INTO `sis`.`score`
(`roll_no`) VALUES (?);";
							PreparedStatement ps1 =
con.prepareStatement(sql1);
							ps1.setString(1, rolldb);
							int res1 = ps1.executeUpdate();
							if (res1 > 0)
								JOptionPane.showMessageDialog(null,
"Blank score card generated.");
							else
								JOptionPane.showMessageDialog(null,
"Unable to generate score card");
						} catch (SQLException e1) {
							e1.printStackTrace();
							JOptionPane.showMessageDialog(null,
e1.getMessage());
						}
						btnNew.setText("New");
						id.setEditable(bf);
						id.setText(t);
						fn.setEditable(bf);
						fn.setText(t);
						ln.setEditable(bf);
						ln.setText(t);
						sex.setEditable(bf);
						sex.setText(t);
						dob.setEditable(bf);
						dob.setText(t);
						roll.setEditable(bf);
						roll.setText(t);
					}
				}
			});
			btnNew.setFont(new Font("Tahoma", Font.PLAIN, 16));
			JPanel panel_7 = new JPanel();
```

```
panel_7.setBackground(new Color(176, 224, 230));
panel_7.setBounds(10, 58, 536, 429);
panel_1.add(panel_7);
id.setEditable(false);
id.setFont(new Font("Tahoma", Font.PLAIN, 12));
id.setBounds(120, 10, 100, 20);
panel_4.add(id);
id.setColumns(10);
JLabel lblSex = new JLabel("Sex *");
panel_4.add(dob);
JLabel lblLastName = new JLabel("Last Name");
lblLastName.setFont(new Font("Tahoma", Font.BOLD, 14));
lblLastName.setBounds(316, 40, 100, 20);
panel_4.add(lblLastName);
JLabel lblBirthDate = new JLabel("Birth Date *");
lblBirthDate.setFont(new Font("Tahoma", Font.BOLD, 14));
lblBirthDate.setBounds(316, 70, 100, 20);
panel_4.add(lblBirthDate);
fn = new JTextField();
fn.setEditable(false);
fn.setFont(new Font("Tahoma", Font.PLAIN, 12));
fn.setColumns(10);
fn.setBounds(120, 40, 100, 20);
panel_4.add(fn);
sex = new JTextField();
university.setEditable(false);
university.setFont(new Font("Tahoma", Font.PLAIN, 12));
university.setColumns(10);
university.setBounds(120, 10, 100, 20);
panel_5.add(university);
JLabel lblRegNo = new JLabel("Reg. No. **");
lblRegNo.setFont(new Font("Tahoma", Font.BOLD, 14));
lblRegNo.setBounds(10, 70, 100, 20);
panel_5.add(lblRegNo);
section = new JTextField();
JLabel lblContactNo = new JLabel("Contact No.");
lblContactNo.setFont(new Font("Tahoma", Font.BOLD, 14));
lblContactNo.setBounds(10, 10, 100, 20);
panel_6.add(lblAddress);
address = new JTextArea();
address.setFont(new Font("Monospaced", Font.PLAIN, 12));
address.setWrapStyleWord(true);
address.setEditable(false);
address.setLineWrap(true);
address.setBounds(426, 10, 100, 50);
panel_6.add(address);
JLabel lblNewLabel_1 = new JLabel("* marked fields are mandatory.  **
marked fields are mandatory & should be unique.");
		lblNewLabel_1.setForeground(Color.BLACK);
		lblNewLabel_1.setFont(new Font("Tahoma", Font.ITALIC, 12));
		lblNewLabel_1.setBounds(10, 497, 536, 13);
		panel_1.add(lblNewLabel_1);
		btnScoreCard = new JButton("Score Card");
		btnScoreCard.setEnabled(false);
		btnScoreCard.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				Scorecardfrm scorecardfrmobj = new Scorecardfrm();
				scorecardfrmobj.setVisible(true);
			}
		});
		btnScoreCard.setFont(new Font("Tahoma", Font.PLAIN, 16));
		btnScoreCard.setBounds(425, 10, 121, 24);
		panel_1.add(btnScoreCard);
		JPanel logedin_panel = new JPanel();
		logedin_panel.setBackground(new Color(176, 224, 230));
		logedin_panel.setBounds(10, 10, 1144, 22);
		contentPane.add(logedin_panel);
		logedin_panel.setLayout(null);
		JLabel lblUserName = new JLabel(Glvar.logedinuser);
		lblUserName.setForeground(Color.BLACK);
		lblUserName.setFont(new Font("Tahoma", Font.PLAIN, 14));
		lblUserName.setBounds(764, 0, 106, 22);
		logedin_panel.add(lblUserName);
		JButton btnChangeUser = new JButton("Change User");
		btnChangeUser.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				dispose();
				Chusrfrm chusrfrmobj = new Chusrfrm();
				chusrfrmobj.setVisible(true);

			}
		});
		btnChangeUser.setFont(new Font("Tahoma", Font.PLAIN, 14));
		btnChangeUser.setBounds(1012, 0, 122, 21);
		logedin_panel.add(btnChangeUser);
		JButton btnSwitchToAdministrator = new JButton("Switch to administrator
mode");
		btnSwitchToAdministrator.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				dispose();
				Admainfrm admainfrmobj = new Admainfrm();
				admainfrmobj.setVisible(true);
			}
		});
```

```
btnSwitchToAdministrator.setFont(new Font("Tahoma", Font.PLAIN, 14));
btnSwitchToAdministrator.setBounds(10, 0, 227, 21);
logedin_panel.add(btnSwitchToAdministrator);
JLabel lblStudentMode = new JLabel("Student Mode");
lblStudentMode.setHorizontalAlignment(SwingConstants.CENTER);
lblStudentMode.setFont(new Font("Tahoma", Font.BOLD | Font.ITALIC, 14));
lblStudentMode.setBounds(247, 0, 153, 22);
logedin_panel.add(lblStudentMode);
JButton btnLogOut = new JButton("Log Out");
btnLogOut.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
                Loginfrm loginfrmobj = new Loginfrm();
                loginfrmobj.Visible(true);
                dispose();
        }
});
btnLogOut.setFont(new Font("Tahoma", Font.PLAIN, 14));
btnLogOut.setBounds(880, 0, 122, 21);
logedin_panel.add(btnLogOut);
JPanel mnu_btn_panel = new JPanel();
mnu_btn_panel.setBackground(new Color(176, 224, 230));
mnu_btn_panel.setBounds(10, 54, 1144, 34);
contentPane.add(mnu_btn_panel);
mnu_btn_panel.setLayout(null);
JButton loaddb = new JButton("Load Database");
loaddb.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
                loaddb.setText("Refresh Database");
                SearchKey.setText("Enter Id here");
                boolean bf = false;
                String t = null;
                Glvar.stdn_roll = null;
                btnDelete.setEnabled(false);
                btnEdit.setEnabled(false);
                btnScoreCard.setEnabled(false);
                id.setEditable(bf);
                id.setText(t);
                fn.setEditable(bf);
                fn.setText(t);
                ln.setEditable(bf);
                ln.setText(t);
                sex.setEditable(bf);
                sex.setText(t);
                dob.setEditable(bf);
                dob.setText(t);
                try {
                        String sql = "SELECT * FROM sis.student;";
```

```
					PreparedStatement ps = con.prepareStatement(sql);
					ResultSet rs = ps.executeQuery();
					table.setModel(DbUtils.resultSetToTableModel(rs));
				} catch (SQLException e1) {
					e1.printStackTrace();
					JOptionPane.showMessageDialog(null,
e1.getMessage());
				}
			}
		});
		loaddb.setFont(new Font("Tahoma", Font.BOLD, 18));
		loaddb.setBounds(10, 0, 227, 34);
		mnu_btn_panel.add(loaddb);
	}
}
```

Student Information System - Login Module

```
import java.awt.EventQueue;
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.Font;
import javax.swing.JTextField;
public class Loginfrm {
	private JFrame frmStudentInformationSystem;
	private JTextField textField;
	private JPasswordField passwordField;
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					Loginfrm window = new Loginfrm();
					window.frmStudentInformationSystem.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}
	Connection con = null;
	public Loginfrm() {
		initialize();
	}
	private void initialize() {
		con = DbConnector.Connector();
		frmStudentInformationSystem = new JFrame();
		frmStudentInformationSystem.setResizable(false);
		frmStudentInformationSystem.setBackground(SystemColor.menu);
		frmStudentInformationSystem.getContentPane().setBackground(new
Color(176, 224, 230));
		frmStudentInformationSystem.setTitle("Student Information System - Login");
		JPanel panel = new JPanel();
		panel.setBackground(new Color(176, 224, 230));
		panel.setBounds(10, 53, 304, 290);
		frmStudentInformationSystem.getContentPane().add(panel);
		panel.setLayout(null);
		JLabel lblLoginId = new JLabel("Login Id");
		lblLoginId.setFont(new Font("Tahoma", Font.BOLD, 18));
		lblLoginId.setBounds(10, 51, 111, 28);
		panel.add(lblLoginId);
		passwordField = new JPasswordField();
		passwordField.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				String id = textField.getText();
				String pwd = new String(passwordField.getPassword());
				String sql = "select login_id, password from faculty where
login_id = ?";
				String loginid = null;
				String password = null;
				try {
					PreparedStatement ps = con.prepareStatement(sql);
					ps.setString(1, id);
					ResultSet rs = ps.executeQuery();

					while (rs.next()) {
						loginid = rs.getString("login_id");
						password = rs.getString("password");
					}
					ps.close();
				} catch (SQLException e1) {
					e1.printStackTrace();
				}
				if (id.equals(loginid) && pwd.equals(password)) {
					Glvar.logedinuser = loginid;
					Glvar.logedinusr_pwd = password;
					frmStudentInformationSystem.dispose();
					Mainfrm mainfrmobj = new Mainfrm();
					mainfrmobj.setVisible(true);
				}
				else {
					JOptionPane.showMessageDialog(null, "Wrong Id or
Password");
```

```
					}
				}
			});
			passwordField.setBounds(131, 100, 141, 27);
			panel.add(passwordField);
			JButton btnLogin = new JButton("LogIn");
			btnLogin.addActionListener(new ActionListener() {
				public void actionPerformed(ActionEvent e) {
					String id = textField.getText();
					String pwd = new String(passwordField.getPassword());
					String sql = "select login_id, password from faculty where
login_id = ?";
					String loginid = null;
					String password = null;
					try {
						PreparedStatement ps = con.prepareStatement(sql);
						ps.setString(1, id);
						ResultSet rs = ps.executeQuery();
						while (rs.next()) {
							loginid = rs.getString("login_id");
							password = rs.getString("password");
						}
						ps.close();
					} catch (SQLException e1) {
						e1.printStackTrace();
					}
					if (id.equals(loginid) && pwd.equals(password)) {
						Glvar.logedinuser = loginid;
						Glvar.logedinusr_pwd = password;
						frmStudentInformationSystem.dispose();
						Mainfrm mainfrmobj = new Mainfrm();
						mainfrmobj.setVisible(true);
					}
					else {
						JOptionPane.showMessageDialog(null, "Wrong Id or
Password");
					}


				}
			});
			btnLogin.setFont(new Font("Tahoma", Font.BOLD, 18));
			btnLogin.setBounds(99, 185, 105, 28);
			panel.add(btnLogin);
			JButton btnSignUp = new JButton("Sign Up");
			btnSignUp.addActionListener(new ActionListener() {
				public void actionPerformed(ActionEvent e) {
					Createusrfrm createusrfrmobj = new Createusrfrm();
					createusrfrmobj.setVisible(true);
				}
		});
		btnSignUp.setFont(new Font("Tahoma", Font.BOLD, 18));
		btnSignUp.setBounds(104, 136, 105, 28);
		panel_1.add(btnSignUp);
		JPanel panel_2 = new JPanel();
		panel_2.setBackground(Color.DARK_GRAY);
		panel_2.setBounds(335, 53, 10, 124);
		frmStudentInformationSystem.getContentPane().add(panel_2);
		JPanel panel_3 = new JPanel();
		panel_3.setBackground(Color.DARK_GRAY);
		panel_3.setBounds(335, 219, 10, 124);
		frmStudentInformationSystem.getContentPane().add(panel_3);
	}
	public void Visible(boolean b) {
		Loginfrm window = new Loginfrm();
		window.frmStudentInformationSystem.setVisible(true);
	}
}
```

Student Information System – DbConnector Class

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import com.mysql.jdbc.Driver;
public class DbConnector {
	public static Connection con = null;
	public static String url =
"jdbc:mysql://localhost/sis?autoReconnect=true&useSSL=false";
	public static String user = "root";
	public static String password = "root";
	public static Connection Connector() {
		try {
			Driver d = new Driver();
			DriverManager.registerDriver(d);
			con = DriverManager.getConnection(url, user, password);
			return con;
		} catch (SQLException e) {
			e.printStackTrace();
			return null;
		}
	}
}
```

Student Information System – Authentication Module

```
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import java.awt.Font;
import javax.swing.JPasswordField;
import java.awt.Color;
public class Auth extends JFrame {
	private JPanel contentPane;
	private JPasswordField passwordField;
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					Auth frame = new Auth();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}
	Connection con = null;
	public Auth() {
		setTitle("Student Information System - Authentication");
		con = DbConnector.Connector();
		setResizable(false);
		setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
		setBounds(935, 520, 383, 227);
		contentPane = new JPanel();
		contentPane.setBackground(new Color(176, 224, 230));
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
		setContentPane(contentPane);
		contentPane.setLayout(null);
		JPanel panel = new JPanel();
		panel.setBackground(new Color(176, 224, 230));
		panel.setBounds(10, 39, 349, 141);
		contentPane.add(panel);
		panel.setLayout(null);
		JLabel lblNewLabel = new JLabel("User Id :");
		lblNewLabel.setFont(new Font("Tahoma", Font.PLAIN, 14));
		lblNewLabel.setBounds(10, 10, 60, 26);
		panel.add(lblNewLabel);
		passwordField = new JPasswordField();
		passwordField.setBounds(199, 46, 140, 25);
		panel.add(passwordField);
		JButton btnConfirm = new JButton("Confirm");
		btnConfirm.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				String pwd = new String(passwordField.getPassword());
				if (pwd.equals(Glvar.logedinusr_pwd)) {
					if (Glvar.openfrm.equals("Admainfrm")) {
						try {
							String sql = "DELETE FROM `sis`.`faculty` 
WHERE (`login_id` = ?)";
							PreparedStatement ps = 
con.prepareStatement(sql);
							ps.setString(1, Glvar.uid_db);
							int res = ps.executeUpdate();
							if (res > 0) {
								
	JOptionPane.showMessageDialog(null, "Profile Deleted. Please refresh database.");
								Glvar.stdn_id = null;
								Glvar.stdn_roll = null;
							}
							else
								
	JOptionPane.showMessageDialog(null, "Unable to delete.");
						} catch (SQLException e1) {
							e1.printStackTrace();
						}
						Glvar.uid_db = null;
						dispose();
					}
					else if (Glvar.openfrm.equals("Mainfrm")) {
						try {
							String sql0 = "DELETE FROM `sis`.`score` 
WHERE (`roll_no` = ?)";
							PreparedStatement ps0 = 
con.prepareStatement(sql0);
							ps0.setString(1, Glvar.stdn_roll);
							int res0 = ps0.executeUpdate();
							if (res0 > 0)
								
	JOptionPane.showMessageDialog(null, "Score card deleted.");
							else
								
	JOptionPane.showMessageDialog(null, "Unable to delete score card.");
							String sql = "DELETE FROM `sis`.`student` 
WHERE (`id` = ?)";
							PreparedStatement ps = 
con.prepareStatement(sql);
```

```
								ps.setString(1, Glvar.stdn_id);
								int res = ps.executeUpdate();
								if (res > 0)

	JOptionPane.showMessageDialog(null, "Profile deleted. Please refresh database.");
								else

	JOptionPane.showMessageDialog(null, "Unable to delete.");
							} catch (SQLException e1) {
								e1.printStackTrace();
							}
							dispose();
						}
					}
					else
						JOptionPane.showMessageDialog(null, "Wrong
password");
				}
			});
			btnConfirm.setFont(new Font("Tahoma", Font.BOLD, 14));
			btnConfirm.setBounds(10, 110, 109, 21);
			panel.add(btnConfirm);
			JButton btnCancel = new JButton("Cancel");
			btnCancel.addActionListener(new ActionListener() {
				public void actionPerformed(ActionEvent e) {
					dispose();
				}
			});
			btnCancel.setFont(new Font("Tahoma", Font.BOLD, 14));
			btnCancel.setBounds(230, 110, 109, 21);
			panel.add(btnCancel);
			JLabel lblPleaseConfirmYour = new JLabel("Please confirm your password");
			lblPleaseConfirmYour.setHorizontalAlignment(SwingConstants.CENTER);
			lblPleaseConfirmYour.setFont(new Font("Tahoma", Font.BOLD, 14));
			lblPleaseConfirmYour.setBounds(10, 10, 349, 26);
			contentPane.add(lblPleaseConfirmYour);
	}
}
```