#### A

#### Project Report

**On**

**Student Record Management**

**Submitted in partial fulfillment of the requirements for the award of the “Bachelor in Computer Application”**

**(B.C.A.)**



**Submitted By**

**Harsh Pal - (ROLL NO. 1709541769)**

**Mukul Kashyap (ROLL NO. 1709541793)**

**Asif Ali (ROLL NO. 1709541760)**

**UNDER THE GUIDANCE**

**Mrs. Ritu Mittal (Sr. Lecturer)**

**D.A.V. (P.G.) College, Muzaffarnagar**

DECLARATION

This is to certify that the dissertation / project report entitled **“Student Record Management”** is an authentic work carried out for the partial fulfillment of the Bachelors Degree of Computer Application under the guidance of **Mrs. Ritu Mittal (Sr. Lecturer)** The matter embodied in this project work has not been submitted earlier for award of any degree or diploma to the best of my knowledge and belief.

**Harsh Pal**

**Mukul Kashyap**

**Asif Ali**

**Acknowledgement**

This project report on **"Student Record Management”** is the result of idea and suggestions to me by **Mrs. Ritu Mittal (Lecturer).**

I have received unfailing encouragement and inspiration of **Mrs. Ritu Mittal (Lecturer)** whose exceptional knowledge and unparalleled behavior is full of ardent inspiration in it. However, we can never adequate thank all those who have their assistance, guidance, cooperation criticism contributed to the improvement of this report. I am ebullient in expressing my intense in debtless heartiest gratitude to all of them.

Since performance feedback is essential for effective communication, mistakes and creative feedback of the report may be unhesitatingly communicated to me, which will be as far as possible duly acknowledged and most welcome.

In this report, whatever is beneficial comes from almighty, and whatever is faulty is mine.

**Harsh Pal**

**Mukul Kashyap**

**Asif Ali**

**INDEX**

**Topic**

1. **INTRODUCTION**

* **SCOPE**
* **Project Description**
* **Objective of the Project**

1. **TECHNOLOGY USED**

* **FRONT END**
* **BACK END**

1. **REQUIREMENTS**

* **SOFTWARE**
* **HARDWARE**

1. **INTRODUCTION TO JAVA**
2. **DATA BASE TABLE**
3. **SNAPSHOTS**
4. **FUTURE SCOPE**
5. **CONCLUSION**
6. **CODING**
7. **BIBLIOGRAPHY**

**INTRODUCTION**

University wants to computerize its admission process for higher education courses. Basic objectives are to extend their reach to geographically scattered students, reducing time in activities, centralized data handling and paperless admission with reduced manpower. Cost cutting, operational efficiency, consist view of data and integration with other institutions are other factors. Main challenges are effectively sync internal and external operations in such a manner that job can be finished within time limit and integration with different agencies on an agreed upon common data format.

**Existing System**

The Student Record Management is a browser which is not totally computerized especially for university admission process. The system takes lots of time in performing different activities, and there is no centralized data handling. There is no integration in the current system upon common data format.

**Proposed System**

The Proposed system is a browser which is completely related to internet browsing. The web enabled information management system designed to automate the entire operations of a modern. This system allows multi-divisional, multi-department system handling that includes various activities.

**Functions**

* Administrator
* Student User

**Admin users** - Has full access to all the modules of this system. Responsible for the accounts of all students, colleges and remittances. Prepares and submits also Daily Reports, student reports, colleges reports, etc.

**SCOPE OF THE PROJECT**

The proposed system can manage all drawback of the existing manual system. The system has user friendly environment where the end user can perform all the activity. Even if the user has little knowledge of computer can easily operate.

This project can easily be installed in computer system.

The system has following objective to be achieved ……

* User friendly environment
* Less space
* Fast retrieval
* Easy to operate
* Accuracy
* Cost reduction

**Project Description**

The Student Record Management is decided to computerize its facilities in their customer to implement Customer. Student Record Management is high performance software, which speeds up the business operations of the organization. Every organization, which deals with the student information, put its great effort in the efficient utilization of its Stock according to its need and requirement. The organization has to perform number of tasks and operations in order to run its business in a manual system. For example:

**Problem Definition of Existing System**

Existing system is manual so it has many limitations and buggs these limitations are.

* If the lost any file or records of customers, it is difficult to find that because there is only one copy of the record.
* It take more time to process the record in manual system and very harm full to find any record of good in previews days.
* Humans can always make errors, but this is not true with system.
* This system is very user friendly.

**Requirement Specification**

The existing system in the Student Record Management manually where are work is done manually by exiting the record in the various file. Maintaining these file requires a lot of space as well as lot of man power also there is possibility of delicacy of data. Therefore the requirement of the project an as follow.

Greater storage capacity Greater processing speed of input and output operations More reliable and consistent produce to eliminate errors in handling day to day processing.

**Disadvantages of current system:**

**As we know the manual processing is quite tedious, time Consuming, less accurate in comparison to computerized processing. Obviously the Present system is not is exception consultant encountering all the above problems.**

* **Time consuming.**
* **It is very tedious.**
* **All information is not placed separately.**
* **Lot of paper work.**
* **Slow data processing.**
* **Not user-friendly environment.**
* **It is difficult to found records due file management system.**

**Advantages of proposed system over current system:**

**In new computerized system I tried to give these facilities**

**1. Manually system changes into computerized System**

**2. Friendly user interface.**

**3. Time saving.**

**4. Save paper work.**

**5. Connecting to database so we use different type of queries, data report.**

**6. Give facility of different type of inquiry.**

**7. Formatted data.**

**8. Data’s are easily approachable**

**OBJECTIVE OF THE PROJECTOBJECTIVE OF THE PROJECT**

**Objective:** basically is software used for the management of students in a school. In this software we can perform several task related to students like

* We can insert information about a new student into database or we can say that we can perform the task of admission.
* We can modify any detail related to student.
* We can also delete record of any student.
* We can see status of any student either by roll no wise or class wise with stream.
* If we see status roll no wise then we have to supply roll know of that particular student.
* In case when we see status according to class and stream then we have to select particular class as well as stream of our requirements.
* We can create new user login for project.
* We can also change password.
* Our project also contains about us form which shows our introduction.

The main objective of the Student Record Management is to perform all functions is correctly and accurately.

It overcome all the problems that we have in our existing manual system, our existing system is as follow.

**Data Duplicate**

Duplication of data means that the same record was repeated a number of times, due to this, our data base become lengthy and difficult to maintain.

If there is duplication of data, then there is a possibility of inconsistency of data, due to the above problem it is very difficult to handle the database.

**Check the Validity**

Check the validity of Input data and give a feedback to the user in case of errors. Protection towards customer privacy concerns. Fast data accessing according to requirements.

**Category:**

**Student Record Management Project comes under the category of application software. A GUI (Graphical User Interface) environment is provided to the Storekeeper so that he can use it very easily as GUI environment is more users friendly.**

**Requirement**

**H/W and S/W**

**Requirement H/W and S/W**

**HARDWARE**

Processor : Dual Core, and Above

Memory : 512 MB and above

Hard Disk : 500 GB

**SOFTWARE**

Operating System : Window XP, 8, and 10

Front End : Netbean

Back End : MS Access

**DATABASE DESIGN**

**Introduction:**

The database is a collection of information and is systematically stored in tables in the form of rows and columns. The table in the database has unique name that identifies its contents. The database in turn is further described in detail giving all the fields used with the data types, constraints available, primary key and foreign key.

Database design is used to manage large of information. In this database we describe the entire 2 table available in the software, which are used to store all the records.

**Data types and its description:**

Fields in database table have a data type used in database table are explained below.

1. **int:** one optional sign character (+ or -) followed by at least one digit (0-9). Leading and trailing blanks are ignored. No other character is allowed.
2. **nvarchar:** It is used to store alpha numeric characters. In this data type we can set the maximum number of characters up to 8000 ranges by defaults SQL server will set the size to 50 characters range.
3. **nchar:** It is used to store alpha numeric characters. By defaults SQL server will set the size to 50 characters range.

**TABLES**

Result Processing System contains the following tables.

1. Login
2. Personal

Description of these tables are as given below.

**SYSTEM EVALUATION AND TESTING**

During system testing, the system is used experimentally to ensure that the software does not fall, i.e. it will run according to specification and in the way users expect. Special test data input for processing and the result examination. A limited number of users may be allowed to use the system so that analysis can see whether they use it in unforeseen ways. It is preferable to discover any surprises the organization implements the system and encounters them later on. This type of testing which allows only a few, selected users to work on the system is known as BETA TESTING. On the other hand the testing done by the developer (s), themselves is known as ALPHA TESTING.

Testing is generally performed by persons other than those who the original programs to ensure complete and unbiased testing and more reliable software.

The norms followed during the testing phase were thoroughly tested by me, the developer; I was required to release the program’s source code. The source code was copied into the area. If some changes were desired in the program, I was suitably informed and was required to make the necessary modifications.

Following testing procedures were used

**UNIT TESTING**

Unit testing is the testing of a single program module in an isolated environment. The testing of the processing procedure is the main focus. In this regard, all the modules were separately tested first as isolated and complete entities. This helped a lot in discovering problems related to a single module and rectifying them in the contest of the module itself, rather than considering module-related problem in a global context.

**INTEGRATION TESTING**

Integration testing is the process of testing the interfaces among system modules. Some testing ensures that data moves between systems as intended. In the regard of a particular module was obtained in correct format, so that the next module could accept it appropriately. This could be done either after the entire system was developed, or in stages than one module was developed.

**SYSTEM TESTING**

System testing is the testing of the system against its initial objectives. Such testing is done either in a simulated environment or in a live environment.

In the case of the system which I developed, the testing was done in a pseudo-real environment. The organization has an area called the ‘Development ‘, where all development work is carried out, whereas the ‘Client’ is the area where all on-line work is undertaken. Since the kind of work which I was involved dealt with insertion, updating and in tables, all the work was developed in the ‘Development’ area. I call the environment pseudo-real since all data which is being used for on-line transaction can be copied into the tables of the ‘Development’ and live data can be worked on. Thus whatever would be the output on the ‘Client ‘is reflected exactly on the ‘ Development ’ also. Thus, discovery of error became simpler in that the errors which would occur in the real-time environment were exactly duplicated in the ‘Development ‘environment.

All the above types of testing were carried out and various extreme cases were also introduced to check for whether the system responded as expected. The various Forms, database packages, procedures, function; java programs and functions were tested to discover they behaved in the manner expected of them and gave accurate results. However, the factor, which helped most in the modification and rectification of the system, was the user response. The user after using the system would invariably come up with some idea to improve the system. Such ideas, if feasible, were incorporated into the system, thus leading to an improvement in the overall efficiency of the system.

**SYSTEM IMPLEMENTATION**

Implementation, literally, means to put into effect or to carry out. The system implementation phase of software engineering deals with all activities that take place to convert from the old system to the new. Proper implementation is essential to provide system to meet organization’s requirements.

During the implementation phase debugging, documentation of the system was created out, module in the project for

Accuracy of Results,

Minimization of response Time,

Clarity and Simplicity of the code,

Minimization of Hard –Coding i.e. (a generic approach was followed)

Minimization of amount of memory used.

Various types of errors were discovered while debugging the modules. These ranged from errors to failure to account for various processing cases. Proper documentation of each module was done by embedding comments in the executable portion of the code. To enhance the readability, comments, indentation, parenthesis, black spaces, blank lines and borders were around the blocks of comments. Care was taken to use descriptive names for table, field, modules, forms etc., the proper use of indentation, parenthesis and blank lines was also ensured during coding.

Testing of the Report Generation module was carried out to find out the response time of the system for the generating reports. To make the response time negligible

**SYSTEM MAINTENANCE**

Maintenance is a fact of life in the development of information systems. However the making of changes & adjustments don’t necessarily signal correction of error or the occurrence of problems.

Among the most frequent changes requested by end users is the addition of information to a report format. Information requirements may be revised as the result of system usage or changing operational needs. Perhaps oversights that occurred during the development process need to be corrected.

Often, the need arises to capture additional data for storage in a database or perhaps when it is necessary to add error detection features to prevent system users from inadvertently taking an unwanted action.

These are the realities of application maintenance, when they occur, however they are an indication that the system is being used & that it is serving a useful function rather than being ‘’shelved’’ by the intended, such as :

More accurately defining the user’s requirements during system development.

Assembling better systems documentation.

Making better use of existing tools & techniques.

Including proper comments.

**SYSTEM DOCUMENTATION**

Documentation is the process of collecting, organizing, storing and maintaining on paper (or some relatively permanently medium) a complete record of why application were developed, for whom, what function they perform, and how these functions are carried out. Thus it provides an explanation of how a system operates & what feature characterize models & algorithms. Producing documentation is an important – but often neglected – activity of programmers.

Documentation of the system always reduces maintenance costs and makes maintenance much more easier. Documentation can be broadly classified as :

INTERNAL DOCUMENTATION

EXTERNAL DOCUMENTATION

**INTERNAL DOCUMENTATION**

Internal documentation in the code can be extremely useful in enhancing the understandability of programs. Internal documentation of programs is done by the use of comments. All the languages provide means of writing comments in the programs. Comments are textual statements that are meant for the reader and are not executed. Comments, if properly written, and if kept consistent with the code, can be invaluable during maintenance.

The purpose of comments is not to explain the internal logic of the program – the program itself is the best documentation for the details of logic. The comments should explain what the code is done is doing, and not how it is done it.

Comments should be provided for the block of code, particularly those parts of code which are hard to follow.

Providing comments for module is most useful as module form the unit testing, compiling, verification and modification. Comments for a module are often called prologue for the module. It is best to standardize the structure of the prologue of the module. It is desirable that prologue contains the following information:

* Module functionality or what the module is doing
* Parameters are their purpose
* Assumptions about the input, if any
* Global variable accessed and or modified in the module
* As explanation of parameters (whether they are input only, output only or both input and output, why they are needed by the module and how the parameters are modified ) can be quite useful during maintenance.

In addition to that given above, often information can be included, depending on the local coding standards. Example includes:

* Name of the author,
* Date of compilation
* Last date of modification.

Note that if the module is modified, then the prologue should also be modified, if necessary. A prologue that is inconsistent with the internal logic of the module is prologue worse than having no prologue at all.

Apart from this comment lines have been inserted wherever it was felt that they were necessary. Moreover meaningful variable names/constants have been assigned to different variable/constants used in the programs.

**EXTERNAL DOCUMENTATION**

The external documentation of the system contains all the relevant information about the system which is not included within the programs. The external documentation typically explains:

What is the basic purpose of the system?

Why the system was developed?

The platform on which the system will run.

The various trouble shooting mechanisms.

The operation’s guide.

**DATA FLOW DIAGRAM**

# 0 Level DFD

LOGIN

STUDENT RECORD

ADMIN

ATTENDANCE

FACULTY

STUDENT

RESULT ATTENDANCE

# 

# 1 Level DFD

STUDENT DETAIL

FEE DETAIL

STUDENT ATTANDCE

STUDENT MARKS

**2nd level DFD**

Student

Registration Detail

Register

Result info

Issue/Search

Labs detail

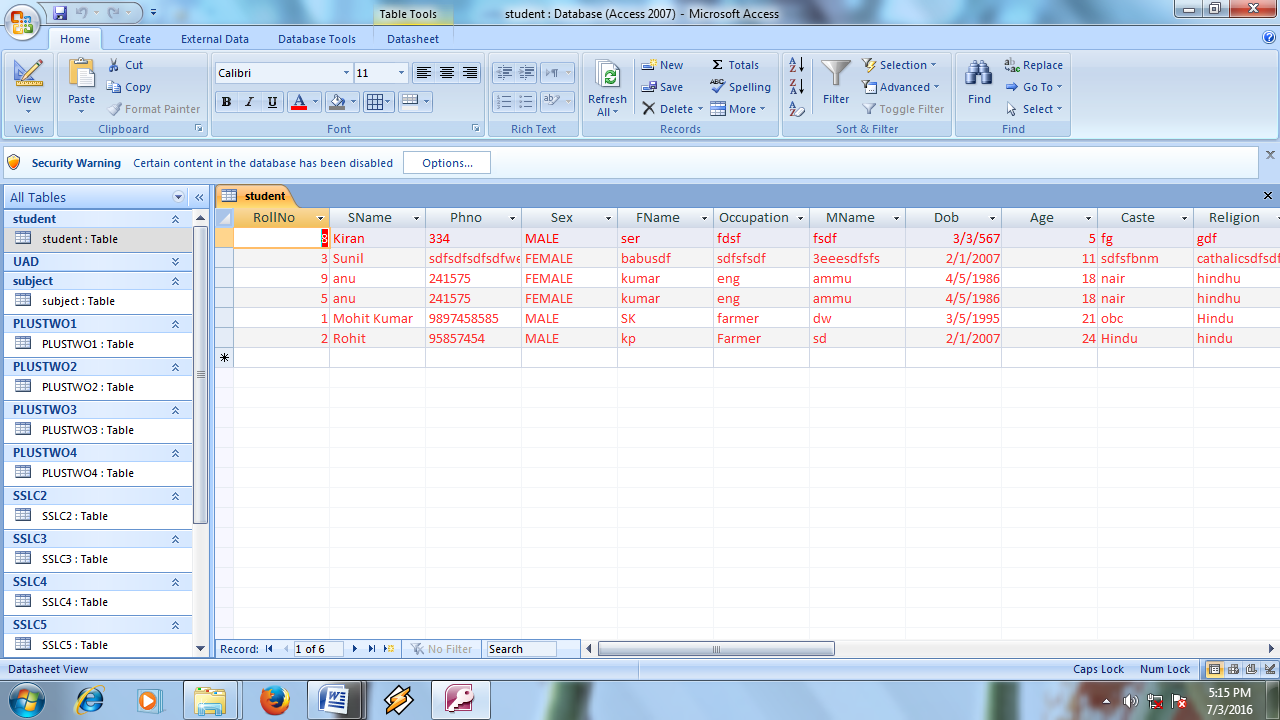
Library

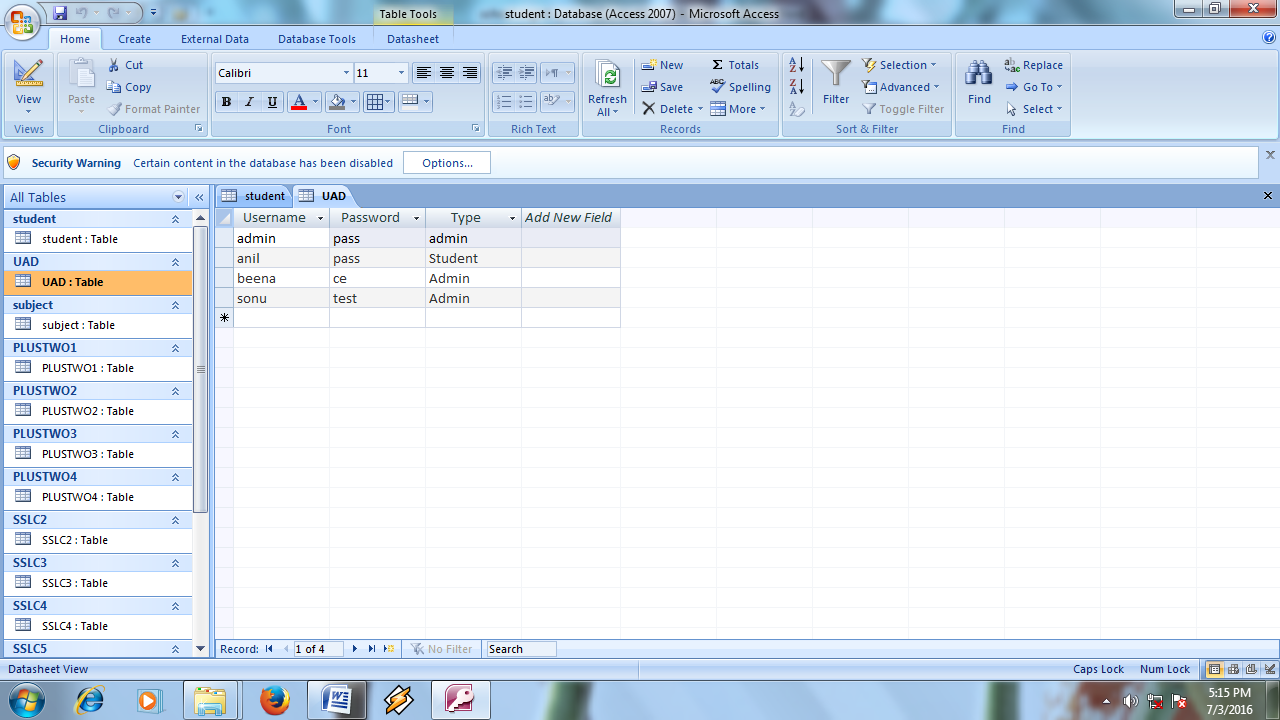
Add/delete

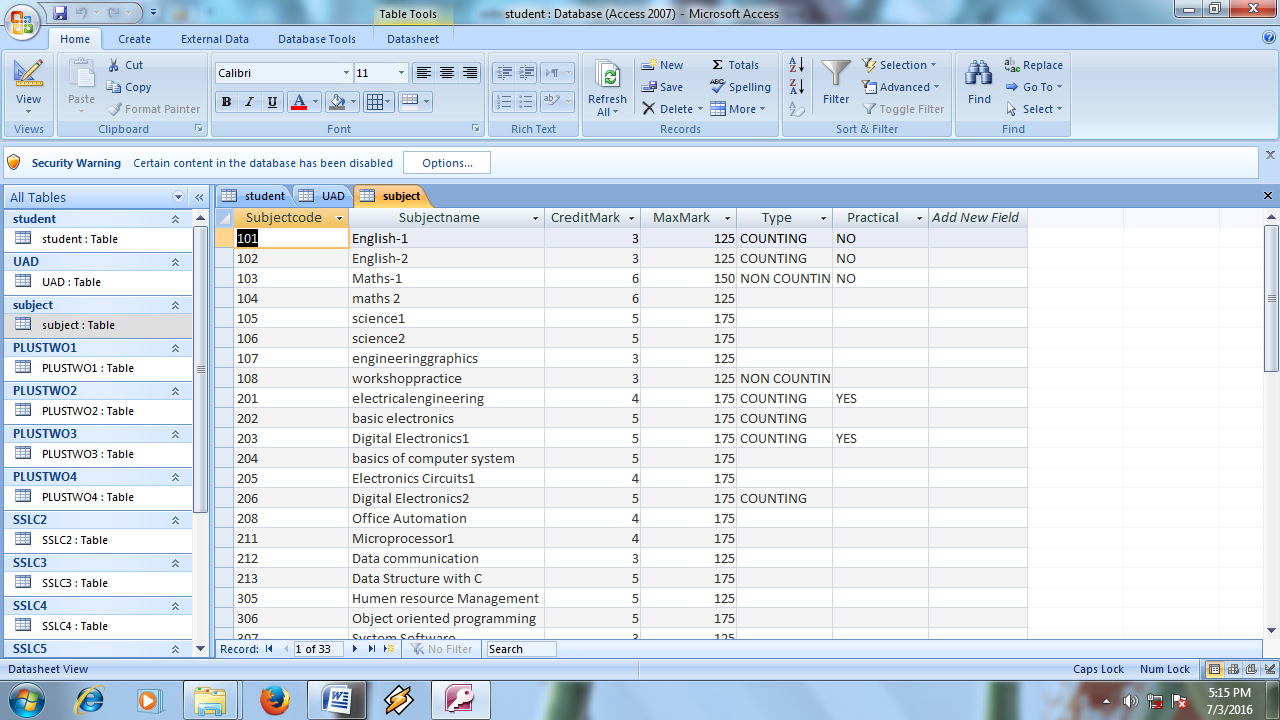
Marks info

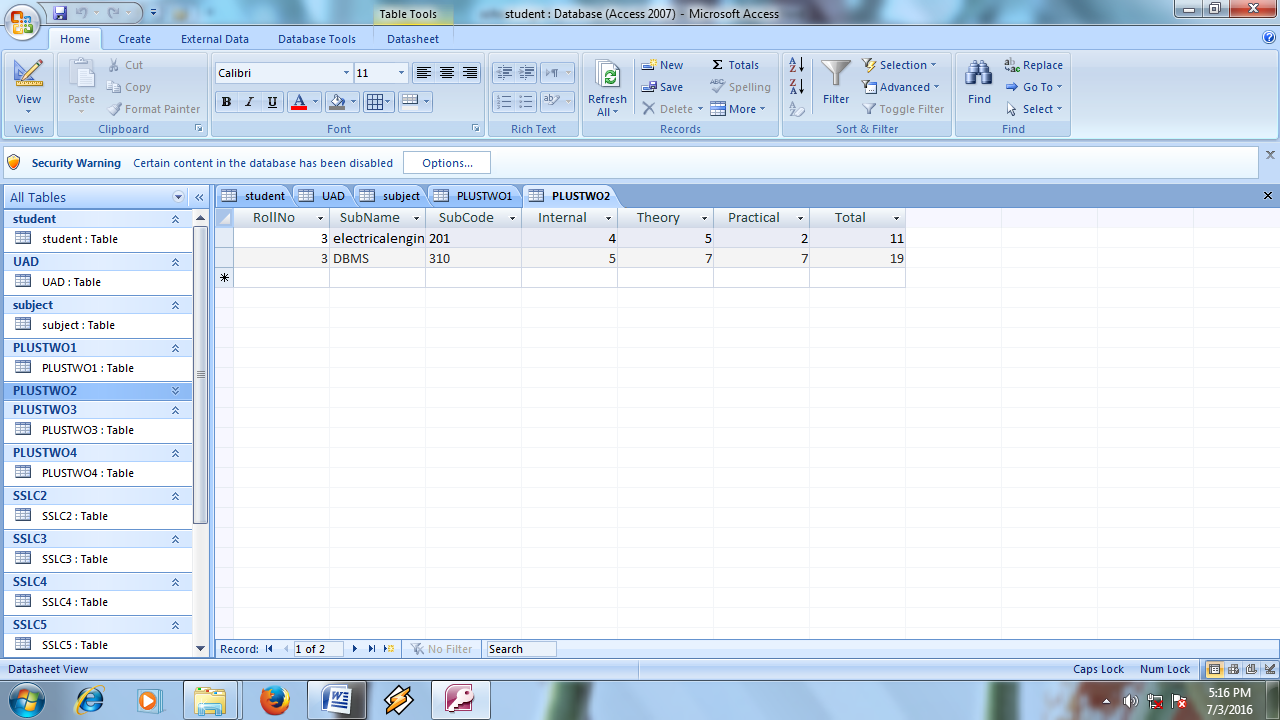
Attendance Details

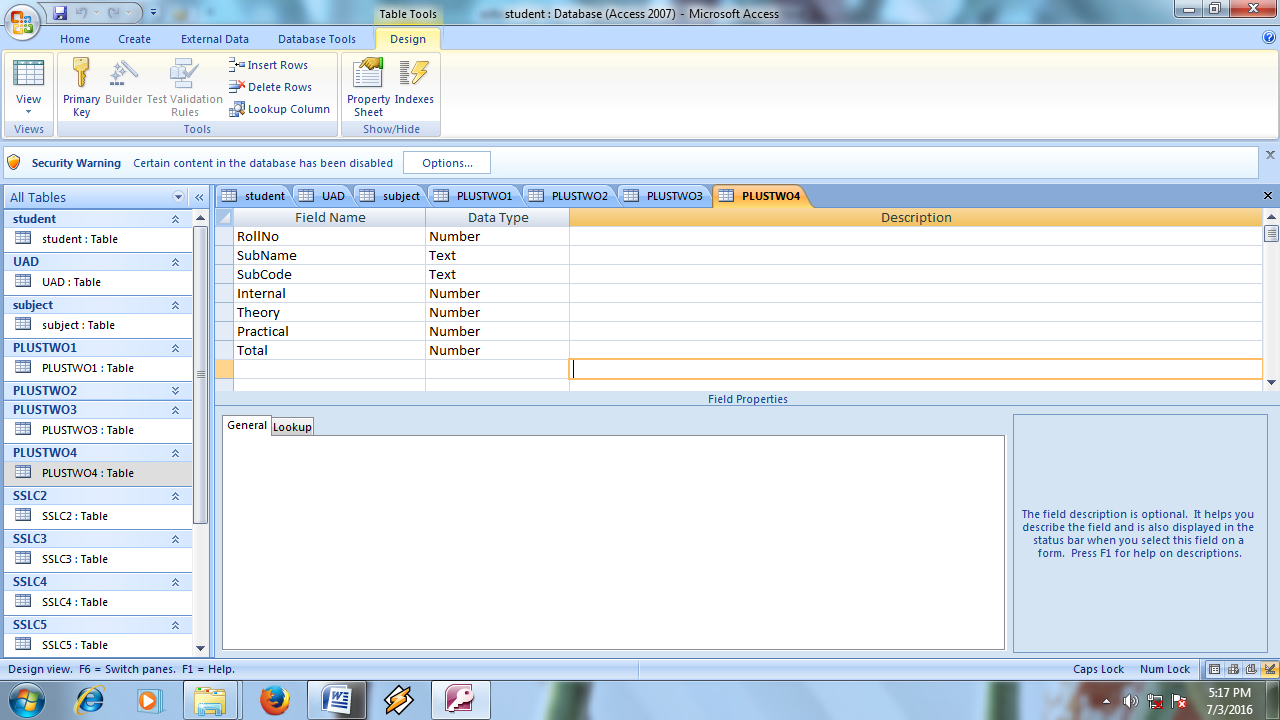
DATA BASE DESIGN TABLE

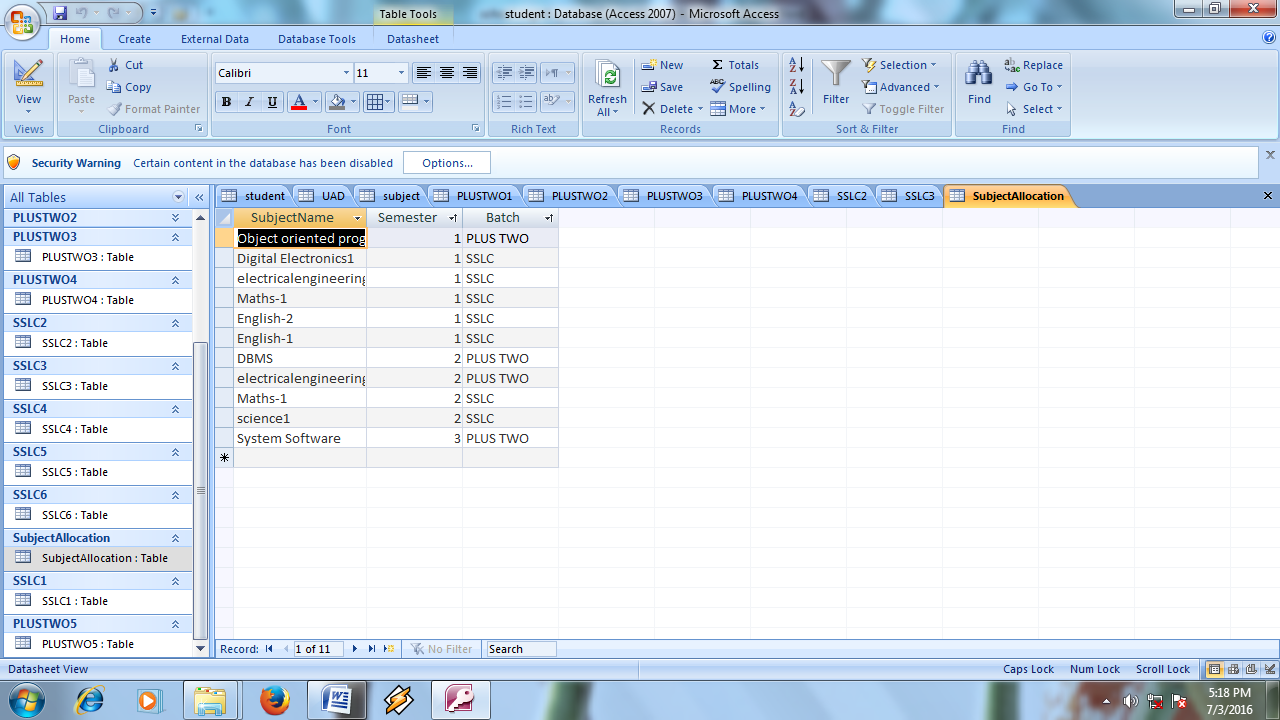


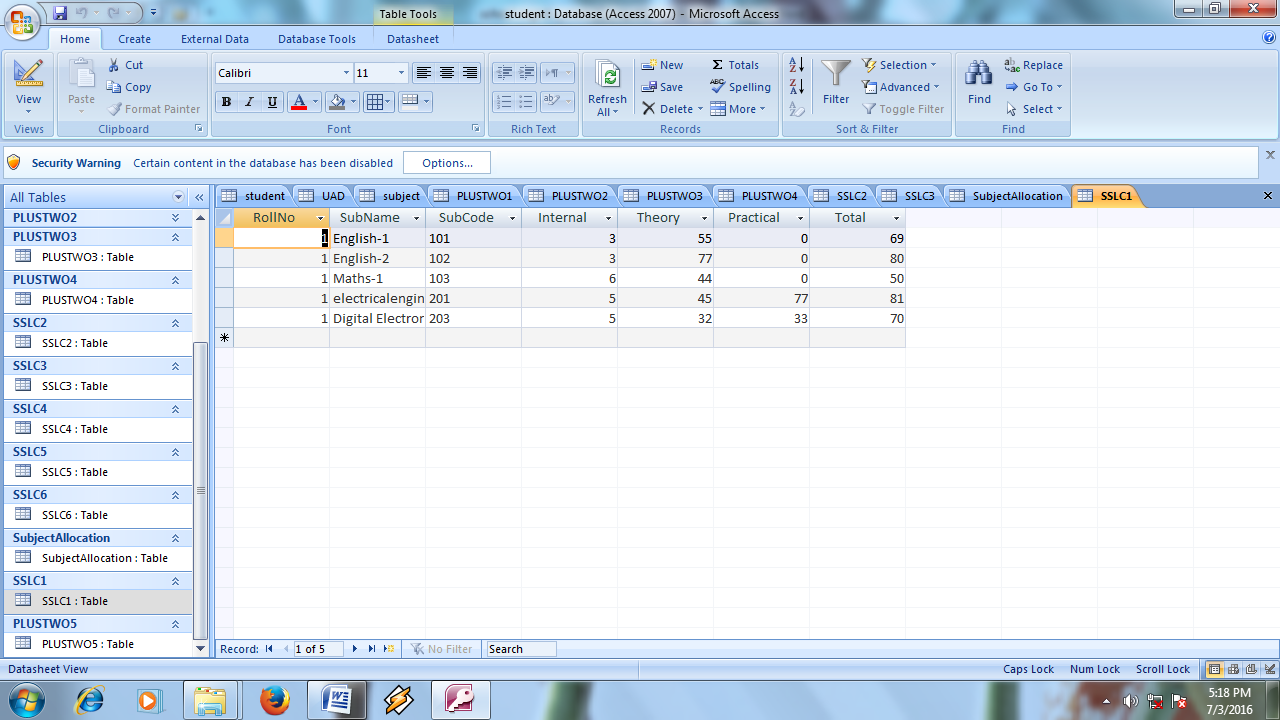


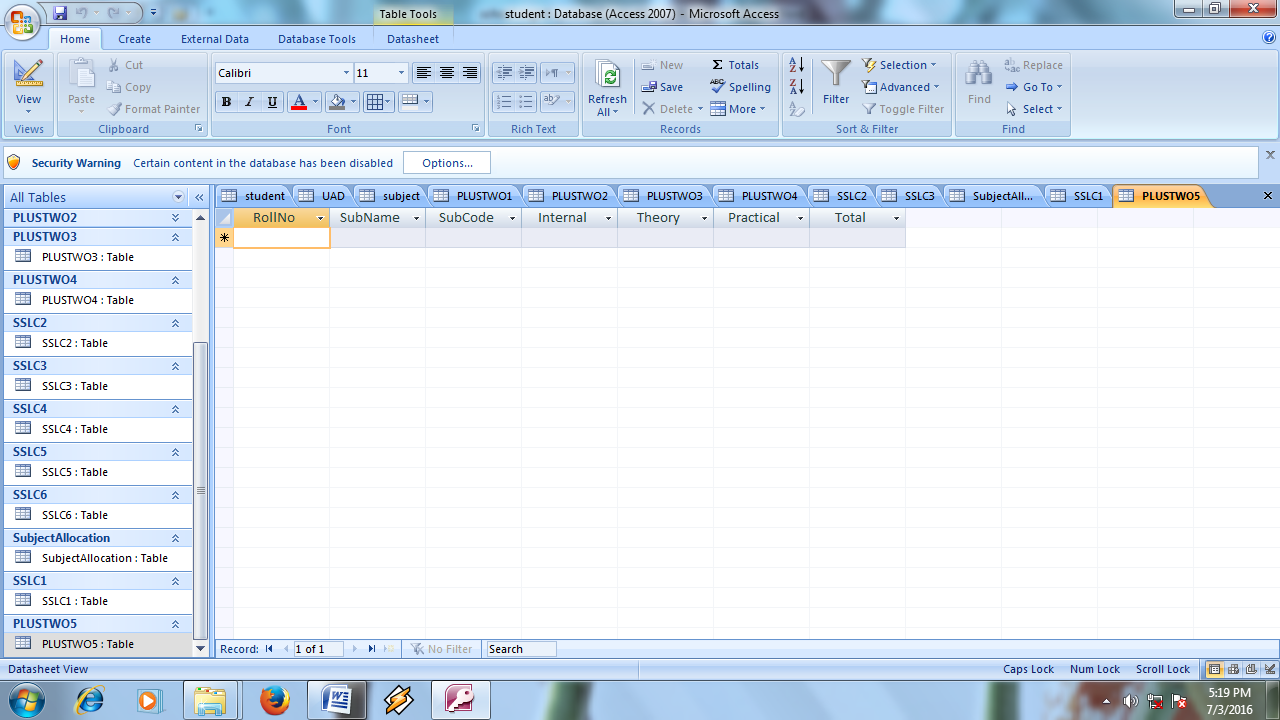








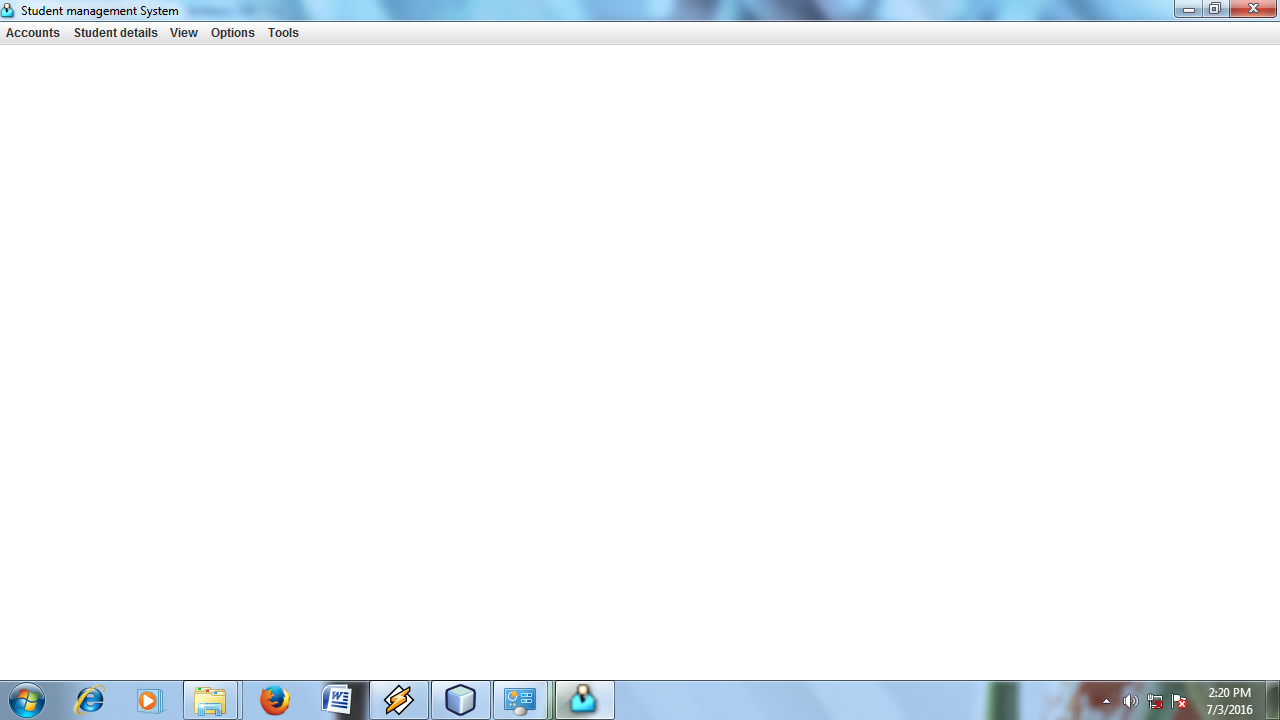




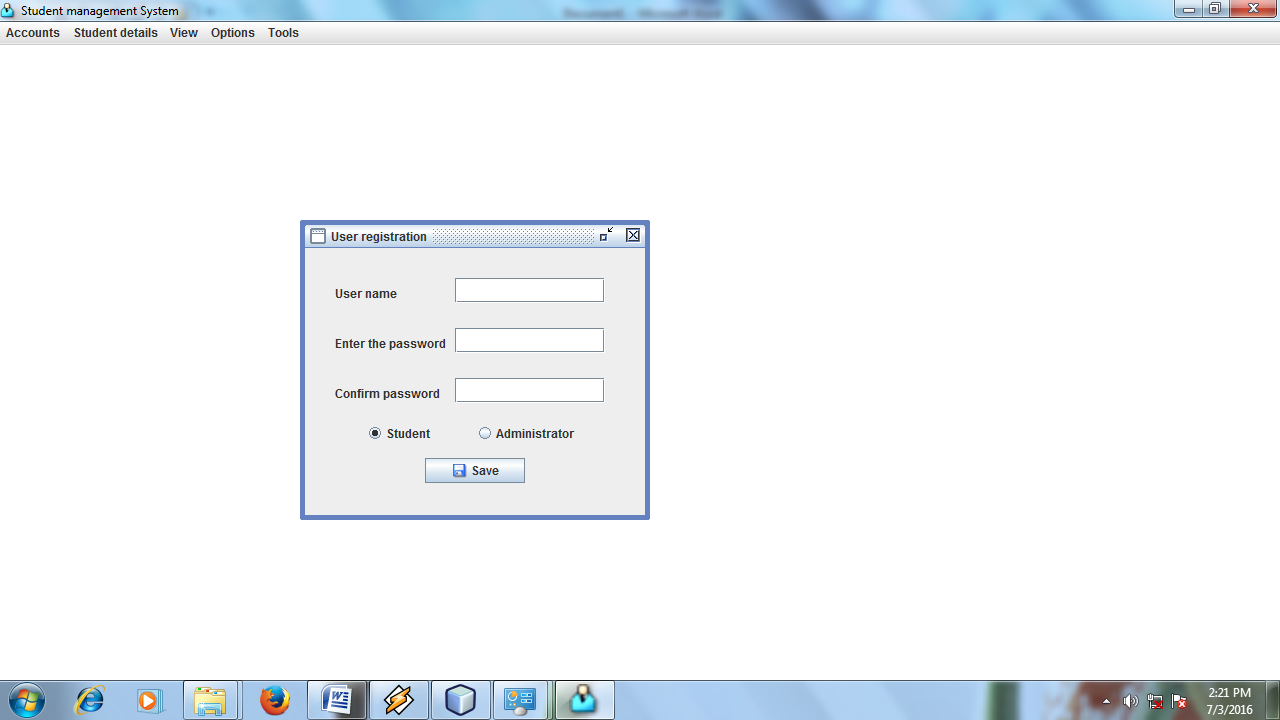
MDI FORM

IN

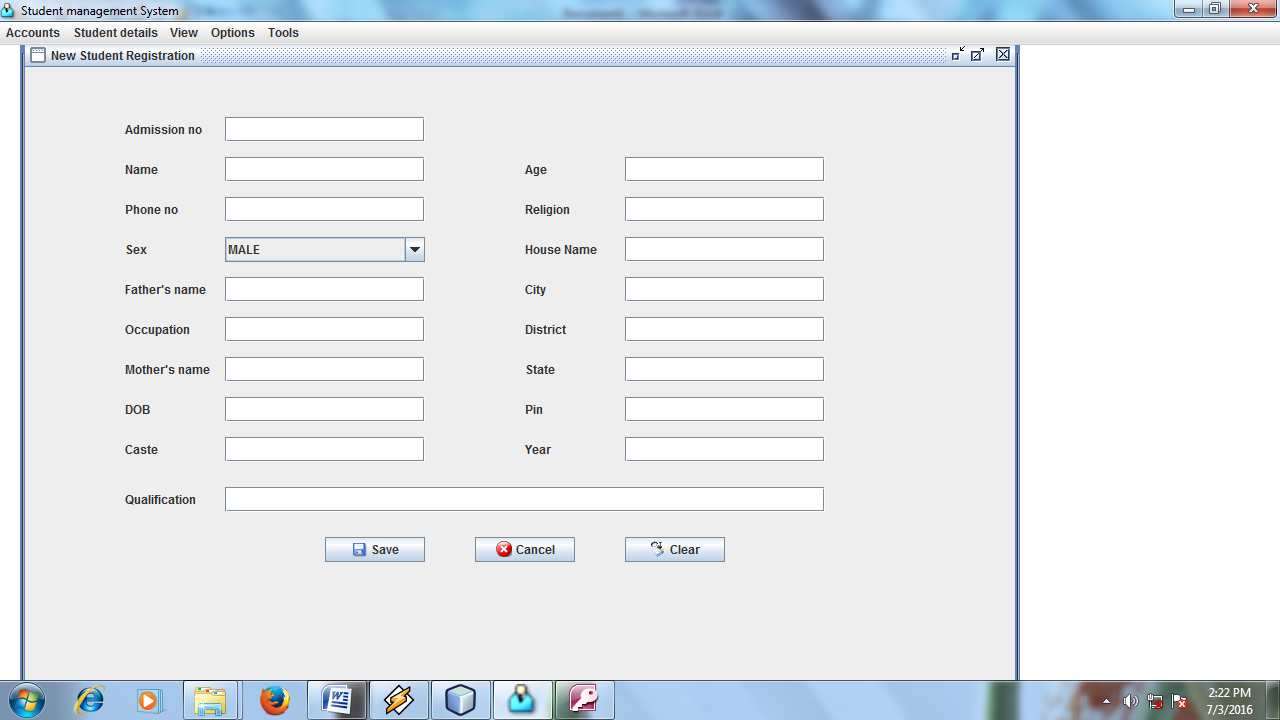
STUDENT RECORD MANAGEMENT



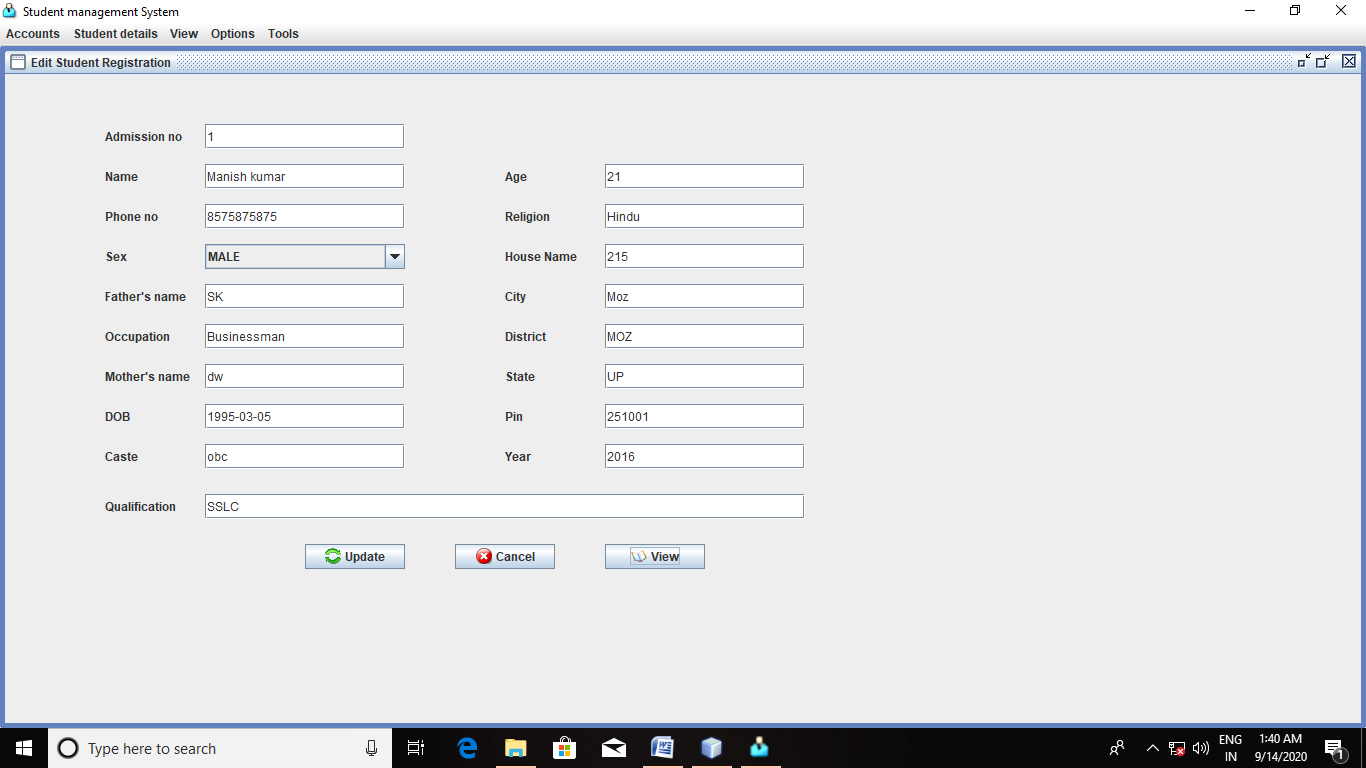
**USER REGISTRATION**



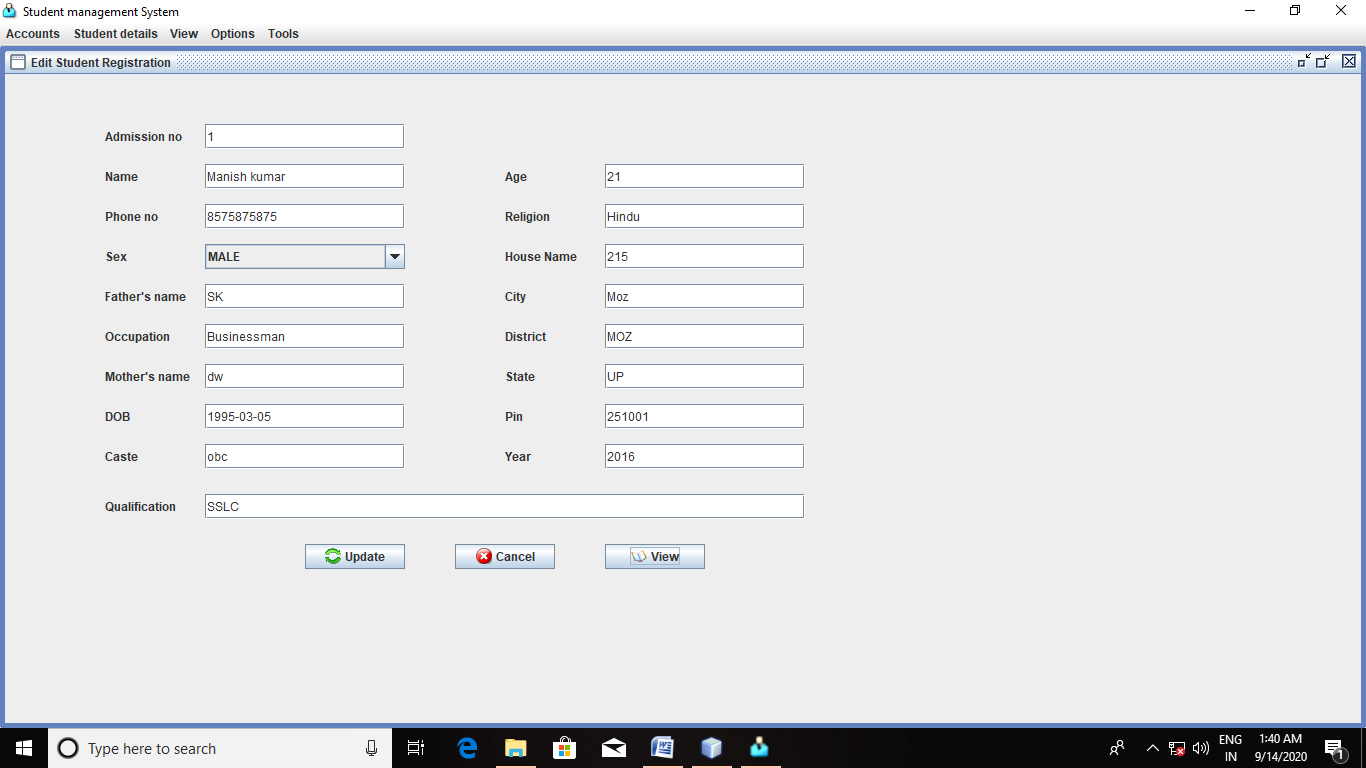
**NEW STUDENT REGISTRATION**



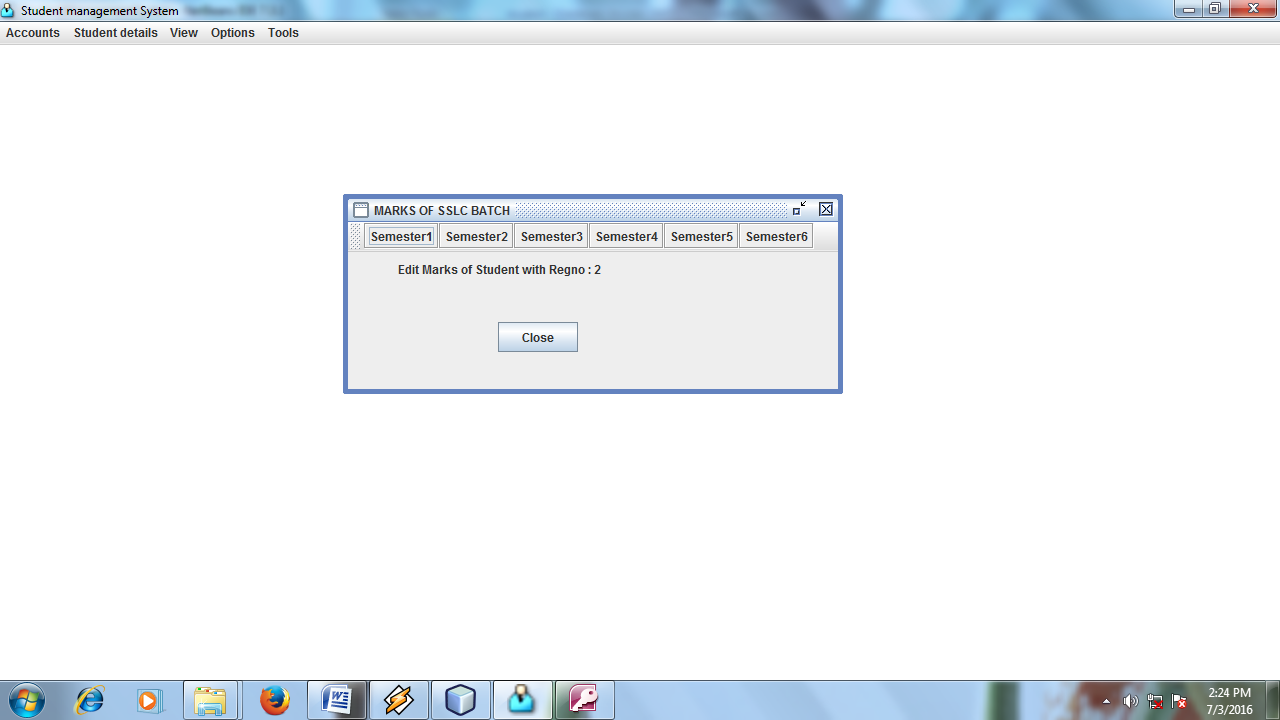
**EDIT STUDENT REGISTRATION**



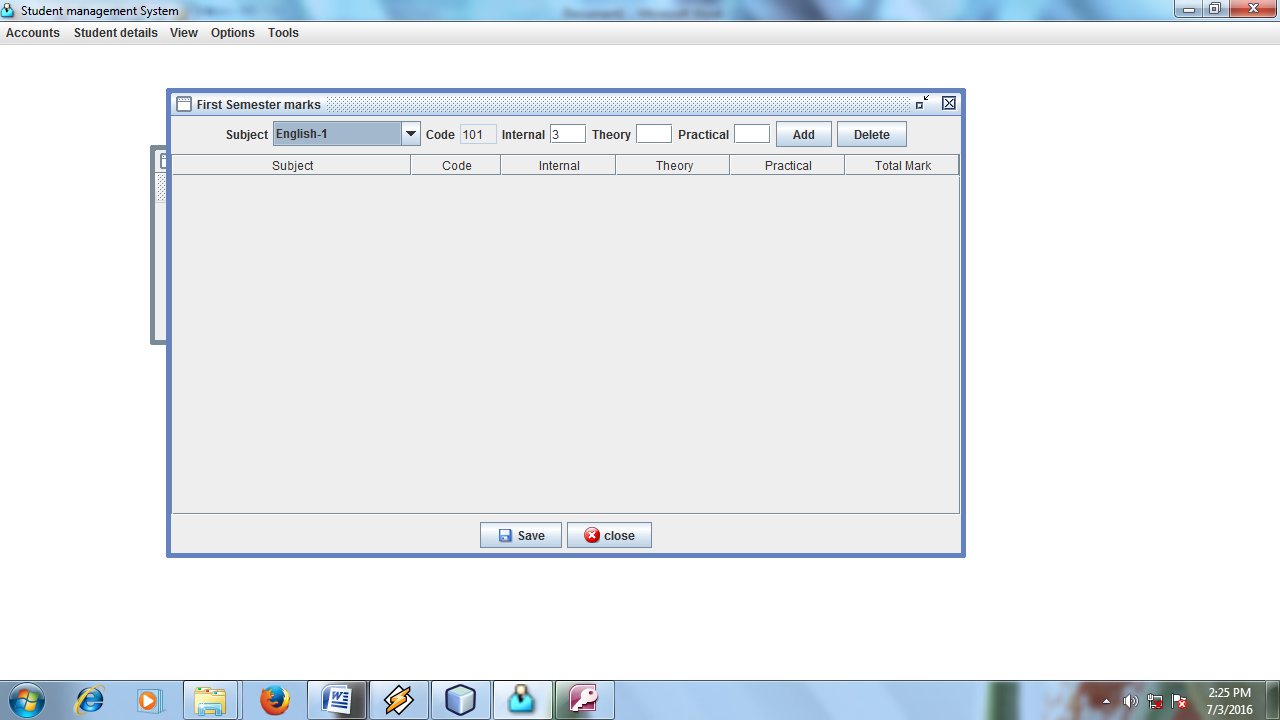
**DELETE RECORD**



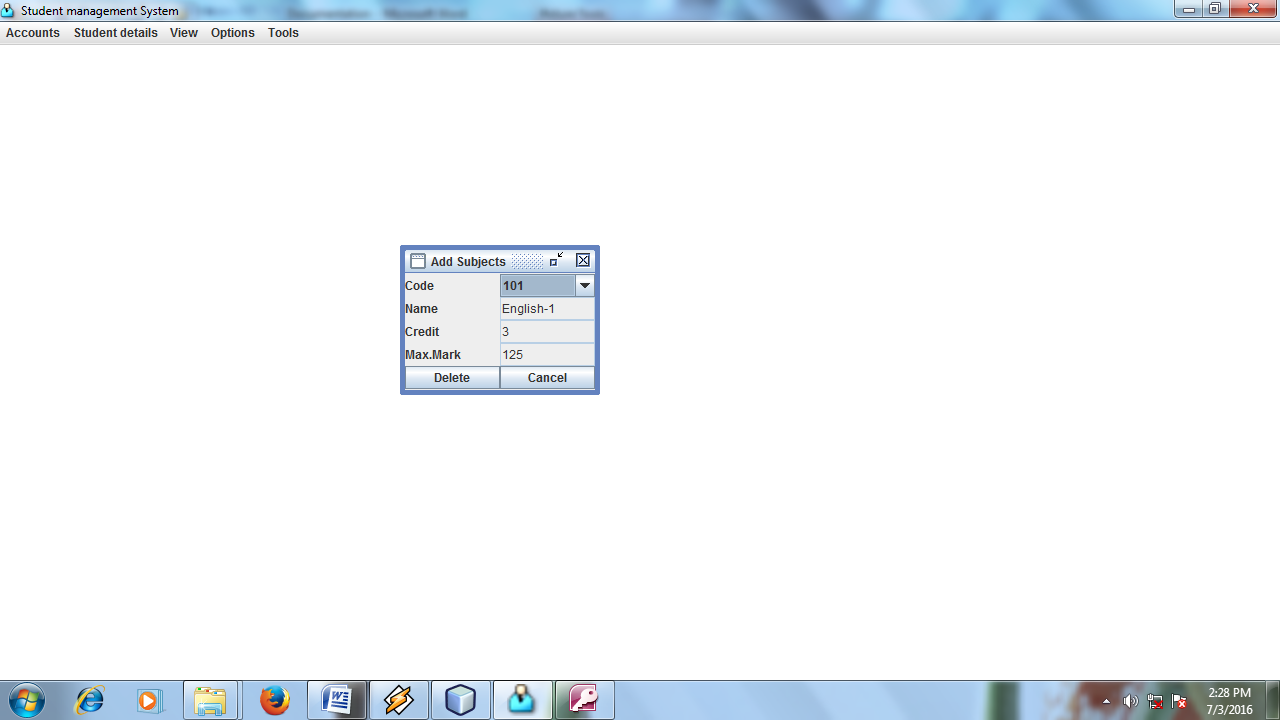
**EDIT MARKS**

****

**SEMESTER MARKS**

****

**ADD NEW SUBJECTS**

****

## What is Java?

Java is a **programming language** and a **platform**.

**Platform** Any hardware or software environment in which a program runs, known as a platform. Since Java has its own Runtime Environment (JRE) and API, it is called platform.

## Where it is used?

According to Sun, 3 billion devices run java. There are many devices where java is currently used. Some of them are as follows:

1. Desktop Applications such as acrobat reader, media player, antivirus etc.
2. Web Applications such as irctc.co.in, javatpoint.com etc.
3. Enterprise Applications such as banking applications.
4. Mobile
5. Embedded System
6. Smart Card
7. Robotics
8. Games etc.

History of Java

**Java history** is interesting to know. Java team members (also known as **Green Team**), initiated a revolutionary task to develop a language for digital devices such as set-top boxes, televisions etc.For the green team members, it was an advance concept at that time. But, it was suited for internet programming. Later, Java technology as incorporated by Netscape.

[**James Gosling**](http://en.wikipedia.org/wiki/James_Gosling)

Currently, Java is used in internet programming, mobile devices, games, e-business solutions etc. There are given the major points that describes the history of java.

1) **James Gosling**, **Mike Sheridan**, and **Patrick Naughton** initiated the Java language project in June 1991. The small team of sun engineers called **Green Team**.

2) Originally designed for small, embedded systems in electronic appliances like set-top boxes.

3) Firstly, it was called **"Greentalk"** by James Gosling and file extension was .gt.

4) After that, it was called **Oak** and was developed as a part of the Green project

Features of Java

There is given many features of java. They are also known as java buzzwords.

1. Simple
2. Object-Oriented
3. Platform independent
4. Secured
5. Robust
6. Architecture neutral
7. Portable
8. Dynamic
9. Interpreted
10. High Performance
11. Multithreaded
12. Distributed

Object-oriented

|  |
| --- |
| Object-oriented means we organize our software as a combination of different types of objects that incorporates both data and behaviour. |
| Object-oriented programming(OOPs) is a methodology that simplify software development and maintenance by providing some rules. |
| Basic concepts of OOPs are: |
| 1. Object 2. Class 3. Inheritance 4. Polymorphism 5. Abstraction 6. Encapsulation |

Platform Independent

|  |
| --- |
| A platform is the hardware or software environment in which a program runs. There are two types of platforms software-based and hardware-based. Java provides software-based platform. The Java platform differs from most other platforms in the sense that it's a software-based platform that runs on top of other hardware-based platforms.It has two components:   1. Runtime Environment 2. API(Application Programming Interface) |

Secured

|  |
| --- |
| Java is secured because: |
| * No explicit pointer * Programs run inside virtual machine sandbox |

Robust

|  |
| --- |
| Robust simply means strong. Java uses strong memory management. There are lack of pointers that avoids security problem. There is automatic garbage collection in java. There is exception handling and type checking mechanism in java. All these points makes java robust. |

Architecture-neutral

|  |
| --- |
| There is no implementation dependent features e.g. size of primitive types is set. |

Portable

|  |
| --- |
| We may carry the java bytecode to any platform. |

High-performance

|  |
| --- |
| Java is faster than traditional interpretation since byte code is "close" to native code still somewhat slower than a compiled language (e.g., C++) |

Distributed

|  |
| --- |
| We can create distributed applications in java. RMI and EJB are used for creating distributed applications. We may access files by calling the methods from any machine on the internet. |

Multi-threaded

A thread is like a separate program, executing concurrently. We can write Java programs that deal with many tasks at once by defining multiple threads. The main advantage of multi-threading is that it shares the same memory. Threads are important for multi-media, Web applications etc.

# java OOPs Concepts

Object Oriented Programming is a paradigm that provides many concepts such as **inheritance**, **data binding**,**polymorphism** etc.

**Simula** is considered as the first object-oriented programming language. The programming paradigm where everything is represented as an object, is known as truly object-oriented programming language.

**Smalltalk** is considered as the first truly object-oriented programming language.

## OOPs (Object Oriented Programming System)

**Object** means a real word entity such as pen, chair, table etc. **Object-Oriented Programming** is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some concepts:

* Object
* Class
* Inheritance
* Polymorphism
* Abstraction
* Encapsulation

**Object**

Any entity that has state and behavior is known as an object. For example: chair, pen, table, keyboard, bike etc. It can be physical and logical.

**Class**

Collection of objects is called class. It is a logical entity.

**Inheritance**

When one object acquires all the properties and behaviours of parent object i.e. known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

**Polymorphism**

When one task is performed by different ways i.e. known as polymorphism. For example: to convense the customer differently, to draw something e.g. shape or rectangle etc.

In java, we use method overloading and method overriding to achieve polymorphism.

Another example can be to speak something e.g. cat speaks meaw, dog barks woof etc.

**Abstraction**

Hiding internal details and showing functionality is known as abstraction. For example: phone call, we don't know the internal processing.

In java, we use abstract class and interface to achieve abstraction.

**Encapsulation**

Binding (or wrapping) code and data together into a single unit is known as encapsulation. For example: capsule, it is wrapped with different medicines.

A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.

**Object and Class in Java**

In this page, we will learn about java objects and classes. In object-oriented programming technique, we design a program using objects and classes.

Object is the physical as well as logical entity whereas class is the logical entity only.

**Object in Java**

An entity that has state and behavior is known as an object e.g. chair, bike, marker, pen, table, car etc. It can be physical or logical (tengible and intengible). The example of integible object is banking system.

**An object has three characteristics:**

* **state:** represents data (value) of an object.
* **behavior:** represents the behavior (functionality) of an object such as deposit, withdraw etc.
* **identity:** Object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. But,it is used internally by the JVM to identify each object uniquely.

|  |
| --- |
| For Example: Pen is an object. Its name is Reynolds, color is white etc. known as its state. It is used to write, so writing is its behavior. |

|  |
| --- |
| **Object is an instance of a class.** Class is a template or blueprint from which objects are created. So object is the instance(result) of a class. |

**Class in Java**

|  |
| --- |
| A class is a group of objects that has common properties. It is a template or blueprint from which objects are created. |

A class in java can contain:

* **data member**
* **method**
* **constructor**
* **block**
* **class and interface**

**Inheritance in Java**

**Inheritance** is a mechanism in which one object acquires all the properties and behaviours of parent object.

The idea behind inheritance is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you reuse (or inherit) methods and fields, and you add new methods and fields to adapt your new class to new situations.

Inheritance represents the **IS-A relationship**.

**Why use Inheritance?**

* For Method Overriding (So Runtime Polymorphism).
* For Code Reusability.

**Syntax of Inheritance**

1. **class** Subclass-name **extends** Superclass-name
2. {
3. //methods and fields
4. }

|  |
| --- |
| The keyword extends indicates that you are making a new class that derives from an existing class. In the terminology of Java, a class that is inherited is called a superclass. The new class is called a subclass. |

Types of Inheritance

On the basis of class, there can be three types of inheritance: single, multilevel and hierarchical.

Multiple and Hybrid is supported through interface only. We will learn about interfaces later.

# types of inheritance in javamultiple inheritance in javaInterface in Java

An **interface** is a blueprint of a class. It has static constants and abstract methods.The interface is **a mechanism to achieve fully abstraction** in java. There can be only abstract methods in the interface. It is used to achieve fully abstraction and multiple inheritance in Java.

Interface also **represents IS-A relationship**.

It cannot be instantiated just like abstract class.

## Why use Interface?

There are mainly three reasons to use interface. They are given below.

* It is used to achieve fully abstraction.
* By interface, we can support the functionality of multiple inheritance.
* It can be used to achieve loose coupling.

#### Understanding relationship between classes and interfaces

As shown in the figure given below, a class extends another class, an interface extends another interface but a **class implements an interface**.

## relationship between class and interface

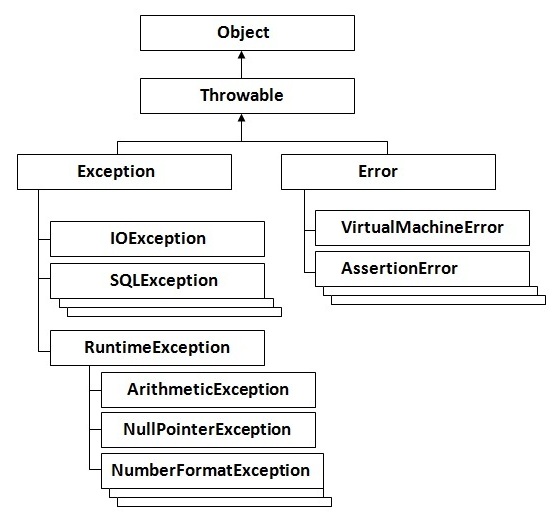
## Multiple inheritance in Java by interface

If a class implements multiple interfaces, or an interface extends multiple interfaces i.e. known as multiple inheritance.

# Package in Java

A **package** is a group of similar types of classes, interfaces and sub-packages.

Package can be categorized in two form, built-in package and user-defined package. There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.



Five keywords used in Exception handling:

|  |
| --- |
| 1. try 2. catch 3. finally 4. throw 5. throws |

# Multithreading in Java

|  |
| --- |
| Multithreading is a process of executing multiple threads simultaneously. |
| Thread is basically a lightweight subprocess, a smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking. But we use multithreading than mulitprocessing because threads share a common memory area. They don't allocate separate memory area so save memory, and context-switching between the threads takes less time than processes. |
| Multithreading is mostly used in games, animation etc. |

# Multitasking

|  |
| --- |
| Multitasking is a process of executing multiple tasks simultaneously. We use multitasking to utilize the CPU. Multitasking can be achieved by two ways:   * Process-based Multitasking(Multiprocessing) * Thread-based Multitasking(Multithreading) |

1)Process-based Multitasking (Multiprocessing)

|  |
| --- |
| * Each process have its own address in memory i.e. each process allocates separate memory area. * Process is heavyweight. * Cost of communication between the process is high. * Switching from one process to another require some time for saving and loading registers, memory maps, updating lists etc. |

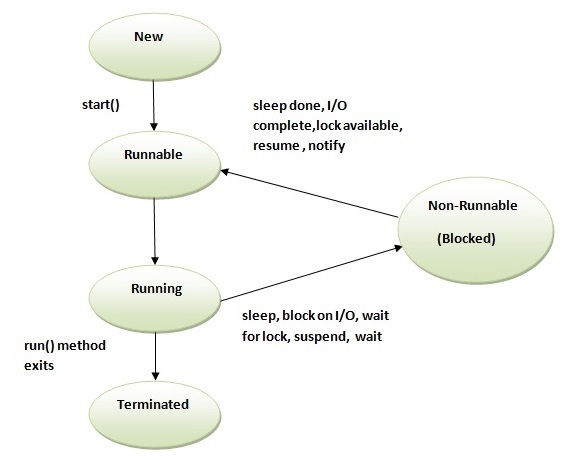
2)Thread-based Multitasking (Multithreading)

|  |
| --- |
| * Threads share the same address space. * Thread is lightweight. * Cost of communication between the thread is low. * **Note:**At least one process is required for each thread. |

# Life cycle of a Thread (Thread States)

A thread can be in one of the five states in the thread. According to sun, there is only 4 states new, runnable, non-runnable and terminated. There is no running state. But for better understanding the threads, we are explaining it in the 5 states. The life cycle of the thread is controlled by JVM. The thread states are as follows:

1. New
2. Runnable
3. Running
4. Non-Runnable (Blocked)
5. Terminated



|  |
| --- |
| 1)New  The thread is in new state if you create an instance of Thread class but before the invocation of start() method. |

2) Runnable

The thread is in runnable state after invocation of start() method, but the thread scheduler has not selected it to be the running thread.

3) Running

The thread is in running state if the thread scheduler has selected it.

4) Non-Runnable (Blocked)

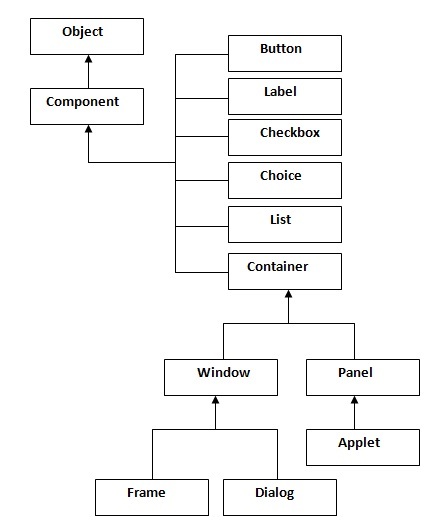
This is the state when the thread is still alive, but is currently not eligible to run.

5) Terminated

A thread is in terminated or dead state when its run() method exits.

Abstract Windowing Toolkit (AWT):

AWT Container Hierarchy:



Container:

|  |
| --- |
| The Container is a component in AWT that can contain another components like buttons, textfields, labels etc. The classes that extends Container class are known as container. |

Window:

|  |
| --- |
| The window is the container that have no borders and menubars. You must use frame, dialog or another window for creating a window. |

Panel:

|  |
| --- |
| The Panel is the container that doesn't contain title bar and MenuBars. It can have other components like button, textfield etc. |

Frame:

|  |
| --- |
| The Frame is the container that contain title bar and can have MenuBars. It can have other components like button, textfield etc. |

Commonly used Methods of Component class:

|  |
| --- |
| **1)public void add(Component c)** |
| **2)public void setSize(int width,int height)** |
| **3)public void setLayout(LayoutManager m)** |
| **4)public void setVisible(boolean)** |

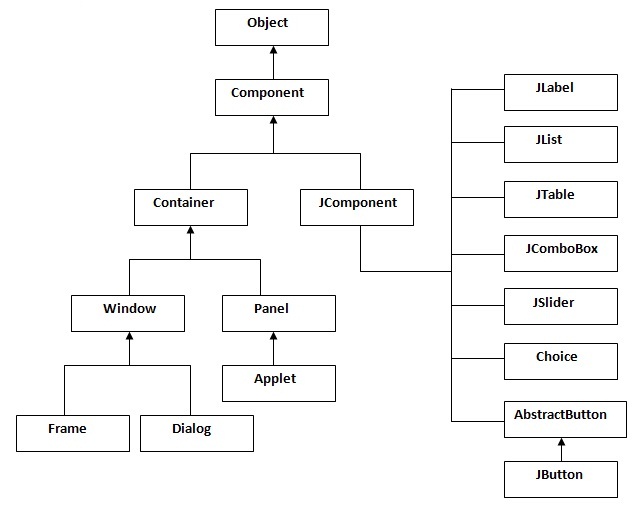
Swing (Graphics Programming in java):

|  |
| --- |
| Swing is a part of JFC (Java Foundation Classes) that is used to create GUI application. It is built on the top of AWT and entirely written in java. |

Advantage of Swing over AWT:

|  |
| --- |
| There are many advantages of Swing over AWT. They are as follows:   * Swing components are Plateform independent. * It is lightweight. * It supports pluggable look and feel. * It has more powerful components like tables, lists, scroll panes, color chooser, tabbed pane etc. |

Hierarchy of swing:



Commonly used Methods of Component class:

|  |
| --- |
| **1)public void add(Component c)** |
| **2)public void setSize(int width,int height)** |
| **3)public void setLayout(LayoutManager m)** |
| **4)public void setVisible(boolean)** |
| JDBC  This JDBC tutorial covers all the topics of JDBC with the simple examples. JDBC is a Java API that is used to connect and execute query to the database. JDBC API uses jdbc drivers to connects to the database. | |

Why use JDBC?

|  |
| --- |
| Before JDBC, ODBC API was used to connect and execute query to the database. But ODBC API uses ODBC driver that is written in C language which is plateform dependent and unsecured. That is why Sun Microsystem has defined its own API (JDBC API) that uses JDBC driver written in Java language. |

### What is API?

API (Application programming interface) is a document that contains description of all the features of a product or software. It represents classes and interfaces that software programs can follow to communicate with each other. An API can be created for applications, libraries, operating systems, etc

JDBC Driver

|  |
| --- |
| JDBC Driver is a software component that enables java application to interact with the database.There are 4 types of JDBC drivers:   1. JDBC-ODBC bridge driver 2. Native-API driver (partially java driver) 3. Network Protocol driver (fully java driver) 4. Thin driver (fully java driver) |

1) JDBC-ODBC bridge driver

|  |
| --- |
| The JDBC-ODBC bridge driver uses ODBC driver to connect to the database. The JDBC-ODBC bridge driver converts JDBC method calls into the ODBC function calls. This is now discouraged because of thin driver. |

**LOGIN FORM**

import java.awt.Container;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.Statement;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPasswordField;

import javax.swing.JTextField;

public class Login extends JFrame implements ActionListener {

Container c = getContentPane();

private JButton btnLogin, btnCancel;

private JLabel lblUName, lblPasswd;

private JTextField txtUName;

private JPasswordField txtPasswd;

public Login() {

super("Login ...");

this.setSize(350, 200);

this.setLayout(null);

this.setResizable(false);

this.setLocation((Settings.getScreenSize().width / 2) - 175, (Settings.getScreenSize().height / 2) - 150);

this.setDefaultCloseOperation(EXIT\_ON\_CLOSE);

lblUName = new JLabel("Username");

lblPasswd = new JLabel("Password");

txtUName = new JTextField();

txtPasswd = new JPasswordField();

btnLogin = new JButton("Login", new ImageIcon(ClassLoader.getSystemResource("Images/login.png")));

btnCancel = new JButton("Cancel",new ImageIcon(ClassLoader.getSystemResource("Images/cancel.png")));

lblUName.setBounds(50, 40, 140, 25);

txtUName.setBounds(150, 40, 130, 25);

lblPasswd.setBounds(50, 80, 140, 25);

txtPasswd.setBounds(150, 80, 130, 25);

btnLogin.setBounds(50, 120, 100, 25);

btnCancel.setBounds(180, 120, 100, 25);

btnLogin.addActionListener(this);

btnCancel.addActionListener(this);

this.add(lblUName);

this.add(lblPasswd);

this.add(txtUName);

this.add(txtPasswd);

this.add(btnLogin);

this.add(btnCancel);

}//constructor closed

public void actionPerformed(ActionEvent e) {

if (e.getSource() == btnLogin) {

try {

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

Connection con = DriverManager.getConnection("jdbc:odbc:student");

try {

Statement st = con.createStatement();

ResultSet rs = st.executeQuery("SELECT \* FROM UAD WHERE Username='" + txtUName.getText() +

"' and Password='" + txtPasswd.getText() + "'");

if (rs.next()) {

if (rs.getString(3).equals("Student")) {

userMDI frm = new userMDI();

frm.setVisible(true);

} else {

new frmAdminMDI().setVisible(true);

}

this.dispose();

}else{

JOptionPane.showMessageDialog(null,"Invalid username or password","Invalid",JOptionPane.ERROR\_MESSAGE);

}

con.close();

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Invalid username or password", "Invalid", JOptionPane.ERROR\_MESSAGE);

txtUName.setText("");

txtPasswd.setText("");

}//inner try catch closed

} catch (Exception x) {

JOptionPane.showMessageDialog(null, "Unable to connect to the database", "Connection error", JOptionPane.ERROR\_MESSAGE);

}//outer try catch closed

}//if closed

if (e.getSource() == btnCancel) {

System.exit(0);

}//if closed

}//actionPerformed() closed

public static void main(String args[]) {

new Login().setVisible(true);

}

}//class closed

}

**MDI FORM**

class HomePage extends Frame implements ActionListener, WindowListener

{

MenuBar mb;

Menu mAdmin,mStudentDetails,mStatus,mAboutUs,mLogout;

MenuItem miCreateUser,miChangePswd;

MenuItem miAddStd,miModifyStd,miRemoveStd;

MenuItem miStatusRollnoWise,miStatusClassWise;

MenuItem miAboutUs;

MenuItem miLogout;

HomePage()

{

mb = new MenuBar();

this.setMenuBar(mb);

mAdmin = new Menu("Administrator");

mStudentDetails = new Menu("Student Details");

mStatus = new Menu("Students Status");

mAboutUs = new Menu("About Us");

mLogout = new Menu("Logout");

mb.add(mAdmin);

mb.add(mStudentDetails);

mb.add(mStatus);

mb.add(mAboutUs);

mb.add(mLogout);

miCreateUser = new MenuItem("Create New User");

miChangePswd = new MenuItem("Change Password");

mAdmin.add(miCreateUser);

mAdmin.add(miChangePswd);

miCreateUser.addActionListener(this);

miChangePswd.addActionListener(this);

miAddStd = new MenuItem("Add New Student");

miModifyStd = new MenuItem("Modify Student Data");

miRemoveStd = new MenuItem("Remove Student Data");

mStudentDetails.add(miAddStd);

mStudentDetails.add(miModifyStd);

mStudentDetails.add(miRemoveStd);

miAddStd.addActionListener(this);

miModifyStd.addActionListener(this);

miRemoveStd.addActionListener(this);

miStatusRollnoWise = new MenuItem("RollNo-Wise");

mStatus.add(miStatusRollnoWise);

miStatusRollnoWise.addActionListener(this);

miStatusClassWise = new MenuItem("Class-Wise");

mStatus.add(miStatusClassWise);

miStatusClassWise.addActionListener(this);

miAboutUs = new MenuItem("About Us");

mAboutUs.add(miAboutUs);

miAboutUs.addActionListener(this);

miLogout = new MenuItem("Logout");

mLogout.add(miLogout);

miLogout.addActionListener(this);

addWindowListener(this);

pack();

setLocationRelativeTo(null);

}

public void actionPerformed(ActionEvent e)

{

HandelWindows hw = new HandelWindows();

if(e.getSource()==miAboutUs)

{

try

{

hw.makeAboutUs();

}

catch(Exception e1)

{

}

}

else if(e.getSource()==miCreateUser)

{

try

{

hw.makeNewAccount();

}

hw.makeLogin();

}

catch(Exception e1)

{

}

}

}

public void windowClosing(WindowEvent e)

{

System.exit(0);

}

public void windowClosed(WindowEvent e){}

public void windowOpened(WindowEvent e){}

public void windowIconified(WindowEvent e){}

public void windowDeiconified(WindowEvent e){}

public void windowActivated(WindowEvent e){}

public void windowDeactivated(WindowEvent e){}

}

record++;

}

rs = st.executeQuery("select \* from Personal where Class='" +

chClass.getSelectedItem().toString() + "'");

Object header[] = {"Roll No","Stream","Name","Father's

Name","Age","Gender","Address","Class"};

Object data[][] = new Object[record][8];

record=0;

i=0;

while(rs.next())

{

data[i][0] = rs.getInt(1);

data[i][1] = rs.getString(2);

data[i][2] = rs.getString(3);

data[i][3] = rs.getString(4);

data[i][4] = rs.getString(5);

data[i][5] = rs.getString(6);

data[i][6] = rs.getString(7);

data[i][7] = rs.getString(8);

i++;

}

jt = new JTable(data,header);

jt.setAutoCreateRowSorter(true);

jsp = new JScrollPane(jt);

add(jsp);

jsp.setBounds(10,150,730,430);

jt.setEnabled(false);

}

}

else if(cbStream.getState()==true)

{

if(chStream.isEnabled()==false)

{

try

{

st = conn.createStatement();

rs = st.executeQuery("select \* from Personal where Class='" +

chClass.getSelectedItem().toString() + "'");

while(rs.next())

{

record++;

**HANDLING WINDOWS**

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.Statement;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import javax.swing.ButtonGroup;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JComboBox;

import javax.swing.JInternalFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JRadioButton;

public class Edituser extends JInternalFrame implements ActionListener {

private JRadioButton rdoStudent;

private JRadioButton rdoAdmin;

private JLabel lblUName;

private JComboBox cmbUName;

private JButton btnUpdate;

private ButtonGroup group;

public Edituser() {

super("Edit User", false, true, false, true);

setBounds(350, 200, 300, 200);

setLayout(null);

rdoStudent = new JRadioButton("Student");

rdoAdmin = new JRadioButton("Administrator");

lblUName = new JLabel("User name");

cmbUName = new JComboBox();

btnUpdate = new JButton("Update", new ImageIcon(ClassLoader.getSystemResource("Images/update.png")));

group = new ButtonGroup();

lblUName.setBounds(30, 30, 100, 30);

cmbUName.setBounds(140, 30, 100, 25);

rdoStudent.setBounds(30, 70, 100, 30);

rdoAdmin.setBounds(140, 70, 150, 30);

btnUpdate.setBounds(100, 110, 100, 25);

add(lblUName);

add(cmbUName);

group.add(rdoAdmin);

group.add(rdoStudent);

add(rdoAdmin);

add(rdoStudent);

add(btnUpdate);

btnUpdate.addActionListener(this);

rdoStudent.setSelected(true);

try {

String name;

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

Connection con = DriverManager.getConnection("jdbc:odbc:student");

Statement st = con.createStatement();

ResultSet rs = st.executeQuery("select Username from UAD");

while (rs.next()) {

name = rs.getString(1);

if (!name.equalsIgnoreCase("admin")) {

cmbUName.addItem(name);

}

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Error while loading users list, Closing window");

this.dispose();

}//try catch clsoed

}//constructor closed

public void actionPerformed(ActionEvent e) {

try {

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

Connection con = DriverManager.getConnection("jdbc:odbc:student");

Statement st = con.createStatement();

try {

String Type;

if (e.getSource() == btnUpdate) {

if (rdoStudent.isSelected()) {

Type = "Student";

} else {

Type = "Admin";

}

String sql = "update UAD set Type ='" + Type + "' where Username='" + cmbUName.getSelectedItem().toString() + "'";

st.executeUpdate(sql);

JOptionPane.showMessageDialog(null, "Database successfully updated", "Success", JOptionPane.INFORMATION\_MESSAGE);

this.dispose();

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Error,Unable to perform database operation", "Error", JOptionPane.ERROR\_MESSAGE);

}//inner try catch closed

} catch (Exception ex1) {

JOptionPane.showMessageDialog(null, "Error on database connection, Cannot perform database operation", "Error", JOptionPane.ERROR\_MESSAGE);

}//outer try catch closed

}//actionPerformed() closed

}//class closed

class HandelWindows

{

void makeLogin() throws SQLException,ClassNotFoundException

{

Login log = new Login();

log.setSize(330,250);

log.setTitle(" Student Record Management System");

log.setVisible(true);

log.setLocation(400,220);

log.setResizable(false);

}

void makeHomePage() () throws SQLException,ClassNotFoundException

{

ShowStatusClassWise rpd = new ShowStatusClassWise();

rpd.setBackground(Color.LIGHT\_GRAY);

rpd.setSize(750,600);

rpd.setVisible(true);

rpd.setTitle("Student Status Class-Wise");

rpd.setLocation(140,90);

rpd.setResizable(false);

}

void makeAboutUs() throws SQLException,ClassNotFoundException

{

AboutUs na = new AboutUs();

na.setSize(420,380);

na.setVisible(true);

na.setLocation(300,170);

na.setTitle("About Us");

na.setResizable(false);

}

}

class Validation

{

public boolean isDigits(String str)

{

boolean flag = true;

for(int i=0;i<str.length();i++)

{

if(((str.charAt(i)+"".hashCode()>=48)&&(str.charAt(i)+"".hashCode()<=57)))

{

flag = true;

}

else

{

flag = false;

break;

}

}

return flag;

}

public boolean isCharacters(String str)

{

boolean flag = true;

for(int i=0;i<str.length();i++)

{

if(((str.charAt(i)+"".hashCode()>=65)&&(str.charAt(i)+"".hashCode()<=90))||

((str.charAt(i)+"".hashCode()>=97)&&(str.charAt(i)+"".hashCode()<=122))||(str.charAt(i)+"".hashCode()==32))

{

flag = true;

}

else

{

flag = false;

break;

}

}

return flag;

}

}

import java.awt.Container;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JButton;

import javax.swing.JInternalFrame;

import javax.swing.JLabel;

import javax.swing.JToolBar;

public class MarkS extends JInternalFrame implements ActionListener {

Container c = getContentPane();

int rno;

JButton jbc = new JButton("Close");

JButton jb1 = new JButton("Semester1");

JButton jb2 = new JButton("Semester2");

JButton jb3 = new JButton("Semester3");

JButton jb4 = new JButton("Semester4");

JButton jb5 = new JButton("Semester5");

JButton jb6 = new JButton("Semester6");

String regn;

public MarkS(String reg) {

super("MARKS OF SSLC BATCH", false, true, false, true);

setDefaultCloseOperation(DISPOSE\_ON\_CLOSE);

setBounds(150, 100, 500, 200);

setLayout(null);

regn = reg;

JLabel jlr = new JLabel("Edit Marks of Student with Regno : " + regn);

JToolBar jtb = new JToolBar();

jtb.setBounds(0, 0, 700, 30);

add(jtb);

jtb.add(jb1);

jtb.add(jb2);

jtb.add(jb3);

jtb.add(jb4);

jtb.add(jb5);

jtb.add(jb6);

jlr.setBounds(50, 35, 500, 25);

jbc.setBounds(150, 100, 80, 30);

add(jlr);

add(jbc);

jb1.addActionListener(this);

jb2.addActionListener(this);

jb3.addActionListener(this);

jb4.addActionListener(this);

jb5.addActionListener(this);

jb6.addActionListener(this);

jbc.addActionListener(this);

}

public void actionPerformed(ActionEvent e) {

if (e.getSource() == jb1) {

SSLC frm = new SSLC(regn,"First Semester marks",1);

frmAdminMDI.desktop.add(frm);

frm.setVisible(true);

}

if (e.getSource() == jb2) {

SSLC frm = new SSLC(regn,"Second Semester marks",2);

frmAdminMDI.desktop.add(frm);

frm.setVisible(true);

}

if (e.getSource() == jb3) {

SSLC frm = new SSLC(regn,"Third Semester marks",3);

frmAdminMDI.desktop.add(frm);

frm.setVisible(true);

}

if (e.getSource() == jb4) {

SSLC frm = new SSLC(regn,"Forth Semester marks",4);

frmAdminMDI.desktop.add(frm);

frm.setVisible(true);

}

if (e.getSource() == jb5) {

SSLC frm = new SSLC(regn,"Fifth Semester marks",5);

frmAdminMDI.desktop.add(frm);

frm.setVisible(true);

}

if (e.getSource() == jb6) {

SSLC frm = new SSLC(regn,"Sixth Semester marks",6);

frmAdminMDI.desktop.add(frm);

frm.setVisible(true);

}

if (e.getSource() == jbc) {

dispose();

}

}

}

**Subject Allocation**

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.Container;

import java.awt.FlowLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import javax.swing.JButton;

import javax.swing.JComboBox;

import javax.swing.JInternalFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.JTableHeader;

public class SubjectAllocation extends JInternalFrame implements ActionListener {

private JLabel lblCourse, lblSem, lblSubject;

}//outer if closed

}//action if closed

}//actionPerformed() clsoed

private void loadTableValues() {

while (subjtable.getRowCount() > 0) {

tablemodel.removeRow(subjtable.getRowCount() - 1);

}

int sem = Integer.parseInt(cmbSem.getSelectedItem().toString());

String batch = cmbCourse.getSelectedItem().toString();

try {

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

Connection con = DriverManager.getConnection("jdbc:odbc:student");

String sql = "SELECT SubjectName FROM SubjectAllocation WHERE Batch='" +

batch + "' and Semester= " + sem;

PreparedStatement ps = con.prepareStatement(sql);

ResultSet rs = ps.executeQuery();

while (rs.next()) {

String str = rs.getString(1);

String[] data = {str};

tablemodel.addRow(data);

}

con.close();

} catch (Exception ex) {

System.out.println(ex.toString());

}

}

}//class closed

**CONCLUSION**

The project made me realize the significance of developing software for client, where all the eventualities have to be taken care of, unlike at different organization, where the sole aim is to keep track of student. What may be seen to the developed to be software complete in all respects, I was required to meet the stringent demands of the in-house client in the materials department. Through this seemed tough at first, I realized that pressure goaded us on to develop better and better applications.

During the project, the real importance for following all the principle of System Analysis and Design drawled on me. I felt the necessity of going through the several stages, because only such a process could make one understand the problems at hand, more so due to the enormous size of data stores involved and their manipulation

In retrospect, I would like to say that the institute provided us with many opportunities to learn MS Access and its tools. I specifically was able to learn MS Access and its Tools. I specially was able to learn java. I also succeeded in understanding a number of tricky concept in database, which were earlier only hazy blobs in the realm of our comprehension. I hope this stint at Student Record Management would stand me in good stead in future endeavors.

However my project Result Processing System is an asset to the organization, even then *FURTHER ENHANCEMENT* are possible in this project such as new forms & reports can be included as and when required.

Unfortunately due to lack of time we can developed any report in this system but this is not the end of this system software, it is beginning. We can change and modify this software due to more requirements of needs and also removed some errors which is finding in this system software.

We also developed some complex reports which is required and makes this software more powerful and friendly for user by which user can handle data more easily and can calculate all the calculation without any other software i.e., user’s all the requirements will be fulfill by this software and user can manage all the numeric data calculations.

**SCOPE FOR FUTURE**

It will be save money according to its performance also will be helpful fast accessing of the organization.

**BIBLIOGRAPHY**

***BOOK NAME AUTHOR***

Complete Reference of MS-Office Tec media

An Introduction to Database Systems C. J. Date

System Analyses and Design E M. Awad

Mastering in java Steve

Working with java Rita Sahoo

Learning Java O’Reilly

Effective Java Joshua Bloch

Head First Java O’Reilly

The Complete Reference Herbert Schildt

Java Concepts Cay Horstmann