Table of Contents

[Chapter 1 6](#_Toc436211247)

[Introduction 6](#_Toc436211248)

[1. Introduction 6](#_Toc436211249)

[1.1 Outline of the Thesis 7](#_Toc436211250)

[1.2 Scope of this application 7](#_Toc436211251)

[1.3 Focusing of the Project 7](#_Toc436211252)

[1.4 Features of the Project 7](#_Toc436211253)

[1.5 Module of Easy Education Management System 8](#_Toc436211254)

[1.1 8](#_Toc436211255)

[1.5.1 Administration 8](#_Toc436211256)

[1.5.2 Personal Information Management System (PIMS) 8](#_Toc436211257)

[1.5.3 Attendance 8](#_Toc436211258)

[1.5.4 Result 9](#_Toc436211259)

[1.5.5 Accounts 9](#_Toc436211260)

[1.5.6 Reports 9](#_Toc436211261)

[1.5.7 Students Information to Parents by Easy SMS Service 9](#_Toc436211262)

[Chapter 2 10](#_Toc436211263)

[2. Development platform Introduction 10](#_Toc436211264)

[2.1 C# Features 10](#_Toc436211265)

[2.2 Database Platform 11](#_Toc436211266)

[2.3 SQL Server database Introduction 11](#_Toc436211267)

[2.4 SQL Server database system features 11](#_Toc436211268)

[Chapter 3 13](#_Toc436211269)

[System Analysis 13](#_Toc436211270)

[3. Introduction 13](#_Toc436211271)

[3.1 System Engineering and Analysis 13](#_Toc436211272)

[3.2 System Analysis 13](#_Toc436211273)

[3.3 Requirement Analysis 14](#_Toc436211274)

[3.4 Software Requirements 14](#_Toc436211275)

[3.5 Language USED 14](#_Toc436211276)

[3.6 Flow Chart 15](#_Toc436211277)

[3.7 Data Flow Diagram (DFD) 16](#_Toc436211278)

[3.8 ER Diagram 17](#_Toc436211279)

[3.9 Relational Database 18](#_Toc436211280)

[3.9.1 User 18](#_Toc436211281)

[3.9.2 Administration 18](#_Toc436211282)

[3.9.3 Student 19](#_Toc436211283)

[3.9.4 Employee 19](#_Toc436211284)

[3.9.5 Routine 20](#_Toc436211285)

[Chapter 4 21](#_Toc436211286)

[System Design & Coding Structure 21](#_Toc436211287)

[4 Introduction 21](#_Toc436211288)

[4.1 Database Design 21](#_Toc436211289)

[4.2 System Architecture Design 22](#_Toc436211290)

[4.3 Code Design 23](#_Toc436211291)

[4.4 Sample Code 24](#_Toc436211292)

[4.5 Distinguishing Features of C# 33](#_Toc436211293)

[4.5.3 Memory access 33](#_Toc436211294)

[4.5.4 Exception 33](#_Toc436211295)

[4.5.5 Polymorphism 33](#_Toc436211296)

[4.5.6 Methods and functions 34](#_Toc436211297)

[4.5.7 User Interface Design 34](#_Toc436211298)

[Chapter 5 35](#_Toc436211299)

[System implementation 35](#_Toc436211300)

[5 System implementation 35](#_Toc436211301)

[5.1 User Training 35](#_Toc436211302)

[5.2 Distributed User Documentation 35](#_Toc436211303)

[5.2.1 Finalized System Documentation 35](#_Toc436211304)

[5.2.2 Installed Production System 36](#_Toc436211305)

[5.2.3 Post-Implementation Review Summary 36](#_Toc436211306)

[5.2.4 Methodology Compliance Form 36](#_Toc436211307)

[5.3 Software deployment 36](#_Toc436211308)

[5.4 Deployment Activities 37](#_Toc436211309)

[5.4.1 Release 37](#_Toc436211310)

[5.4.2 Install and activate 37](#_Toc436211311)

[5.4.3 Deactivate 37](#_Toc436211312)

[5.4.4 Adapt 37](#_Toc436211313)

[5.4.5 Update 37](#_Toc436211314)

[5.4.6 Built-In 38](#_Toc436211315)

[5.4.7 38](#_Toc436211316)

[5.4.8 Version tracking 38](#_Toc436211317)

[5.4.9 Uninstall 38](#_Toc436211318)

[5.4.10 Retire 38](#_Toc436211319)

[Chapter 6 39](#_Toc436211320)

[System Test 39](#_Toc436211321)

[6 Software Testing 39](#_Toc436211322)

[6.1 Principles of software testing 39](#_Toc436211323)

[Chapter 7 41](#_Toc436211324)

[User Manual 41](#_Toc436211325)

[7 Information of Table 41](#_Toc436211326)

[7.1 User Table 41](#_Toc436211327)

[7.2 User Group Table 41](#_Toc436211328)

[7.3 Menu Table 42](#_Toc436211329)

[7.4 Routine Table 42](#_Toc436211330)

[7.5 Student Progress Table 43](#_Toc436211331)

[7.6 Student Table 43](#_Toc436211332)

[8 Manual 44](#_Toc436211333)

[8.1 Login 44](#_Toc436211334)

[8.2 User Management 45](#_Toc436211335)

[8.2.1 User Group 45](#_Toc436211336)

[8.2.2 Group permission 46](#_Toc436211337)

[8.2.3 User Entry 46](#_Toc436211338)

[8.3 Administration 47](#_Toc436211339)

[8.3.1 Shift 47](#_Toc436211340)

[8.3.2 Class 48](#_Toc436211341)

[8.3.3 Subject 48](#_Toc436211342)

[8.3.4 Section 49](#_Toc436211343)

[8.3.5 Student Group 49](#_Toc436211344)

[8.3.6 Group Subjects 50](#_Toc436211345)

[8.3.7 Designation 50](#_Toc436211346)

[8.4 Attendance 51](#_Toc436211347)

[8.4.1 Student Attendance 51](#_Toc436211348)

[8.4.2 Employee Attendance 51](#_Toc436211349)

[8.4.3 Holidays 52](#_Toc436211350)

[8.5 PIMS (Personal Information Module System) 53](#_Toc436211351)

[8.5.1 Student Admission 53](#_Toc436211352)

[8.5.2 Employee Registration 54](#_Toc436211353)

[8.5.3 Student List 55](#_Toc436211354)

[8.5.4 Employee List 55](#_Toc436211355)

[8.5.5 Student Details Report 56](#_Toc436211356)

[8.5.6 Student list Report 56](#_Toc436211357)

[8.5.7 Employee Details Report 57](#_Toc436211358)

[8.5.8 Employee List Report 57](#_Toc436211359)

[8.6 Result 58](#_Toc436211360)

[8.6.1 Students Marks Policy 58](#_Toc436211361)

[8.6.2 Subject Wise Marks Entry 59](#_Toc436211362)

[8.7 Routine 60](#_Toc436211363)

[8.7.1 Routine Slot Entry 60](#_Toc436211364)

[8.7.2 Routine Entry 61](#_Toc436211365)

[8.7.3 Student Class Routine 61](#_Toc436211366)

[8.7.4 Teacher’s Schedule 62](#_Toc436211367)

[8.8 Accounts 62](#_Toc436211368)

[8.8.1 Debit Head 62](#_Toc436211369)

[8.8.2 Credit Head 62](#_Toc436211370)

[8.9 63](#_Toc436211371)

[8.9.1 Bank Deposit 63](#_Toc436211372)

[8.9.2 Bank Withdraw 63](#_Toc436211373)

[8.9.3 Income 64](#_Toc436211374)

[8.9.4 Expense 65](#_Toc436211375)

[Chapter 9 66](#_Toc436211376)

[Future Works and conclusion 66](#_Toc436211377)

[9 Future Expandability 66](#_Toc436211378)

[9.1 Conclusion 66](#_Toc436211379)

[References 67](#_Toc436211380)

# Chapter 1

# Introduction

# Introduction

Now a day’s education plays a great role in development of any country. Many of education organizations try to increase education quality. One of the aspects of this improvement is managing of school resources.

Education Management System carried on by any individual or institution engaged in providing a services to students, teachers, guardians and other persons are intermediary that performs one or more of the following functionalities – Student Admission, Employee Registration, Student List, Employee List, Student Attendance, Employee Attendance, Student Routine, Result Management, Payroll & Accounts.

Education Management System (EMS) is such a service which provides all services for an educational institute to make your life easier and faster by assuring its performance. Easy User Management System, Easy Admission Process, Easy Attendance System.

EMS is a system that will provide you a bird’s eye view of the functioning of the entire educational institution. It is a management information system helps to manage the different processes in an educational institution like General Administration, Staff Management, Academics, Student Management, and Accounts etc. The information is made using the latest technologies and help’s to make decision making a lot faster, effective and easier than ever before. Also helps to improve the overall quality of education of the institution.

We use database and database technology are having a major impact on the growing use of computers. The implementation of the system was done using c# and SQL Server 2012 technologies, allowing system to be run in Windows OS.

In a nutshell, Education Management Software managed your education institution by simplifying and automating processes and addressing the needs of all stakeholders helping them to be more efficient in their respective roles.

## Outline of the Thesis

The overview of related and used technologies in the implementation is given in Chapter 2.

The architecture and way of communication between client and service is explained in Chapter 3.

The detailed information about implementation of the system is presented in Chapter 4.

Chapter 5 provides the summary of the implemented system. The Appendices provides some additional information concerning the system.

## Scope of this application

The difference area where we can use this application as:

* Any education institution makes use of it providing class schedule.
* It can be used in offices and modifications can be easily done according to requirements.

## Focusing of the Project

The system is completed under the guidance of the theory and methods of management information systems, database technology support. This paper first discusses the structure of the background, purpose and significance of the graduate design topics. Then describes the development platform and database technology and the advantages of each, followed by more devoted a system requirements analysis, design, implementation, and the implementation of the tasks, techniques and tools. End system to complete the information input, output, data modification, query and statistics, as well as print statements, make operation simple and quick.

In this project, we try to build up sound software which can operate any challenging situation in the modern time. Administrator and users information are making effective decisions. The decisions are more accurate, relevant and timely the information stored or process is more effective.

## Features of the Project

The common features of the projects are:

* This is very easy to use for each user.
* Increase Efficiencies and Reduce Costs
* Transform IT for Higher Education
* Easy Solution
* Easy Admission Process
* Secure All Data
* Easy Account Maintenance
* Transaction History
* Easy Attendance Process
* The user of the database can see all information and also can edit if necessary.
* Easy implemented routine for student and teacher’s.

## Module of Easy Education Management System

## 

### 1.5.1 Administration

* System User Group Setup
* System User Setup
* System User Authorization Setup
* Class Setup
* Section Setup
* Student Group Setup
* Subject Setup
* Stuff Designation Setup
* Student Routine Setup
* Student Hall Setup
* Student Seat Setup

### 1.5.2 Personal Information Management System (PIMS)

* 1. Student Registration/ Admission
     + Personal Information
     + Guardian Information
  2. Employee/ Stuff Registration
     + Personal Information
     + Academic Information
     + Professional Information
     + Skills/ Training Information

### 1.5.3 Attendance

* Holiday Setup
* Student Attendance
* Employee / Stuff Attendance

### 1.5.4 Result

* Student Exam Setup
* Student Exam Result Entry

### 1.5.5 Accounts

* Debit Head Setup
* Credit Head Setup
* Income
* Expense
* Bank deposit
* Withdraw from Bank

### 1.5.6 Reports

* Student Daily Attendance
* Student’s Result
* Employee/ Staff Attendance
* Student Details
* Student List Report ( For Each Class)
* Class Wise Student Routine
* Employee details Report
* Employee/ Stuff List
* All Income By Date
* Head Wise Income By Month
* Head Wise Monthly Income by Financial Year
* All Expense By Date
* Head Wise Expense By Month
* Head Wise Monthly Expense By Year

### 1.5.7 Students Information to Parents by Easy SMS Service

* Admission Information Confirmation
* Student Attendance Report Each Day ( If Not Come)
* Student Attendance Report Each Exam ( If Not Come)
* Exam Date Information
* Exam Date Cancel Information

# Chapter 2

Platform Introduction

# Development platform Introduction

We use C# (.net framework) and MSSQL Server 2012.

C# is an elegant and type-safe object-oriented language that enables developers to build a variety of secure and robust applications that run on the .NET Framework. You can use C# to create Windows client applications, XML Web services, distributed components, client-server applications, database applications, and much, much more. Visual C# provides an advanced code editor, convenient user interface designers, integrated debugger, and many other tools to make it easier to develop applications based on the C# language and the .NET Framework.

The .NET Framework (pronounced dot net) is a software framework developed by [Microsoft](http://en.wikipedia.org/wiki/Microsoft) that runs primarily on [Microsoft Windows](http://en.wikipedia.org/wiki/Microsoft_Windows). It includes a large [class library](http://en.wikipedia.org/wiki/Class_library) known as Framework Class Library (FCL) and provides [language interoperability](http://en.wikipedia.org/wiki/Language_interoperability) (each language can use code written in other languages) across several [programming languages](http://en.wikipedia.org/wiki/Programming_language). Programs written for .NET Framework execute in a software environment (as contrasted to [hardware](http://en.wikipedia.org/wiki/Computer_hardware) environment), known as Common Language Runtime (CLR), an application virtual machine that provides services such as security, memory management, and exception handling. FCL and CLR together constitute .NET Framework.

## C# Features

* C# is a simple, modern, object oriented language derived from C++ and Java.
* It aims to combine the high productivity of Visual Basic and the raw power of C++.
* It is a part of Microsoft Visual Studio7.0.
* Visual studio supports Vb,VC++,C++,Vbscript, Jscript. All of these languages provide access to the Microsoft .NET platform.
* .NET includes a Common Execution engine and a rich class library.
* Microsofts JVM eqiv is Common language run time (CLR).
* CLR accommodates more than one languages such as C#, VB.NET, Jscript, ASP.NET,C ++.
* Source code --->Intermediate Language code (IL) ---> (JIT Compiler) Native code.
* The classes and data types are common to all of the .NET languages.
* We may develop Console application, Windows application, and Web application using C #.
* In C# Microsoft has taken care of C++ problems such as Memory management, pointers etc.
* It supports garbage collection, automatic memory management and a lot.

## Database Platform

A database is an organized collection of [data](http://en.wikipedia.org/wiki/Data_(computing)). The data is typically organized to model aspects of reality in a way that supports [processes](http://en.wikipedia.org/wiki/Process_(computing)) requiring information, such as modeling the availability of rooms in hotels in a way that supports finding a hotel with vacancies.

Database management systems (DBMS) are [computer software](http://en.wikipedia.org/wiki/Computer_software) applications that interact with the user, other applications, and the database itself to capture and analyze data. A general-purpose DBMS is designed to allow the definition, creation, querying, update, and administration of databases. Well-known DBMSs include [MySQL](http://en.wikipedia.org/wiki/MySQL), [PostgreSQL](http://en.wikipedia.org/wiki/PostgreSQL), [Microsoft SQL Server](http://en.wikipedia.org/wiki/Microsoft_SQL_Server), [Oracle](http://en.wikipedia.org/wiki/Oracle_Database), [Sybase](http://en.wikipedia.org/wiki/Sybase) and [IBM DB2](http://en.wikipedia.org/wiki/IBM_DB2).

## SQL Server database Introduction

In [computing](http://en.wikipedia.org/wiki/Computing), Microsoft SQL Server is a [relational database management system](http://en.wikipedia.org/wiki/Relational_database_management_system), currently developed by Microsoft. As a database server, it is a software product with the primary function of storing and retrieving data as requested by other [software applications](http://en.wikipedia.org/wiki/Software_applications) which may run either on the same computer or on another computer across a network (including the Internet).

Microsoft markets at least a dozen different editions of Microsoft SQL Server - aimed at different audiences and for workloads ranging from small single-machine applications to large Internet-facing applications with many [concurrent users](http://en.wikipedia.org/wiki/Concurrent_user).

## SQL Server database system features

* Supports most administrative tasks for SQL Server.
* A single, integrated environment for SQL Server Database Engine management and authoring.
* Dialogs for managing objects in the SQL Server Database Engine, Analysis Services, and Reporting Services, that allows you to execute your actions immediately, send them to a Code Editor, or script them for later execution.
* Non-modal and resizable dialogs allow access to multiple tools while a dialog is open.
* A common scheduling dialog that allows you to perform action of the management dialogs at a later time.
* Exporting and importing SQL Server Management Studio server registration from one Management Studio environment to another.
* Save or print XML Show plan or Deadlock files generated by SQL Server Profiler, review them later, or send them to administrators for analysis.
* A new error and informational message box that presents much more information, allows you to send Microsoft a comment about the messages, allows you to copy messages to the clipboard, and allows you to easily e-mail the messages to your support team.
* An integrated Web browser for quick browsing of MSDN or online help.

# Chapter 3

# System Analysis

# Introduction

Systems analysis is a problem solving technique that decomposes a system into its component pieces for the purpose of the studying how well those component parts work and interact to accomplish their purpose. As the software system requirements were predictable, it is decided to follow the classical system development life cycle method. This process demands a systematic, sequential approach to software development that begins at the system level and progress through analysis, design, coding, testing and maintenance. The steps that is applicable to all software engineering paradigms. The program is followed by SDLC (Software Development Life Cycle)

## System Engineering and Analysis

Software is always a part of a large system; work begins by establishing requirement for all system elements and then allocating some subsets of this requirement to software. This system view is essential when software must interface with other elements such as hardware, people and database. System engineering and analysis encompasses requirements gathering at the system level with a small amount of top-level design analysis.

## System Analysis

Analysis involves the requirement determination and specification. Systems analysis is a problem solving technique that decomposes a system into its component pieces for the purpose of the studying how well those component parts work and interact to accomplish their purpose. According to the Merriam-Webster dictionary, systems analysis is the process of studying a procedure or business in order to identify its goals and purposes and create systems and procedures that will achieve them in an efficient way. Analysis and synthesis, as scientific methods, always go hand in hand, they complement one another.

## Requirement Analysis

Requirements analysis in [systems engineering](http://en.wikipedia.org/wiki/Systems_engineering) and [software engineering](http://en.wikipedia.org/wiki/Software_engineering), encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product or project, taking account of the possibly conflicting [requirements](http://en.wikipedia.org/wiki/Requirement) of the various [stakeholders](http://en.wikipedia.org/wiki/Stakeholder_(corporate)), analyzing, documenting, validating and managing software or system requirements.

Requirements analysis is critical to the success of a systems or software project. The requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.

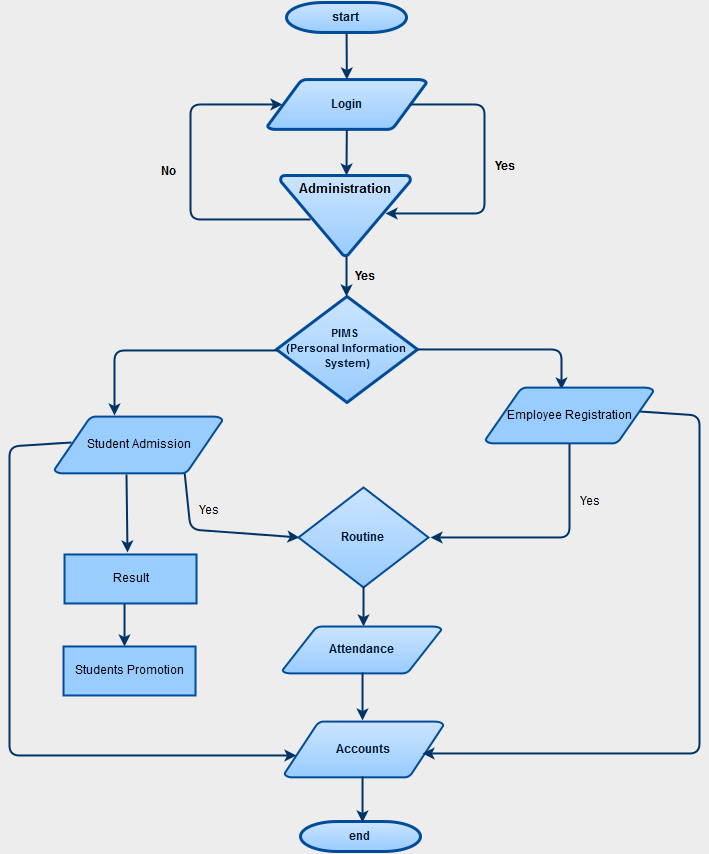
## Software Requirements

* Microsoft windows XP/ Windows 7/ Windows Vista/ Windows 8/ Windows 10/ Windows Server 2003, 2008, 2012.
* Visual Studio 2012 should be installed.
* .Net framework should be installed, Crystal report should be installed for report view (visual studio package not installed in OS).
* MSSQL Server 2012 should be installed.

## Language USED

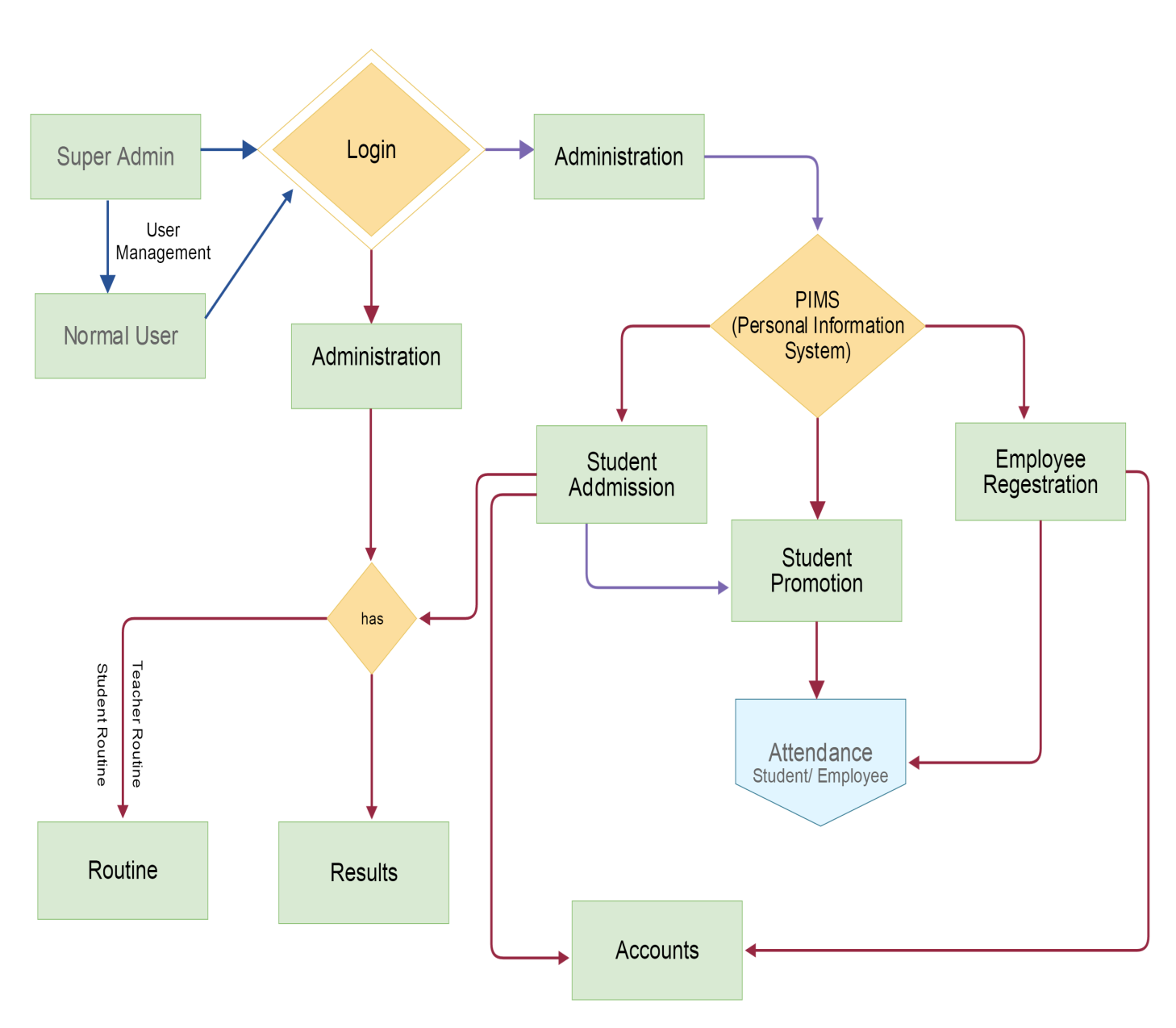
* C# in front end
* .Net Framework
* MS SQL Server DBMS (backend database)

## Flow Chart



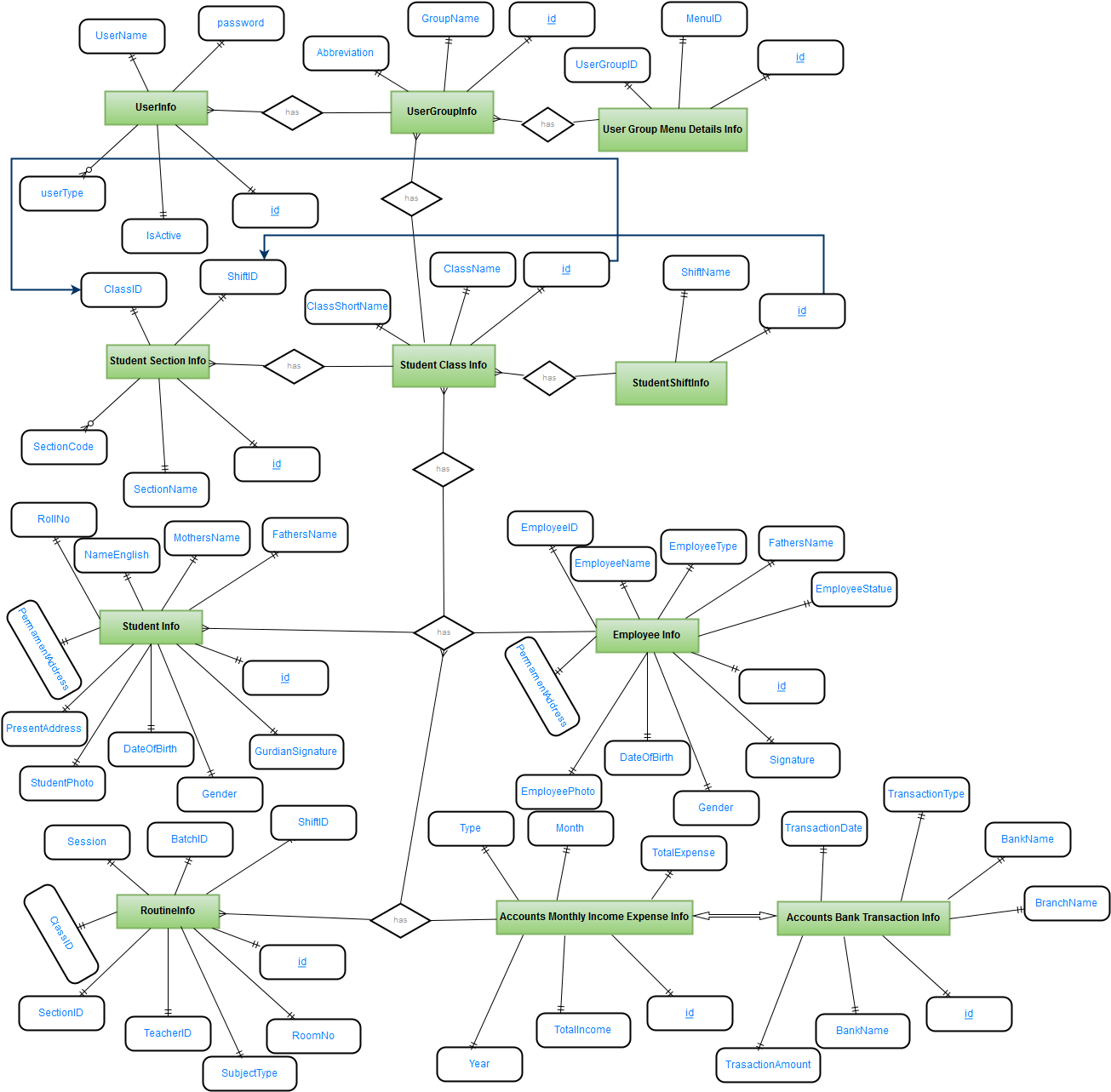
**Figure 0:** DFD Diagram

## Data Flow Diagram (DFD)



**Figure 1:** DFD Diagram

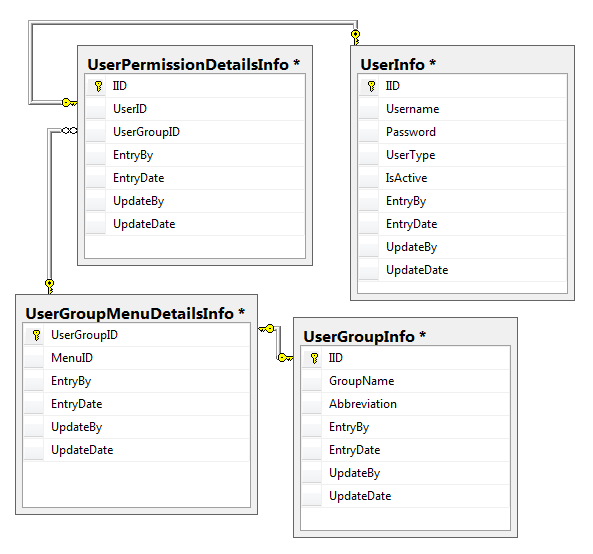
## ER Diagram



**Figure 2:** ER Diagram

## Relational Database

### User



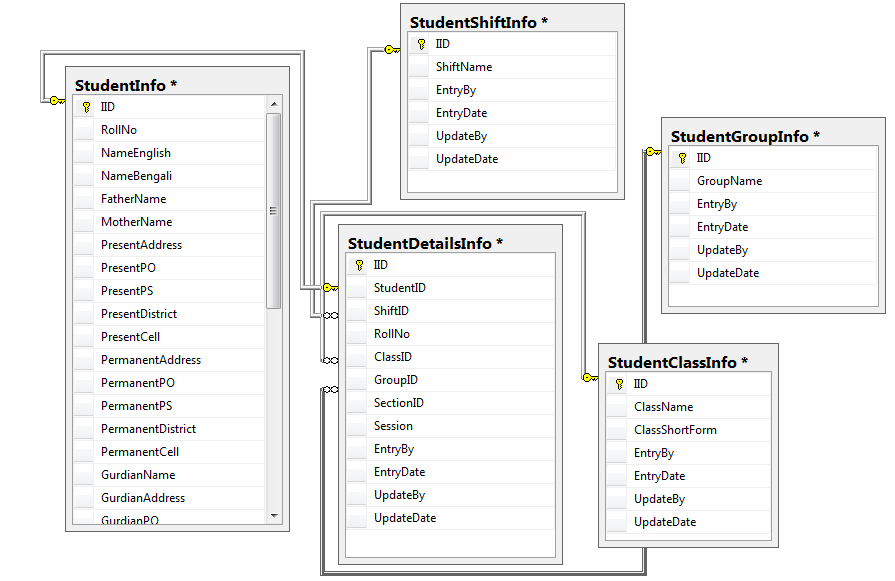
**Figure 3:** User Relational Database

### Administration



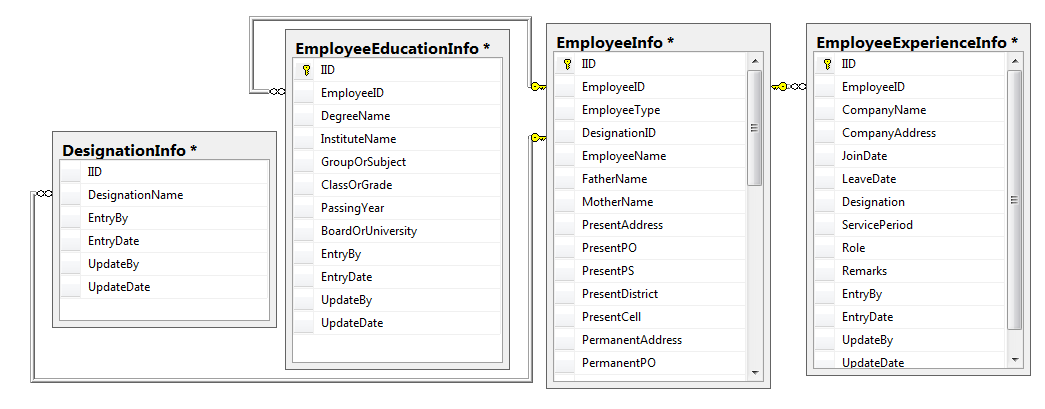
**Figure 4:** Administration Relational Database

### Student



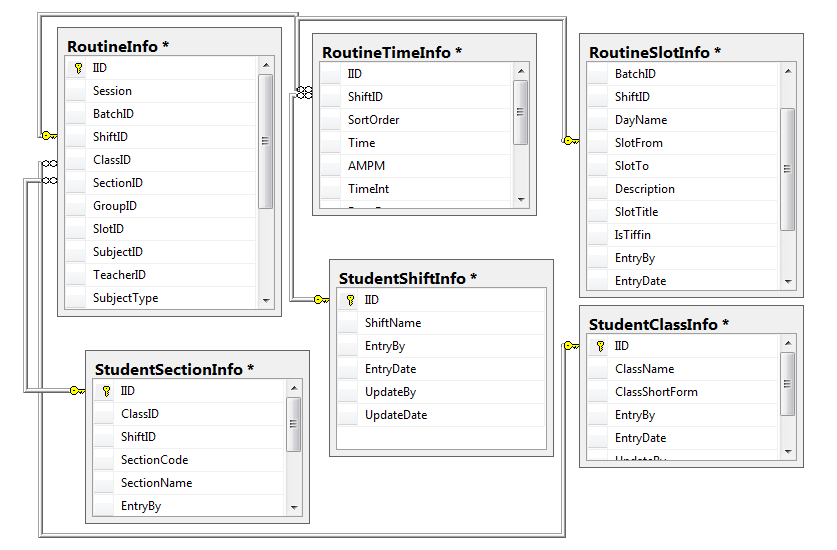
**Figure 5:** Student Relational Database

### Employee



**Figure 6:** Employee Relational Database

### Routine



**Figure 7:** Routine Relational Database

# Chapter 4

# System Design & Coding Structure

# Introduction

The [design phase](http://searchsoftwarequality.techtarget.com/definition/systems-development-life-cycle) is concerned with the physical construction of the system. Included are the design or configuration of the network (hardware, operating system, programming, etc.), design of user interfaces (forms, reports, etc.), design of system interfaces (for communication with other systems), and security issues. It is important that the proposed design be tested for performance, and to ensure that it meets the requirements outlined during the analysis phase. In other words, the main objective of this phase is to transform the previously defined requirements into a complete and detailed set of specifications which will be used during the next phase. Some of the activities that need to take place during the design phase are:

* Design the application
* Design and integrate the network
* Design and integrate the database
* Create a contingency plan
* Start a Maintenance, Training and Operations plan
* Review the design
* Articulate the business processes and procedures
* Establish a transition strategy
* Deliver the System Design Document
* Review final design

A database system is essentially nothing more than a computerized record keeping system the database itself can be regarded as kind of electronic filing cabinet. A database consists of same collection of some collection of persistent data that is used by the applications system of given some instituted. The term “instituted” here is simply a convenient generic term for any reasonable self- contained science, technical or other institution.

## Database Design

A database management system (DBMS) is a collection of programs that enables you to [store](http://www.webopedia.com/TERM/S/store.html), modify, and extract information from a [database](http://www.webopedia.com/TERM/D/database.html). There are many different types of database management systems, ranging from small [systems](http://www.webopedia.com/TERM/S/system.html) that [run](http://www.webopedia.com/TERM/R/run.html) on personal computers to huge systems that run on [mainframes](http://www.webopedia.com/TERM/M/mainframe.html).

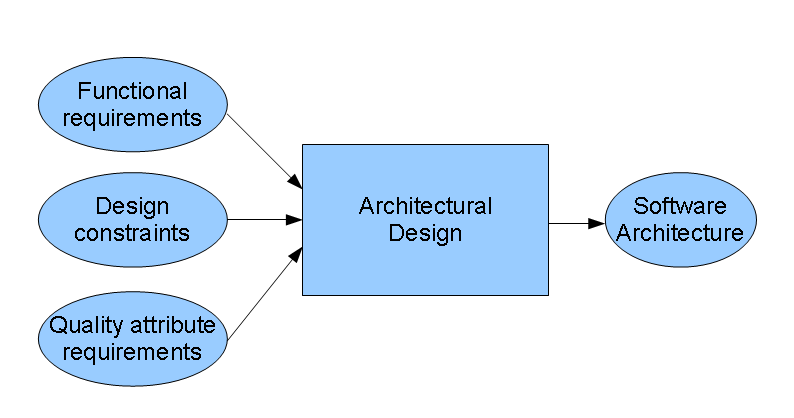
DBMS is a software that handles the storage, retrieval, and updating of data in a computer system.

Ex- SQL Server (Microsoft), MySQL (Freeware), Oracle (Oracle), NoSQL (Oracle), NonStop SQL (Hewlett Packard)

## System Architecture Design

A system architecture or systems architecture is the [conceptual model](http://en.wikipedia.org/wiki/Conceptual_model) that defines the [structure](http://en.wikipedia.org/wiki/Structure), [behavior](http://en.wikipedia.org/wiki/Behavior), and more [views](http://en.wikipedia.org/wiki/View_model) of a [system](http://en.wikipedia.org/wiki/System). An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the [structures](http://en.wikipedia.org/wiki/Structure) and [behaviors](http://en.wikipedia.org/wiki/Behavior) of the system.

System architecture can comprise system [components](http://en.wikipedia.org/wiki/System), the externally visible properties of those components, the relationships (e.g. the behavior) between them. It can provide a plan from which products can be procured, and systems developed, that will work together to implement the overall system. There have been efforts to formalize languages to describe system architecture, collectively these are called [architecture description languages](http://en.wikipedia.org/wiki/Architecture_description_languages).



**Figure 8:** Software Architecture Design

Software architecture refers to the high level structures of a [software system](http://en.wikipedia.org/wiki/Software_system), the discipline of creating such structures, and the documentation of these structures. It is the set of structures needed to reason about the software system. Each structure comprises software elements, relations among them, and properties of both elements and relations. The architecture of a software system is a metaphor, analogous to the [architecture](http://en.wikipedia.org/wiki/Architecture) of a building.

## Code Design

Design patterns are solutions to software design problems you find again and again in real-world application development. Patterns are about reusable designs and interactions of objects.

The 23 Gang of Four (GoF) patterns are generally considered the foundation for all other patterns. They are categorized in three groups: Creational, Structural, and Behavioral (for a complete list see below).

To give you a head start, the C# source code for each pattern is provided in 2 forms: structural and real-world. Structural code uses type names as defined in the pattern definition and UML diagrams. Real-world code provides real-world programming situations where you may use these patterns.

A third form, .NET optimized, demonstrates design patterns that fully exploit built-in .NET 4.5 features, such as, generics, attributes, delegates, reflection, and more. These and much more are available in our [.NET Design Pattern Framework 4.5](http://www.dofactory.com/products/net-design-pattern-framework).

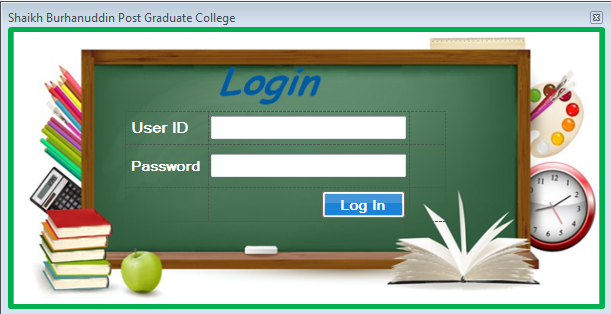
This structural code demonstrates the Singleton pattern which assures only a single instance (the singleton) of the class can be created.

This real-world code demonstrates the Singleton pattern as a Load Balancing objects. Only a single instance (the singleton) of the class can be created because servers may dynamically come on-or off-line and every request must go through the one object that has knowledge about the state of the (web) farm.

## Sample Code

* + 1. **Form Login**

Design:

****

Code:

private void Login()

{

try

{

if (IsValid())

{

UserInfo \_user = DataAccessFacade.Instance.GetUserInfoForLogin(txtUserName.Text.Trim(), txtPassword.Text.Trim());

if (\_user != null && string.Compare(txtPassword.Text.Trim(), \_user.Password.ToString().Trim()) == 0)

{

if (\_user.IsActive.ToUpper() == "YES")

{

MainForm.LoggedUser = \_user;

MainForm main = (MainForm)this.Owner;

main.mnuLogin.Enabled = false;

main.mnuLogout.Enabled = true;

main.AuthenticateUser();

this.Dispose();

}

else

{

MessageBox.Show("Currently you have no access.\nPlease Contact with system administrator.", ConfigurationManager.AppSettings["MessageBoxCaption"]);

}

}

else

{

MessageBox.Show("Error In User Name Or Password.", ConfigurationManager.AppSettings["MessageBoxCaption"]);

}

}

}

catch (Exception ex)

{

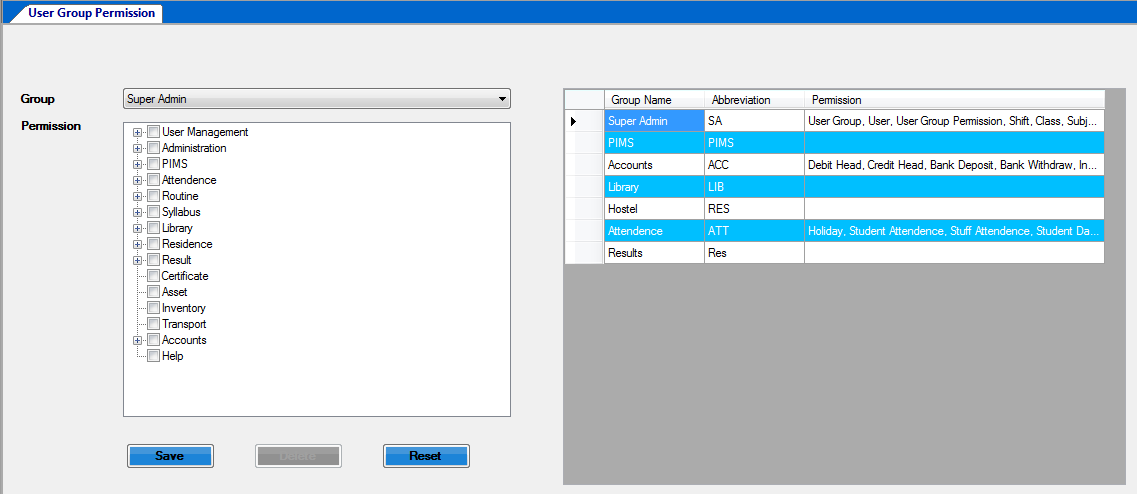
MessageBox.Show("Database Connection Error: Please Contact with System Administrator." + ex, ConfigurationManager.AppSettings["MessageBoxCaption"]);

}

}

* + 1. **User Group Permission**

Design:



Code:

private void btnSave\_Click(object sender, EventArgs e)

{

try

{

DataTable dtMenu = new DataTable();

dtMenu.Columns.Add(new DataColumn("MenuID", typeof(long)));

foreach (TreeNode parentNode in tvPermission.Nodes)

{

foreach (TreeNode tn in parentNode.Nodes)

{

if (tn.Nodes.Count == 0)

{

if (tn.Checked)

{

DataRow row1 = dtMenu.NewRow();

row1["MenuID"] = Convert.ToInt64(tn.Name);

dtMenu.Rows.Add(row1);

}

}

else

{

foreach (TreeNode lastNode in tn.Nodes)

{

if (lastNode.Checked)

{

DataRow row1 = dtMenu.NewRow();

row1["MenuID"] = Convert.ToInt64(lastNode.Name);

dtMenu.Rows.Add(row1);

}

}

}

}

if (IsValid(dtMenu))

{

if (string.IsNullOrEmpty(lblID.Text.Trim()))

{

if (DataAccessFacade.Instance.InsertUserGroupMenuDetailsInfo(dtMenu, Convert.ToInt64(cmbUserGroup.SelectedValue), MainForm.LoggedUser.Username))

{

ClearUIControls();

LoadGrid();

MessageBox.Show("Saved Successfully", ConfigurationManager.AppSettings["MessageBoxCaption"]);

}

else

{

MessageBox.Show("Not Saved", ConfigurationManager.AppSettings["MessageBoxCaption"]);

}

}

else

{

if (DataAccessFacade.Instance.InsertUserGroupMenuDetailsInfo(dtMenu, Convert.ToInt64(cmbUserGroup.SelectedValue), MainForm.LoggedUser.Username))

{

ClearUIControls();

LoadGrid();

MessageBox.Show("Updated Successfully", ConfigurationManager.AppSettings["MessageBoxCaption"]);

}

else

{

MessageBox.Show("Not Updated", ConfigurationManager.AppSettings["MessageBoxCaption"]);

}

}

}

}

catch (Exception ex)

{

MessageBox.Show("Not Saved/Updated: " + ex.Message, ConfigurationManager.AppSettings["MessageBoxCaption"]);

}

}

private void dgvUserGroup\_CellClick(object sender, DataGridViewCellEventArgs e)

{

LoadMenuForPermission();

lblID.Text = "";

btnSave.Text = "Update";

btnDelete.Enabled = true;

if (dgvUserGroup.Rows.Count > 0)

{

int index = dgvUserGroup.SelectedCells[0].RowIndex;

lblID.Text = dgvUserGroup.Rows[index].Cells[0].Value.ToString();

cmbUserGroup.SelectedValue = dgvUserGroup.Rows[index].Cells[0].Value;

List<UserGroupMenuDetailsInfo> \_listUserGroupMenuDetailsInfo = new List<UserGroupMenuDetailsInfo>();

\_listUserGroupMenuDetailsInfo = DataAccessFacade.Instance.GetUserGroupMenuDetailsInfoByRange(1, 100, "MenuID", "ASC", " UserGroupID = " + dgvUserGroup.Rows[index].Cells[0].Value);

if (\_listUserGroupMenuDetailsInfo != null)

{

foreach (UserGroupMenuDetailsInfo \_UserGroupMenuDetailsInfo in \_listUserGroupMenuDetailsInfo)

{

tvPermission.ExpandAll();

foreach (TreeNode tn in tvPermission.Nodes)

{

if (tn.Nodes.Count > 0)

{

SelectNodes(tn, \_UserGroupMenuDetailsInfo.MenuID.ToString());

}

else

{

if (tn.Name == \_UserGroupMenuDetailsInfo.MenuID.ToString())

{

tn.Checked = true;

}

}

}

}

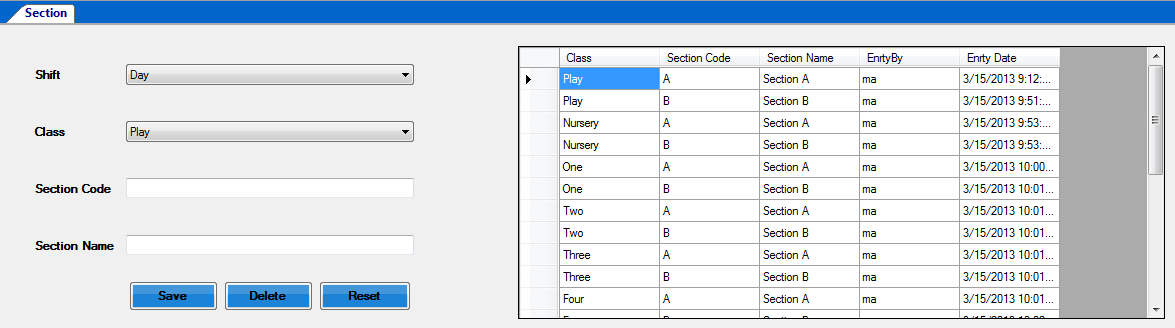
}

}

}

* + 1. **Section Entry**

Design:



Code:

private void btnSave\_Click(object sender, EventArgs e)

{

try

{

if (IsValid())

{

StudentSectionInfo \_StudentSectionInfo = CreateDTO();

if (string.IsNullOrEmpty(lblID.Text.Trim()))

{

if (DataAccessFacade.Instance.IsExist("StudentSectionInfo", " ClassID = " + \_StudentSectionInfo.ClassID.ToString() + " AND SectionCode = '" + \_StudentSectionInfo.SectionCode + "'"))

{

MessageBox.Show("Record already exists.", ConfigurationManager.AppSettings["MessageBoxCaption"]);

return;

}

if (DataAccessFacade.Instance.InsertStudentSectionInfo(\_StudentSectionInfo))

{

ClearUIControls();

LoadGrid();

MessageBox.Show("Saved Successfully", ConfigurationManager.AppSettings["MessageBoxCaption"]);

}

else

{

MessageBox.Show("Not Saved", ConfigurationManager.AppSettings["MessageBoxCaption"]);

}

}

else

{

if (DataAccessFacade.Instance.IsExist("StudentSectionInfo", " IID <> " + \_StudentSectionInfo.IID.ToString() + " AND ClassID = " + \_StudentSectionInfo.ClassID.ToString() + " AND SectionCode = '" + \_StudentSectionInfo.SectionCode + "'"))

{

MessageBox.Show("Record already exists.", ConfigurationManager.AppSettings["MessageBoxCaption"]);

return;

}

if (MessageBox.Show("Are you sure to update this record", ConfigurationManager.AppSettings["MessageBoxCaption"], MessageBoxButtons.YesNo, MessageBoxIcon.Information) == DialogResult.Yes)

{

if (DataAccessFacade.Instance.EditStudentSectionInfo(\_StudentSectionInfo))

{

ClearUIControls();

LoadGrid();

MessageBox.Show("Updated Successfully", ConfigurationManager.AppSettings["MessageBoxCaption"]);

}

else

{

MessageBox.Show("Not Updated", ConfigurationManager.AppSettings["MessageBoxCaption"]);

}

}

}

}

catch (Exception ex)

{

MessageBox.Show("Not Saved/Updated: " + ex.Message, ConfigurationManager.AppSettings["MessageBoxCaption"]);

}

cmbShift.Focus();

}

public void LoadGrid()

{

try

{

dgvSection.Rows.Clear();

dgvSection.AutoGenerateColumns = false;

DataTable dtSection = DataAccessFacade.Instance.GetBySP(DBConstants.SP\_GET\_STUDENTSECTIONINFO\_ALL\_FORGRID);

if (dtSection.Rows.Count > 0)

{

foreach (DataRow dr in dtSection.Rows)

{ dgvSection.Rows.Add(dr["IID"].ToString(), dr["ShiftID"].ToString(), dr["ClassID"].ToString(), dr["ClassName"].ToString(),

dr["SectionCode"].ToString(),

dr["SectionName"].ToString(),

dr["EntryBy"].ToString(),

dr["EntryDate"].ToString());

private void LoadClass()

{

List<StudentClassInfo> \_listStudentClassInfo = new List<StudentClassInfo>();

\_listStudentClassInfo = DataAccessFacade.Instance.GetAllStudentClassInfo();

//cmbUserGroup.Items.Add("Please Select");

if (\_listStudentClassInfo != null)

{

cmbClass.ValueMember = "IID";

cmbClass.DisplayMember = "ClassName";

cmbClass.DataSource = \_listStudentClassInfo;

}

}

private void LoadShift()

{

List<StudentShiftInfo> \_listStudentShiftInfo = new List<StudentShiftInfo>();

\_listStudentShiftInfo = DataAccessFacade.Instance.GetAllStudentShiftInfo();

//cmbUserGroup.Items.Add("Please Select");

if (\_listStudentShiftInfo != null)

{

cmbShift.ValueMember = "IID";

cmbShift.DisplayMember = "ShiftName";

cmbShift.DataSource = \_listStudentShiftInfo;

}

}

private bool IsValid()

{

bool isValid = true;

if (string.IsNullOrEmpty(txtSectionName.Text.Trim()))

{

errorProvider1.Clear();

errorProvider1.SetError(txtSectionName, "Section Name is required.");

isValid = false;

}

return isValid;

}

private StudentSectionInfo CreateDTO()

{

StudentSectionInfo \_StudentSectionInfo = new StudentSectionInfo();

if (!string.IsNullOrEmpty(lblID.Text.Trim()))

{

\_StudentSectionInfo.IID = Convert.ToInt64(lblID.Text.Trim());

\_StudentSectionInfo.UpdateBy = MainForm.LoggedUser.Username;

}

\_StudentSectionInfo.SectionName = txtSectionName.Text.Trim();

\_StudentSectionInfo.SectionCode = txtSectionCode.Text.Trim();

\_StudentSectionInfo.ShiftID = ((StudentShiftInfo)(cmbShift.SelectedItem)).IID;

\_StudentSectionInfo.ClassID = ((StudentClassInfo)(cmbClass.SelectedItem)).IID; //Convert.ToInt64(cmbClass.ValueMember);

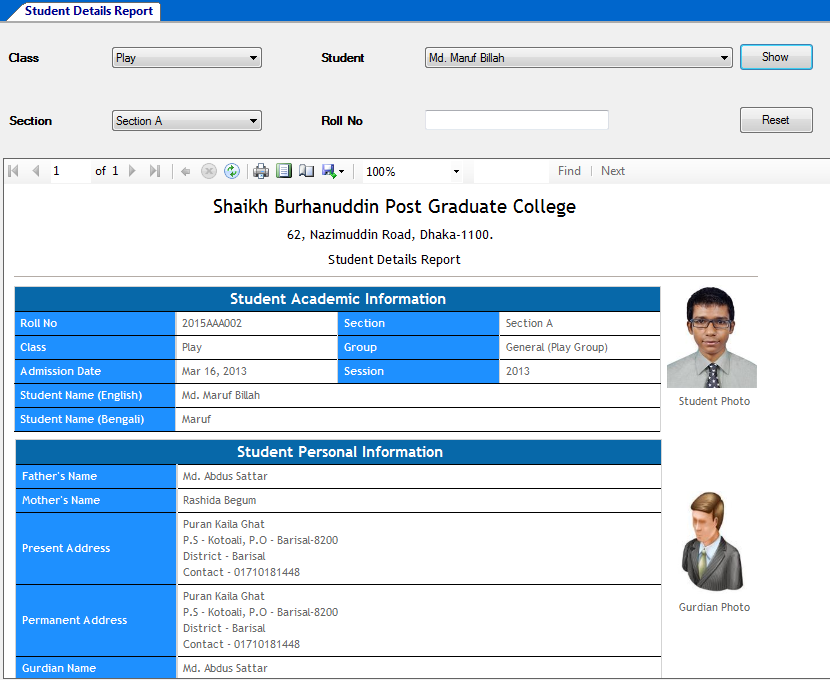
\_StudentSectionInfo.EntryBy = MainForm.LoggedUser.Username;

return \_StudentSectionInfo;

}

* + 1. **Students Daily Report**

Design:



Code:

public void LoadStudent()

{

cmbStudent.DataSource = null;

//cmbStudent.Items.Clear();

if (cmbSection.SelectedIndex != -1)

{

string whereClause = " ClassID = " + ((StudentClassInfo)cmbClass.SelectedItem).IID.ToString() + " AND SectionID = " + ((StudentSectionInfo)cmbSection.SelectedItem).IID.ToString() + " ";

DataTable dtStudent = DataAccessFacade.Instance.GetBySPWithParams(DBConstants.SP\_GET\_ALL\_STUDENT\_FOR\_DROPDOWN, new object[] { whereClause });

if (dtStudent.Rows.Count > 0)

{

cmbStudent.DisplayMember = "NameEnglish";

cmbStudent.ValueMember = "IID";

cmbStudent.DataSource = dtStudent;

}

}

private string GetWhereClause()

{

string strWhereClause = " 1=1 ";

if (!string.IsNullOrEmpty(txtRollNo.Text.Trim()))

{

strWhereClause = strWhereClause + " AND S.RollNo = '" + txtRollNo.Text.Trim() + "' ";

}

else

{

if (cmbStudent.SelectedIndex != -1)

{

strWhereClause = strWhereClause + " AND S.IID = " + ((DataRowView)(cmbStudent.SelectedItem)).Row["IID"].ToString() + " ";

}

}

return strWhereClause;

}

private void DisplayReport(string strWhereClause)

{

try

{

rvwReports.LocalReport.DataSources.Clear();

DataTable dtReportData = ReportViewerDataAccess.FillReportDataSetTable(DBConstants.SP\_REPORT\_STUDENT\_DETAILS, new object[] { strWhereClause });

if (dtReportData.Rows.Count > 0)

{

DataTable dtHistoryData = ReportViewerDataAccess.FillReportDataSetTable(DBConstants.SP\_GET\_STUDENTPROGRESSHISTORY\_ALL, new object[] { " StudentID='" + dtReportData.Rows[0]["IID"].ToString() + "' " });

DataTable dtLogo = ReportViewerDataAccess.GetLogo();

rvwReports.LocalReport.DataSources.Add(new ReportDataSource("SPResults", dtReportData));

rvwReports.LocalReport.DataSources.Add(new

rvwReports.LocalReport.DataSources.Add(new ReportDataSource("Logo", dtLogo));

ReportParameter[] param = new ReportParameter[12];

param[0] = new ReportParameter("DevelopedBy", dtLogo.Rows[0]["DeveloperName"].ToString());

param[1] = new ReportParameter("MarketedBy", dtLogo.Rows[0]["MarketerName"].ToString());

param[2] = new ReportParameter("SpondonWeb", dtLogo.Rows[0]["SystemWeb"].ToString());

param[9] = new ReportParameter("SystemLogo", Convert.ToBase64String((byte[])(dtLogo.Rows[0]["SystemLogo"])));

rvwReports.RefreshReport();

}

else

{

MessageBox.Show("No records return." + Environment.NewLine + "Please change criteria and try again.", ConfigurationManager.AppSettings["MessageBoxCaption"]);

}

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

private void cmbClass\_SelectionChangeCommitted(object sender, EventArgs e)

{

if (cmbClass.SelectedIndex != -1)

{

cmbSection.DataSource = null;

List<StudentSectionInfo> \_listStudentSectionInfo = DataAccessFacade.Instance.GetAllStudentSectionInfo(" ClassID = " + ((StudentClassInfo)(cmbClass.SelectedItem)).IID.ToString());

if (\_listStudentSectionInfo != null)

{

cmbSection.DisplayMember = "SectionName";

cmbSection.ValueMember = "IID";

cmbSection.DataSource = \_listStudentSectionInfo;

cmbSection.SelectedIndex = -1;

}

}

}

private void cmbSection\_SelectionChangeCommitted(object sender, EventArgs e)

{

if (cmbSection.SelectedIndex != -1)

{

LoadStudent();

}

}

}

## Distinguishing Features of C#

* + 1. **Portability**

By design, C# is the programming language that most directly reflects the underlying [Common Language Infrastructure](http://en.wikipedia.org/wiki/Common_Language_Infrastructure) (CLI). Most of its intrinsic types correspond to value-types implemented by the CLI framework. However, the language specification does not state the code generation requirements of the compiler: that is, it does not state that a C# compiler must target a Common Language Runtime, or generate [Common Intermediate Language](http://en.wikipedia.org/wiki/Common_Intermediate_Language) (CIL), or generate any other specific format. Theoretically, a C# compiler could generate machine code like traditional compilers of C++ or FORTRAN.

* + 1. **Namespace**

A C# namespace provides the same level of code isolation as a Java package or a C++ namespace, with very similar rules and features to a package.

### Memory access

In C#, memory address pointers can only be used within blocks specifically marked as unsafe, and programs with unsafe code need appropriate permissions to run. Most object access is done through safe object references, which always either point to a "live" object or have the well-defined [null](http://en.wikipedia.org/wiki/Nullable_type) value; it is impossible to obtain a reference to a "dead" object (one that has been garbage collected), or to a random block of memory.

### Exception

[Checked exceptions](http://en.wikipedia.org/wiki/Checked_exceptions) are not present in C# (in contrast to Java). This has been a conscious decision based on the issues of scalability and version ability.

### Polymorphism

Unlike [C++](http://en.wikipedia.org/wiki/Cplusplus), multiple inheritances is not supported by C#, although a class can implement any number of interfaces. This was a design decision by the language's lead architect to avoid complication and simplify architectural requirements throughout CLI. When implementing multiple interfaces that contain a method with the same signature, C# allows the programmer to implement each method depending on which interface that method is being called through, or, like Java, allows the programmer to implement the method once and have that be the single invocation on a call through any of the class's interfaces.

### Methods and functions

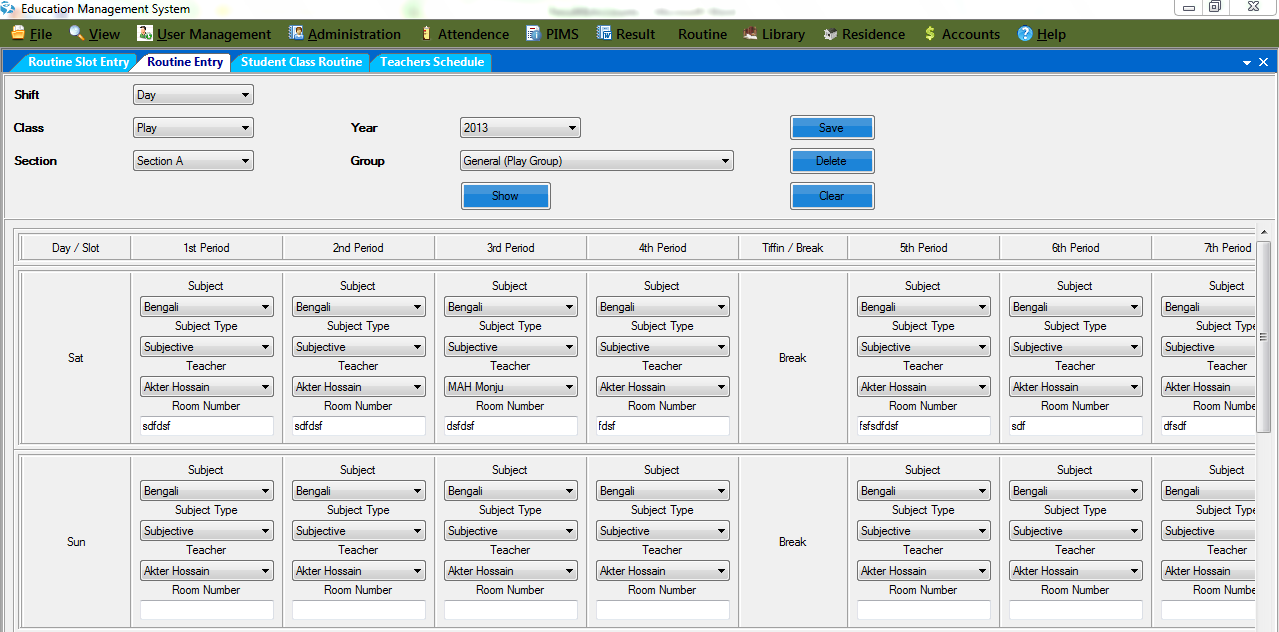
Like C++, and unlike Java, C# programmers must use the keyword virtual to allow methods to be overridden by subclasses.

Extension methods in C# allow programmers to use static methods as if they were methods from a class's method table, allowing programmers to add methods to an object that they feel should exist on that object and its derivatives.

### User Interface Design

User interface design (UI) or user interface engineering is the [design](http://en.wikipedia.org/wiki/Design) of [user interfaces](http://en.wikipedia.org/wiki/User_interface) for machines and software, such as [computers](http://en.wikipedia.org/wiki/Computer), [home appliances](http://en.wikipedia.org/wiki/Home_appliance), [mobile devices](http://en.wikipedia.org/wiki/Mobile_device), and other [electronic devices](http://en.wikipedia.org/wiki/Electronics), with the focus on maximizing the [user experience](http://en.wikipedia.org/wiki/User_experience). The goal of user interface design is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user goals ([user-centered design](http://en.wikipedia.org/wiki/User-centered_design)).

Good user interface design facilitates finishing the task at hand without drawing unnecessary attention to it. [Graphic design](http://en.wikipedia.org/wiki/Graphic_design) and [typography](http://en.wikipedia.org/wiki/Typography) are utilized to support its [usability](http://en.wikipedia.org/wiki/Usability), influencing how the user performs certain interactions and improving the aesthetic appeal of the design; design aesthetics may enhance or detract from the ability of users to use the functions of the interface.



**Figure 9:** User Interface

# Chapter 5

# System implementation

# System implementation

Implementation refers to the final process of moving the solution from development status to production status. Depending on your project, this process is often called deployment, go-live, rollout or installation. For the purposes of Life cycle Step, all of these terms are synonymous with "implementation."

There is no single way to implement an application. It depends on the characteristics of your project and the solution. Some implementations are as easy as saying “we are now live.” This type of implementation can work when the solution is brand new and you are developing and testing in what will become the production environment. In these cases, implementation is just a state of mind. One day the solution is in development, and the next day it is in production.

## User Training

The initial training classes for users are held, and training materials are delivered at the classes. Some help desk personnel should attend the initial user training class. More training classes can be scheduled later, as new personnel start using the application. Training is done on the user acceptance test system, accessing the test database or a special training database.

## Distributed User Documentation

User documentation that was finalized in User Acceptance Testing is now distributed and in the users' possession.

### Finalized System Documentation

System documentation corrected with all updates from the testing phases is handed over to production support.

### Installed Production System

The production system is installed in the appropriate production environment or on the appropriate production server, and on any client workstations that require it.

### Post-Implementation Review Summary

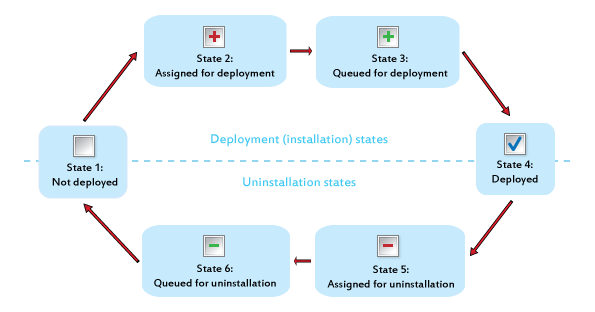
This report is produced after the post-implementation review meeting, and contains a summary of the project success criteria that were met, success criteria that were not met and reasons for the problem, what we can learn from the project to improve practices for the next project. In particular, the report should identify any techniques or practices used in this project that worked extremely well, and which the project team feels would benefit current and future projects.

### Methodology Compliance Form

This form is initialized by the project team, and completed by a methodology representative who has reviewed the project documentation and found it acceptable. It is completed in Word.

## Software deployment

Software deployment is all of the activities that make a [software system](https://en.wikipedia.org/wiki/Software_system) available for use. The general deployment process consists of several interrelated activities with possible transitions between them. These activities can occur at the [producer](https://en.wikipedia.org/wiki/Software_developer) side or at the [consumer](https://en.wikipedia.org/wiki/User_(computing)) side or both. Because every [software system](https://en.wikipedia.org/wiki/Software_system) is unique, the precise [processes](https://en.wikipedia.org/wiki/Process_(computing)) or [procedures](https://en.wikipedia.org/wiki/Algorithm) within each activity can hardly be defined. Therefore, "deployment" should be interpreted as a general process that has to be customized according to specific requirements or characteristics. A brief description of each activity will be presented later.



**Figure 10:** Deployment Process

## Deployment Activities

### Release

The [release](https://en.wikipedia.org/wiki/Software_release) activity follows from the completed [development](https://en.wikipedia.org/wiki/Software_development_process) process. It includes all the operations to prepare a system for [assembly](https://en.wikipedia.org/wiki/Compiler) and transfer to the customer site. Therefore, it must determine the [resources](https://en.wikipedia.org/wiki/Resource_(computer_science)) required to operate at the customer site and collect information for carrying out subsequent activities of deployment process.

### Install and activate

Activation is the activity of starting up the [executable](https://en.wikipedia.org/wiki/Executable) component of software. For simple systems, it involves establishing some form of [command](https://en.wikipedia.org/wiki/Command_(computing)) for execution. For complex systems, it should make all the supporting systems ready to use. (Not to be confused with the common use of the term activation concerning a software license, which is a function of [Digital Rights Management](https://en.wikipedia.org/wiki/Digital_Rights_Management#Limited_install_activations) systems.)

In larger software deployments, the working copy of the [software](https://en.wikipedia.org/wiki/Software) might be installed on a production server in a production environment. Other versions of the deployed software may be installed in a [test environment](https://en.wikipedia.org/w/index.php?title=Test_environment&action=edit&redlink=1), [development environment](https://en.wikipedia.org/wiki/Development_environment_(software_development_process)) and disaster recovery environment.

Further information: [Installation (computer programs)](https://en.wikipedia.org/wiki/Installation_(computer_programs))

### Deactivate

Deactivation is the inverse of activation, and refers to shutting down any executing components of a system. Deactivation is often required to perform other deployment activities, e.g., a software system may need to be deactivated before an update can be performed. The practice of removing infrequently used or obsolete systems from service is often referred to as [application retirement](https://en.wikipedia.org/wiki/Application_retirement) or application decommissioning.

### Adapt

The [adaptation](https://en.wikipedia.org/wiki/Adaptation_(computer_science)) activity is also a process to modify a software system that has been previously installed. It differs from updating in that adaptations are initiated by local events such as changing the [environment](https://en.wikipedia.org/wiki/Runtime_environment) of customer site, while updating is mostly started from remote software producer.

### Update

The update process replaces an earlier version of all or part of a software system with a newer release.

### Built-In

Mechanisms for installing updates are built into some software systems. Automation of these update processes ranges from fully automatic to user initiated and controlled. [Norton Internet Security](https://en.wikipedia.org/wiki/Norton_Internet_Security) is an example of a system with a semi-automatic method for retrieving and installing updates to both the antivirus definitions and other components of the system. Other software products provide query mechanisms for determining when updates are available.

### 

### Version tracking

Version tracking systems help the user find and install updates to software systems installed on PCs and local networks.

* Web based version tracking systems notify the user when updates are available for software systems installed on a local system. For example: Version Tracker Pro checks software versions on a user's computer and then queries its database to see if any updates are available.
* Local version tracking system notifies the user when updates are available for software systems installed on a local system. For example: Catalog stores version and other information for each software package installed on a local system. One click of a button launches a browser window to the upgrade web page for the application, including auto-filling of the user name and password for sites that require a login.
* Browser based version tracking systems notify the user when updates are available for software packages installed on a local system. For example: [wfx-Versions](https://en.wikipedia.org/w/index.php?title=Wfx-Versions&action=edit&redlink=1) is a Firefox extension which helps the user find the current version number of any program listed on the web.

### Uninstall

Uninstallation is the inverse of installation. It is the removal of a system that is no longer required. It also involves some reconfiguration of other software systems in order to remove the uninstalled system’s [files](https://en.wikipedia.org/wiki/Computer_file) and [dependencies](https://en.wikipedia.org/wiki/Coupling_(computer_science)).

### Retire

Ultimately, a software system is marked as [obsolete](https://en.wikipedia.org/wiki/Obsolete) and [support](https://en.wikipedia.org/wiki/Technical_support) by the producers is withdrawn. It is the end of the [life cycle of a software product](https://en.wikipedia.org/wiki/Product_life_cycle_management).

# Chapter 6

# System Test

# Software Testing

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the [software](https://en.wikipedia.org/wiki/Software) to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with the intent of finding [software bugs](https://en.wikipedia.org/wiki/Software_bug) (errors or other defects).

It involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test:

* Meets the requirements that guided its design and development,
* Responds correctly to all kinds of inputs,
* Performs its functions within an acceptable time,
* Is sufficiently usable,
* Can be installed and run in its intended [environments](https://en.wikipedia.org/wiki/Operating_environment), and
* Achieves the general result its stakeholder’s desire.

## Principles of software testing

**Principle 1:** Testing shows that there are defects present in the software  
A self-explanatory point, it states that when a project is tested, it is checked for possible defects or bugs by creating different **software testing strategies**.

**Principle 2:** Testing software exhaustively is impossible.  
This means that testing software is not possible exhaustively and instead, testers need optimum time to test an application, which is based on the risk assessment of the same.

**Principle 3:** Testing software early.  
It is imperative to start testing software as early as possible. This ensures that the defects can be captured and fixed within the stipulated time-frame, thereby allowing developers to deliver the software to the clients on time.

**Principle 4**: Clustering the defects.  
Defect clustering simply state that a small number of modules in an application contains maximum defects detected.

**Principle 5**: The Pesticide Paradox.  
When the same tests are repeated over time and again, then the test cases do not find any new bugs. This situation gives rise to a new principle known as the Pesticide Paradox. However, this can be overcome by reviewing and revising the test cases and adding new and different test cases.

**Principle 6**: Testing is dependent on context.  
This means that when you test a mobile application, it will be on different grounds than while testing a web application. Similarly, testing a Mac application will be different than testing an Android application and the likes.

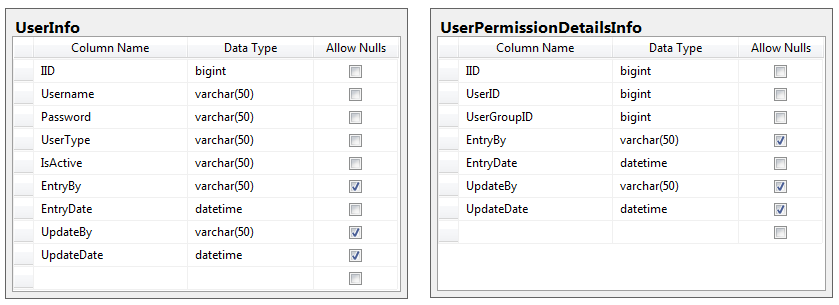
**Principle 7**: Absence of errors – fallacy.  
This principle merely points out to the fact that finding and fixing defects in a software system is of no use if the system build in itself is unusable and is unable to meet the users’ needs and requirements.

# Chapter 7

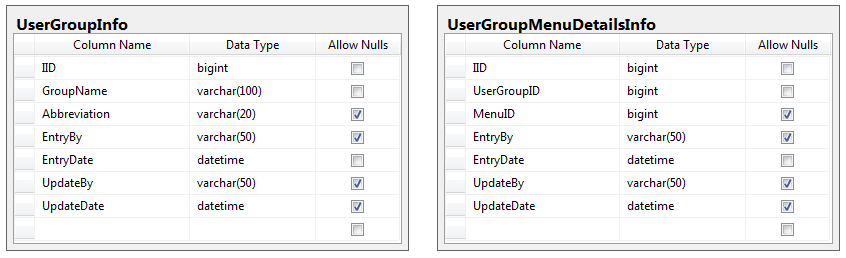
# User Manual

# Information of Table

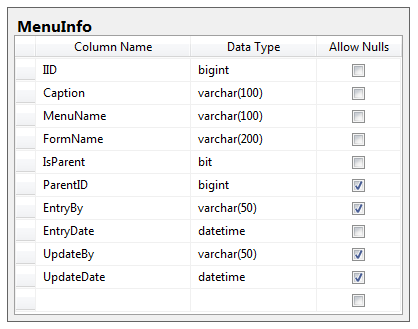
## User Table

****

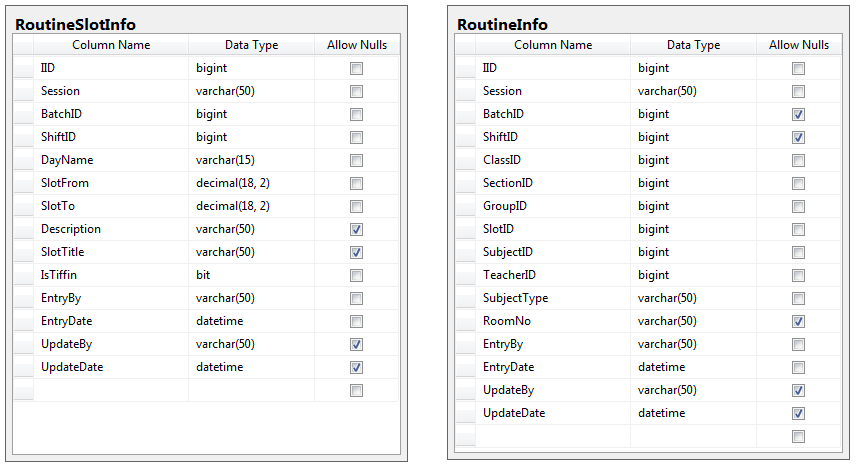
## User Group Table

****

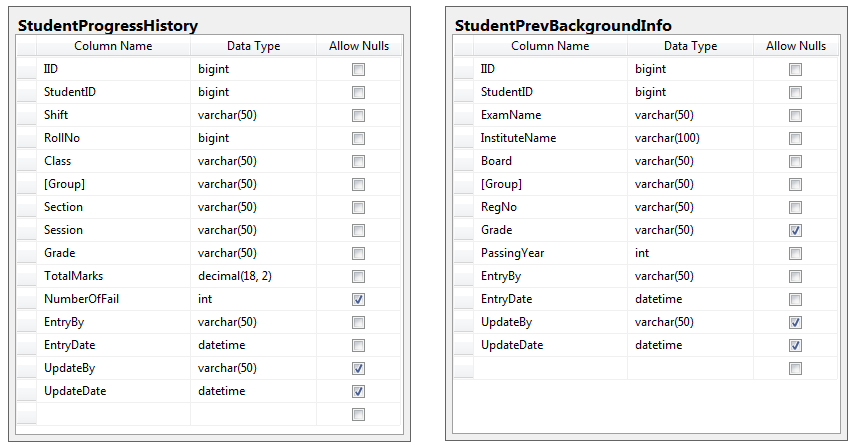
## Menu Table

****

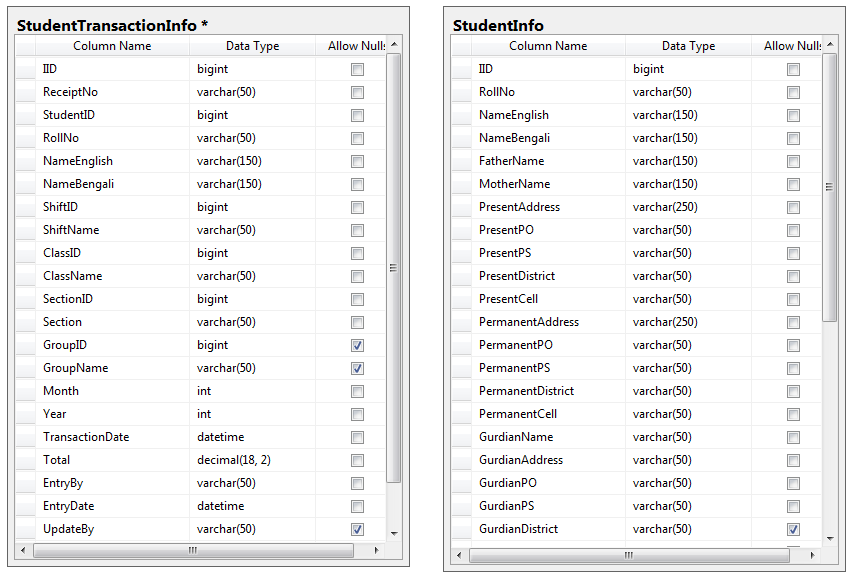
## Routine Table



## Student Progress Table



## Student Table

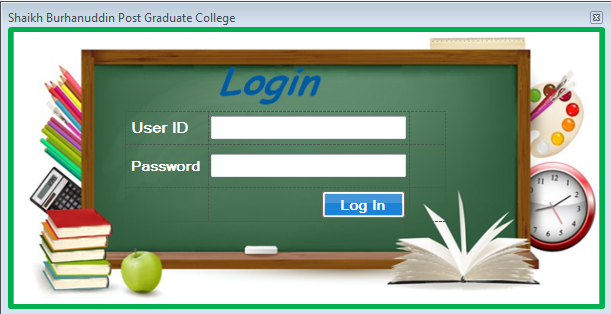


# Manual

## Login

After installing the software system user might be treated as a public user. The user only can see three functional tools bar named File, View and help. When the user clicks into option Login, A Login form appears there into interface. User has to enter the User-ID and Password to login. After typing all required info’s user have to enter login.



**

**Figure 11:** Login Process

The user must have logged in into there to have the full access. Into File tools bar there are six objective options, where the first option is Login. After Login user can see and access with all other functionalities.

## User Management

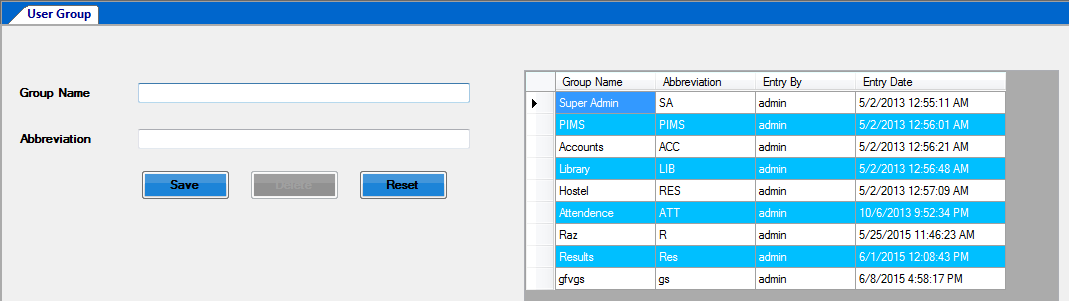
This system shows all the functionalities of a User. They Are user Group, Group Permission and User Entry.



**Figure 12:** Administration

### User Group

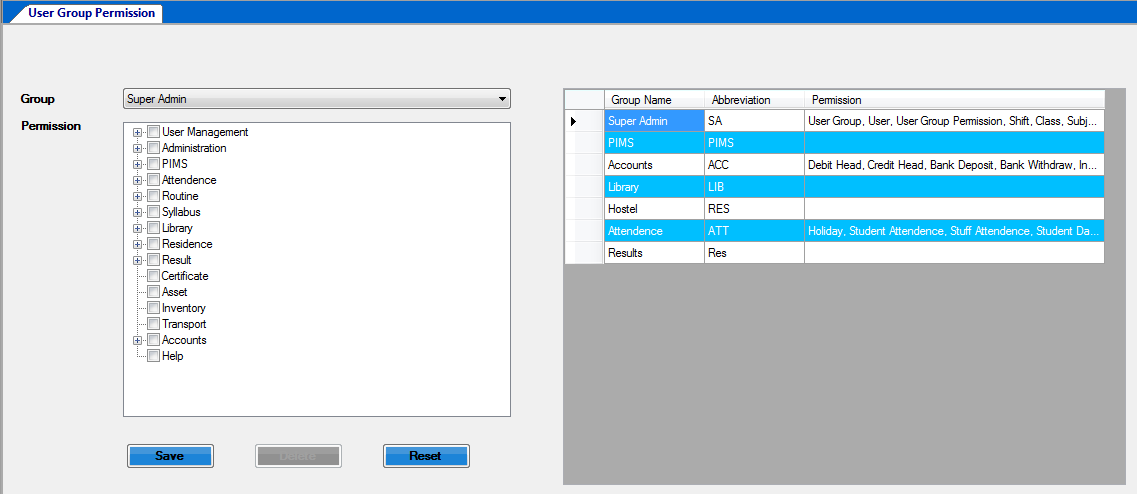
The User can Create or Delete any User Groups. User can also Reset the existing user Groups. User needs just a Group name and abbreviation for Creating a User Group. When a User finished creating a User Group, It appears at the right side of the User Group Form with its Entry by and Entry Date.



**Figure 13:** User Group Entry

### Group permission

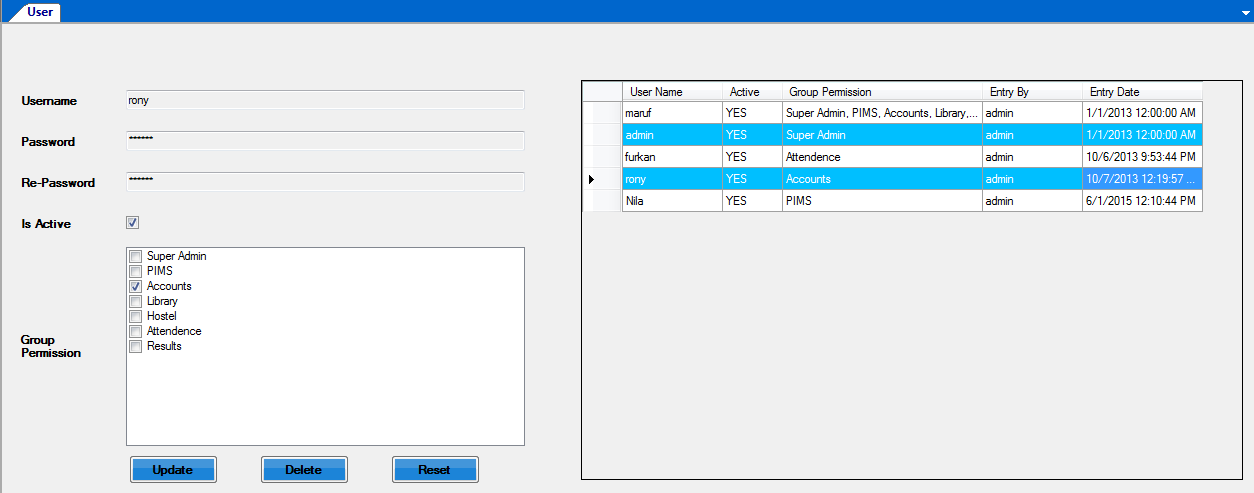
The Groups, user created into the User group can’t access into all the sections. A User Group can access only those sections whose are given. All User Groups cannot access all sections. So How many sections and which sections are permitted to be accessed by a User Group is Decided here.



**Figure 14:** Group Permission

### User Entry

The Super admin can add other users into this system if necessary. It is quite foggy to manage all the systems alone, so user can add several users. There just needs a valid User-name and a password. User must select the Sections which are permitted to that New User.

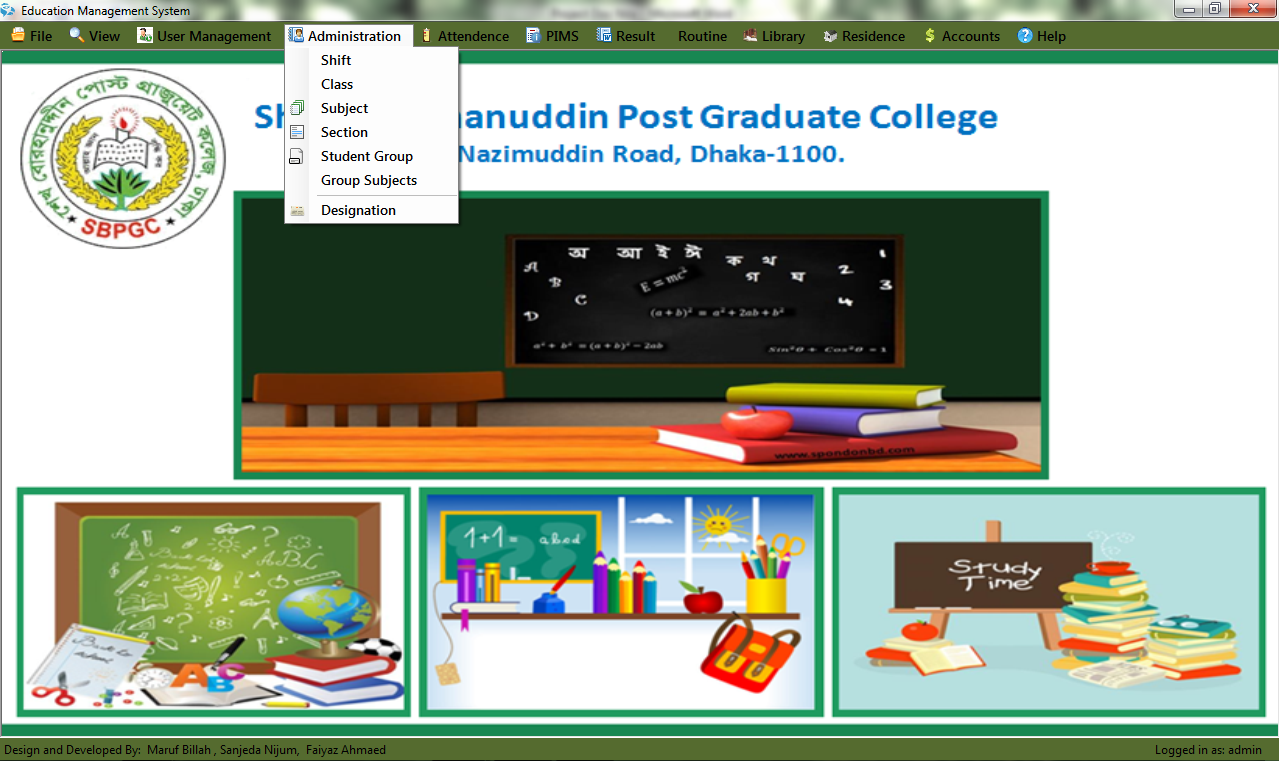


**Figure 15:** User Entry

The User can be active or non-active. After creating the User-Ids, it will appear into the right side of the form with some relational information’s active, group permission, Entry By and Entry date. We can also reset/Update the info’s if needed.

## Administration

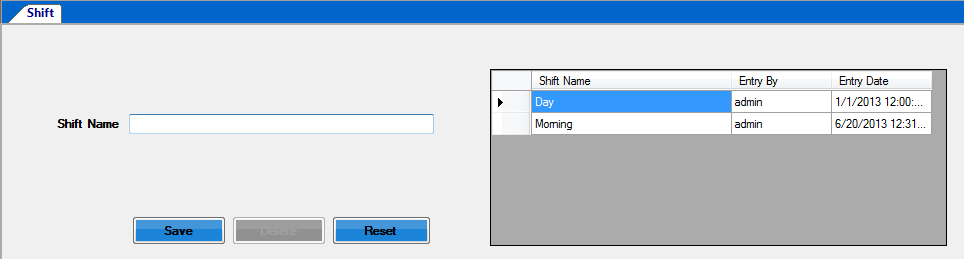
The system management shows all the functionalities of arranging and organizing classes, shifts and sections in a polite and easy way. It shows shifts, classes, sections, subject groups, group subjects and designation when necessary to know.



**Figure 16:** Administration

### Shift

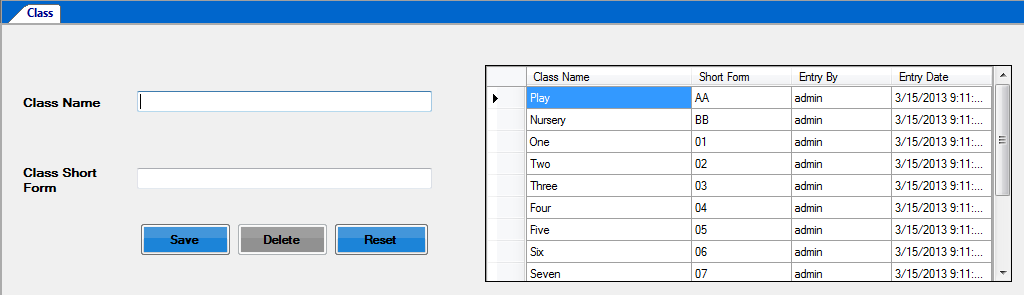
It shows the shifts of any school management system. It May b Day or Morning. Nowadays some Institutions are now started night shifts also.



**Figure 17:** Shift Entry

### Class

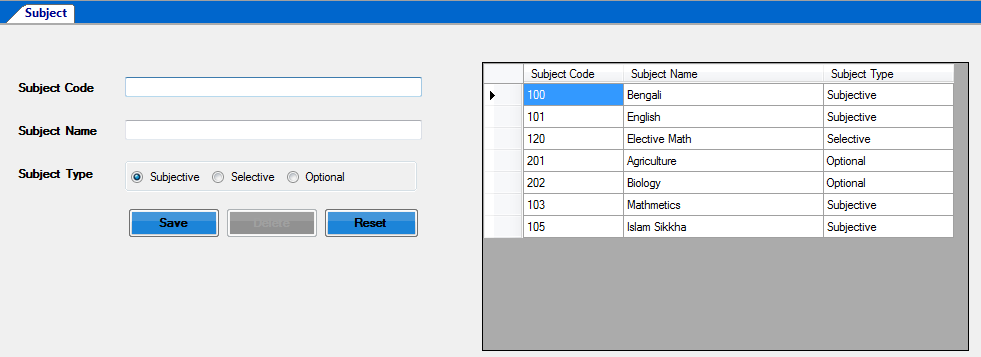
This UI is for enter all existing class in an educational institute. We also assign a Class Short for easy handling.



**Figure 18:** Class Entry

### Subject

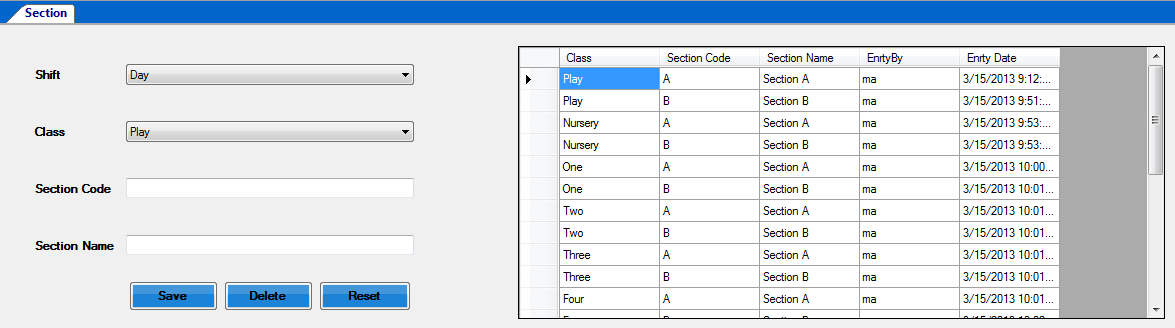
This is Input screen for enlistment of subject with subject code and subject type alike – subjective/ selective.



**Figure 19:** SubjectEntry

### Section

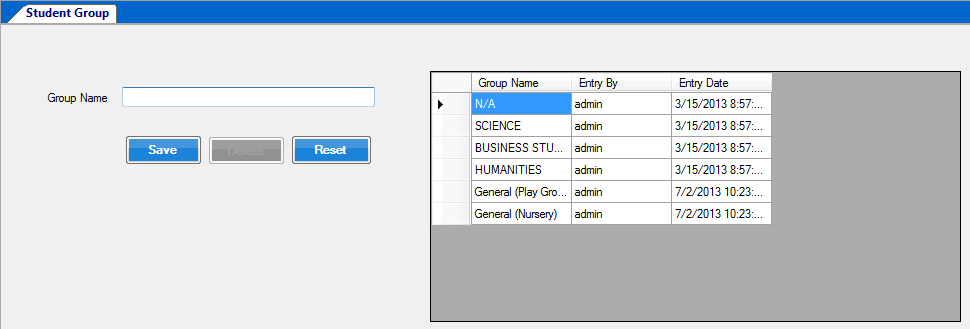
User can see all the sections of all classes with their individual section codes. If the user wants to add a new section for any identical class and shifts it needs just a section name with its section codes. After saving it will appear into the right side of user interface. User also can delete or reset any section data if needed.

****

**Figure 20:** Section Entry

### Student Group

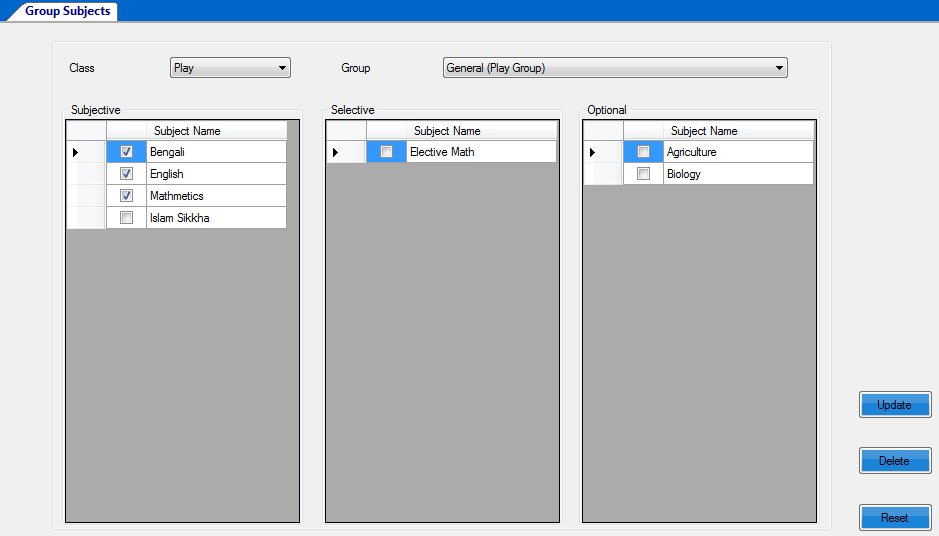
Student group shows the group of any student. User can create any user group with just only a Group-name. Usually there are 3 student groups named Science, Business Communication and Humanities.



**Figure 21:** Student Group Entry

### Group Subjects

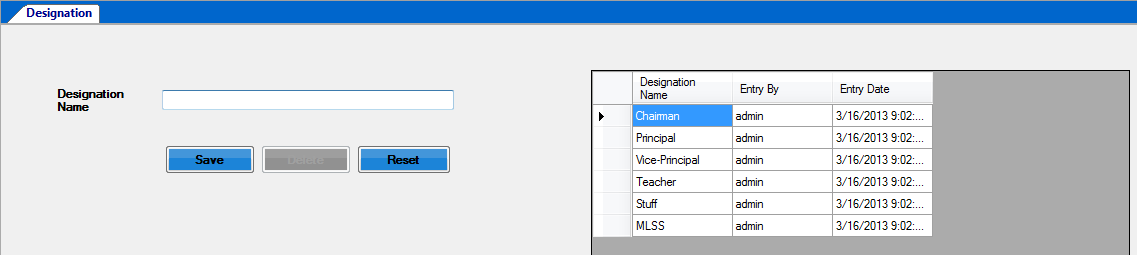
We assign subject name for under subjective, Selective and optional just select particular class and group.



**Figure 22:** Student Group Entry

### Designation

Designation means those, with whose strong monitoring and love a institution runs properly. With the help of this, User could see the entire designations name at a time. User can add other designation name according to necessity. User also can reset or delete the previous data/record if needed.



**Figure 23:** Designation

## Attendance

### Student Attendance

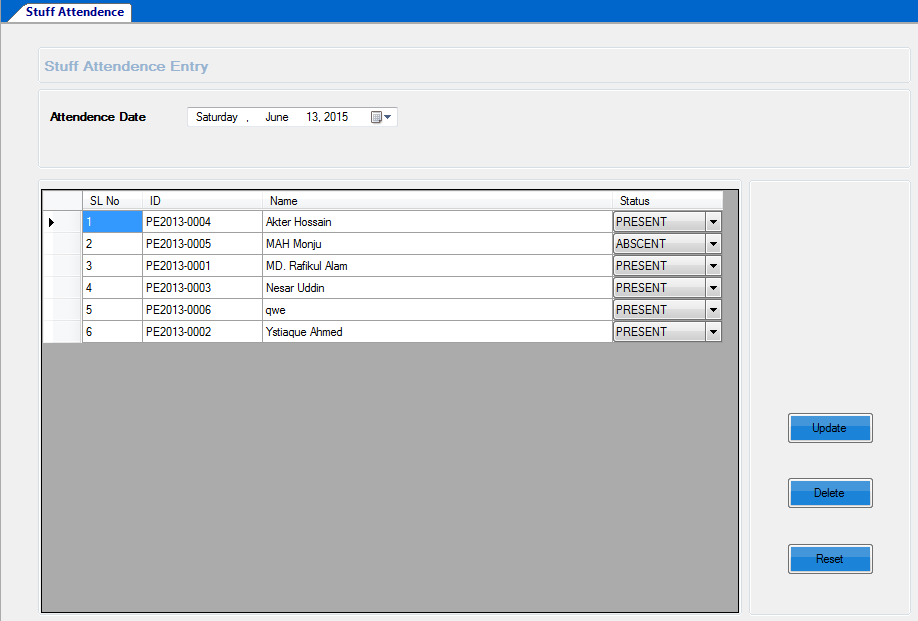
This system shows the student attendance. If the User just want to check any required class’s and section’s students attendance of a desired day, User only have to enter the class name, section name and that date, Ant may b the attendance report of a huge number of students would be shown in a couple of minute. User can save, delete or reset the data’s if needed.



**Figure 24:** Attendance Screen

### Employee Attendance

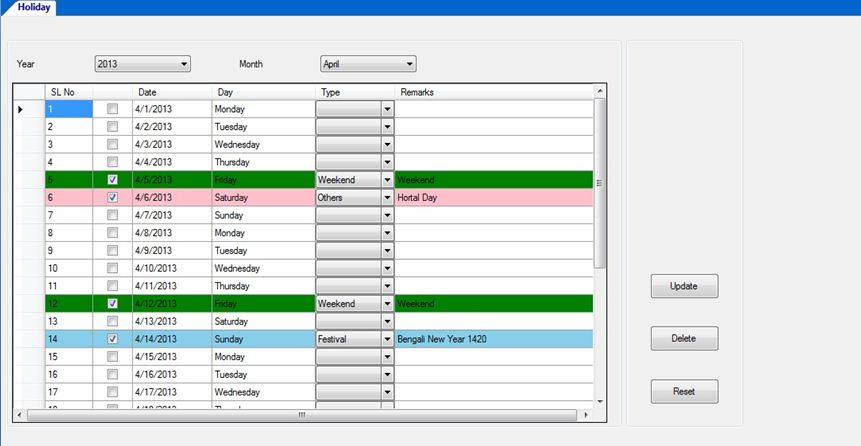
This functionality shows all of the employee’s attendance as like students and input attendance.



**Figure 25:** Employee Attendance

### Holidays

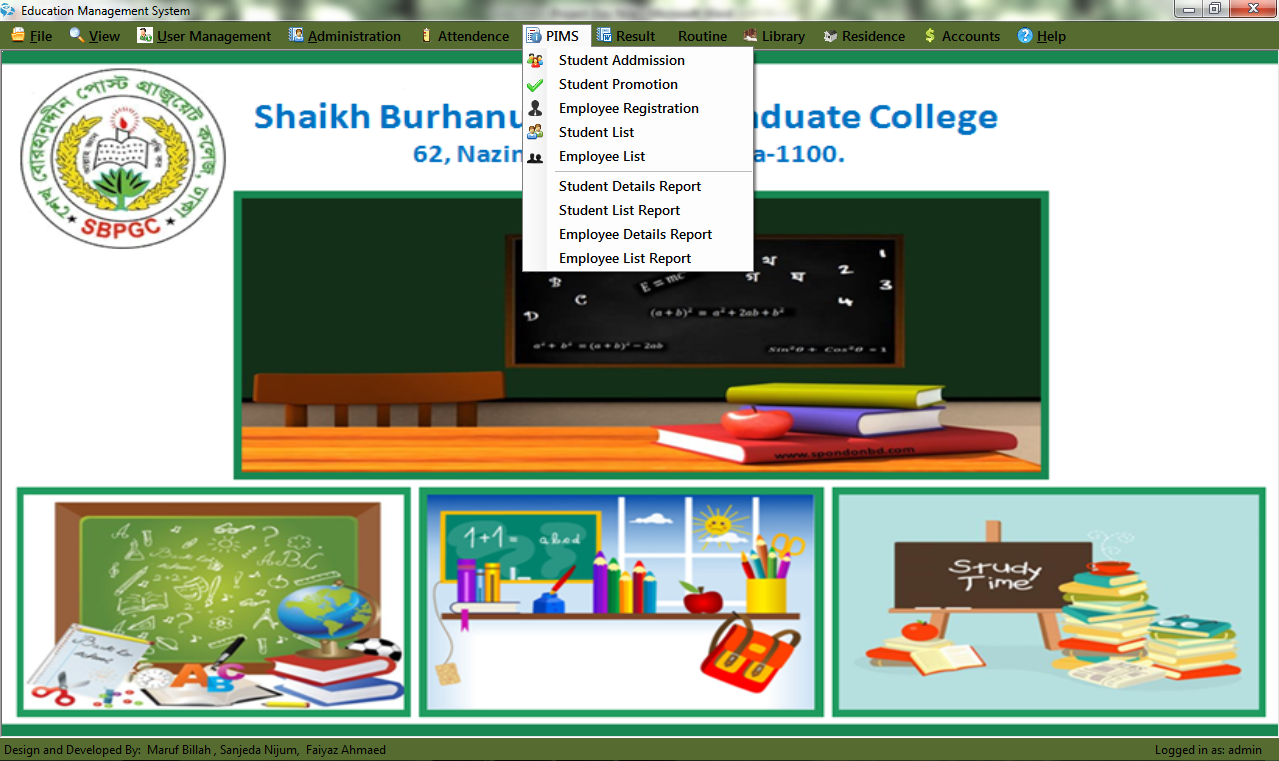
Holiday input screen is for institute can inputted their festival time and weekend for easy to concern anybody with short notes about it like remarks. Friday is by default holiday.



**Figure 26:** Enter Holidays

## PIMS (Personal Information Module System)

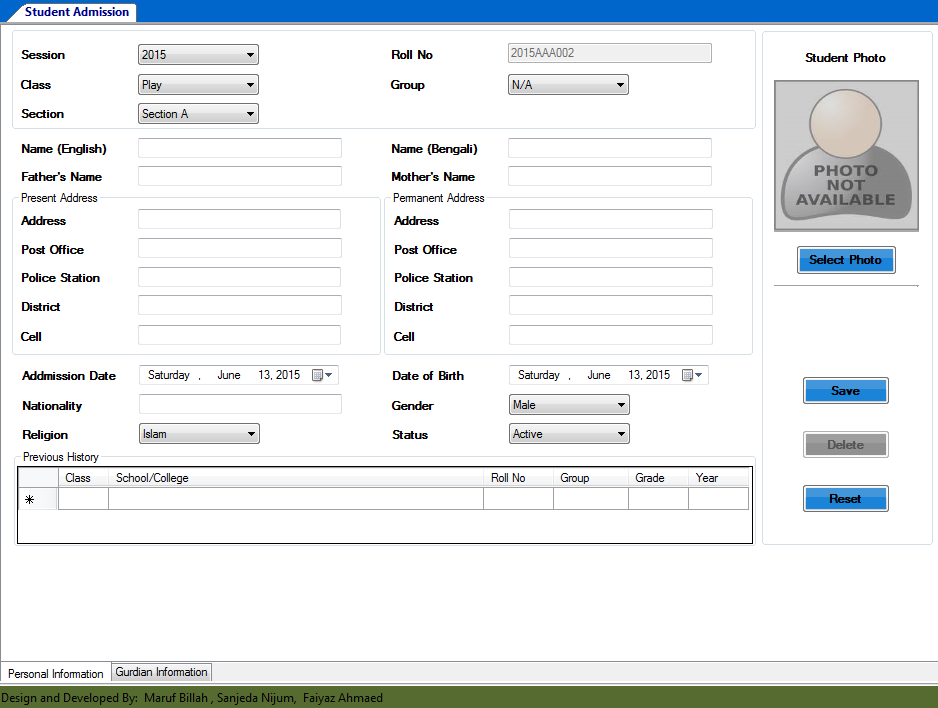
This Functionality represents Personal Information Module System. It includes/shows all kind of student and employee in information, personal or professional. Student Admission, Student promotion, Employee Registration, Student List, Employee List, Student Details report, student list Report, Employee Details Report and Employee List report all are given here if needed.



**Figure 27:** PIMS

### Student Admission

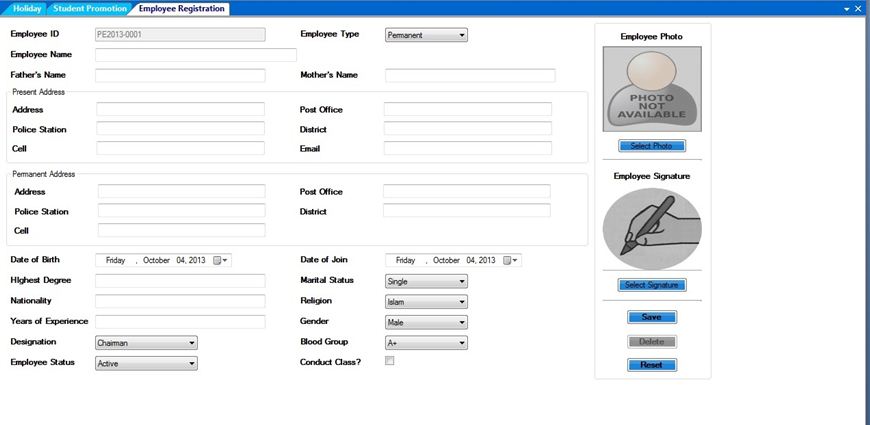
As this is a School management System there must be students, and also there must have an organized way to admit them. At here, User can admit students with fulfilling this form. After getting into all information user can save them as students of that institutions. It also include Students photo on it. It’s easy to identify. Guardian’s information is available with mobile number and photo.



**Figure 28:** Student Admission

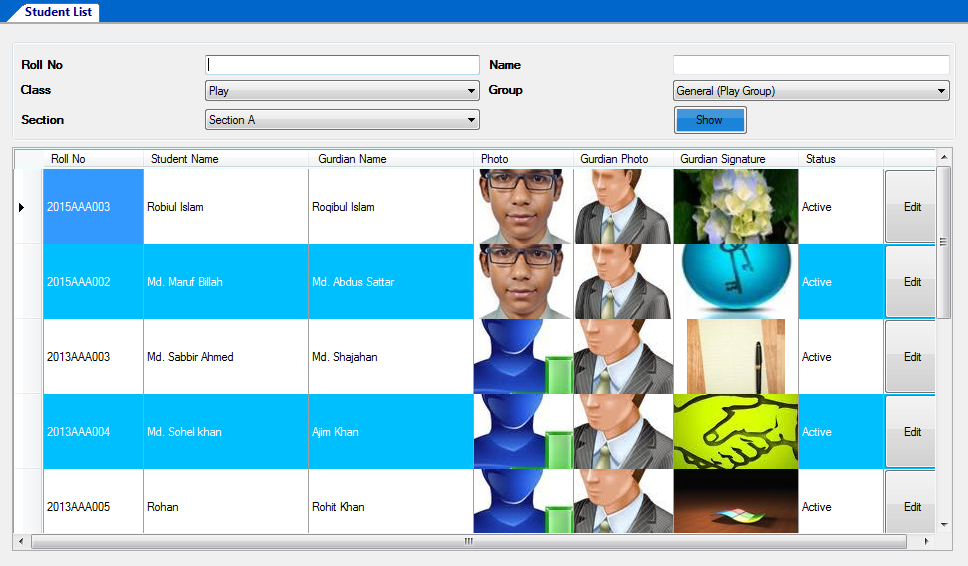
### Employee Registration

Admin can Register Employs according to necessity of management system. User can also update/Reset the old employ’s data’s if necessary. It saves the employs photo and digital signature which is more secure and identical.

**Figure 29:** Employee Registration

### Student List

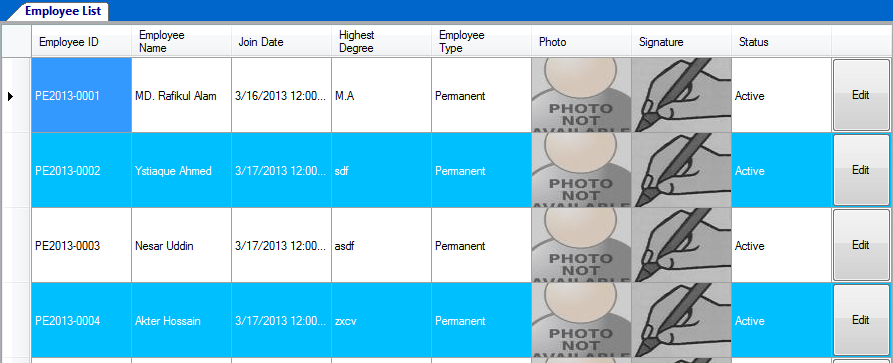
Admin / User can watch the student list. According to a School Management system, there are a lot of Students. So, may be User want to watch only some selective students list. As it, users have just input the required class and section onto it and here result comes. Admin can edit any information from here just click edit button and update data.



**Figure 30:** Student List

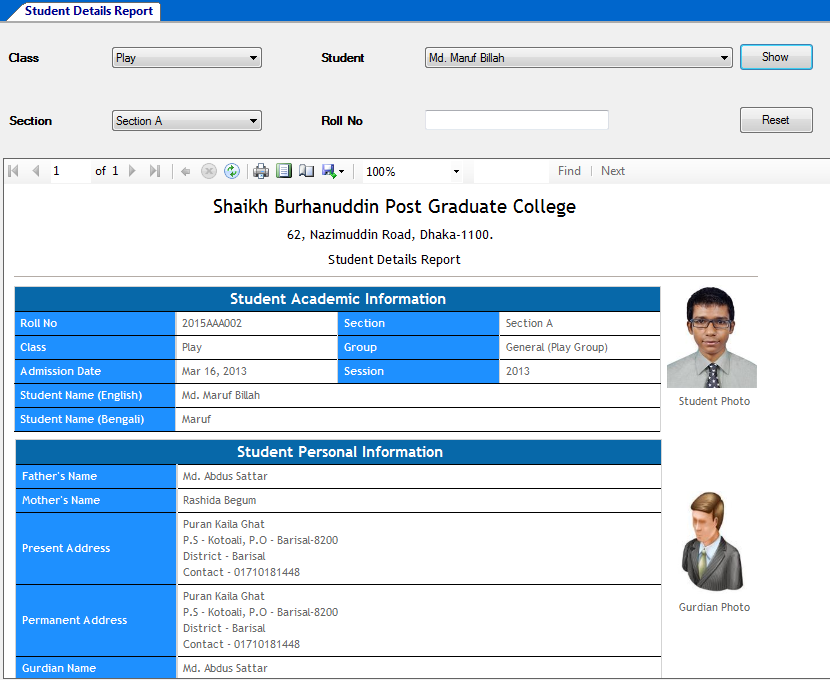
### Employee List

It shows the entire employee’s data and information with their photo and digital signatures.

**Figure 31:** Employee List

### Student Details Report

User can see the full details report of any particular student if needed from student list. It is depicted into figure 22.



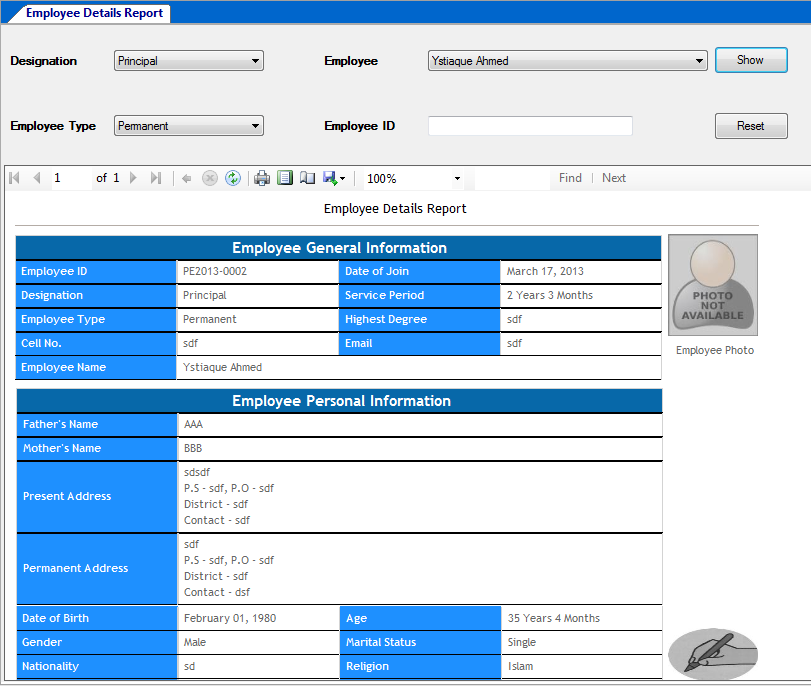
**Figure 32:** Student Details Report

### Student list Report

User can also see the Student list Report from student list report class wise.

### Employee Details Report

User can watch a particular employ’s details report here like their Educational Information, Professional Information, Skill and training Info’s.



**Figure 33:** Employee List

### Employee List Report

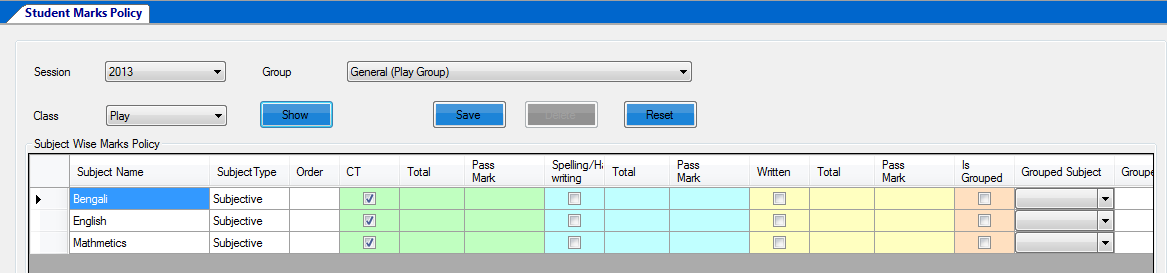
Admin/ User can view employee list report from Employee List. It can be filtered by Designation, Marital Status or Blood group.

## Result

**Figure 34:** Result Module

### Students Marks Policy

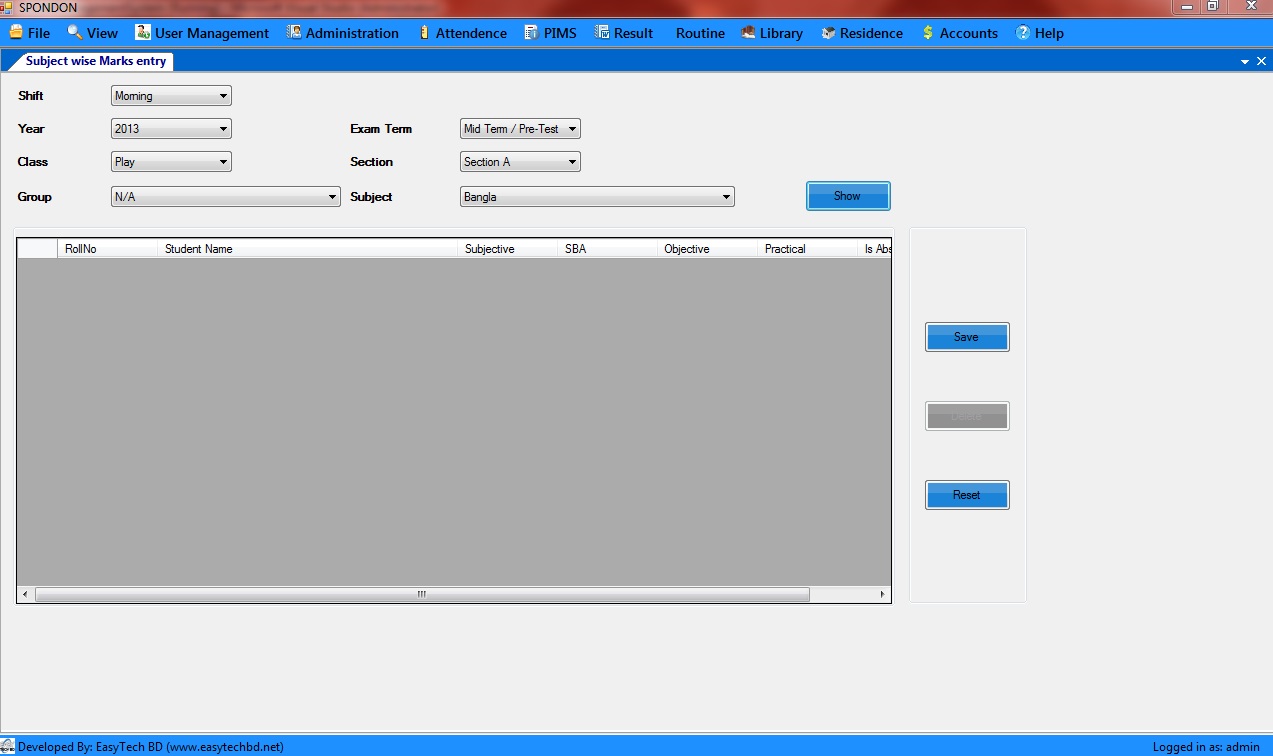
Every Educational Institute has its own marks policy. So they can input their marks policy by this UI. Simply Select class, session and groups then show. Under all subject they can assign marks policy like – have class test or not, class test total marks, pass marks, is marks in hand writing/ spelling, written pass marks, total pass marks etc.



**Figure 35:** Students Marks Policy

### Subject Wise Marks Entry

Subject Wise Marks Entry is marks entry for individual subjects marks entry. When select particular shift, class, year, group, exam-term and subject then its show all student list in grid view. It’s an easy UI for user who entry the marks. He simply input the marks for particular student and save it. If anybody absent in exam then it should be marks in checkbox. When result is published those student result shows automatically absent in those particular subject.



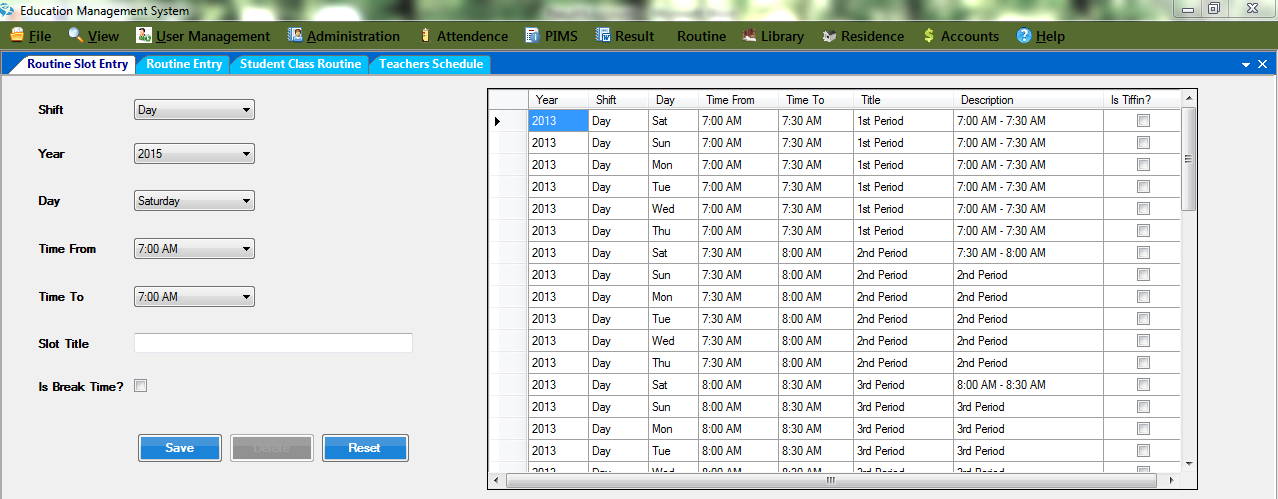
**Figure 36:** Subject Wise Marks Entry

## Routine

**Figure 37:** Routine Module

### Routine Slot Entry

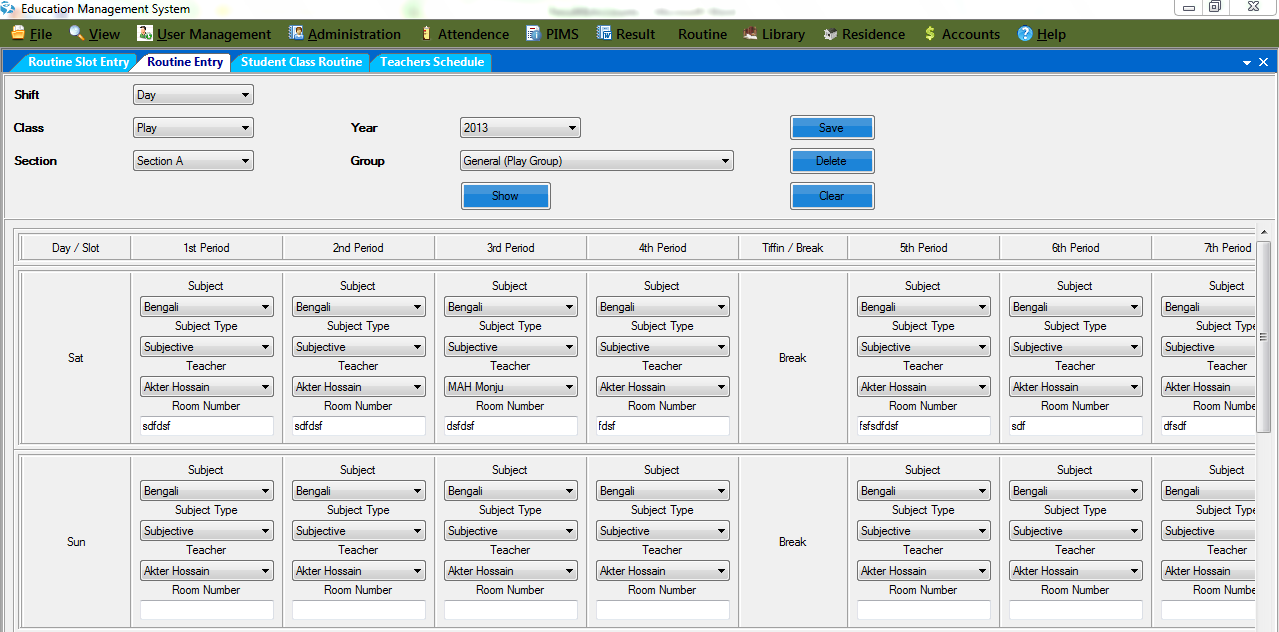
It’s a slot for routine entry. It’s just easy to entry the routine. Just select Shift, Year, Day and Time From (7:00 AM) – Time To (7:30 AM) and Slot Title (1st period). Time Slot is individual period time (7:00 AM- 7:30 AM). Almost these period are generated in routine which we entry in Routine Slot Entry UI. Break Time is also assignable by mark the checkbox is break.



**Figure 38:** Routine Slot Entry

### Routine Entry

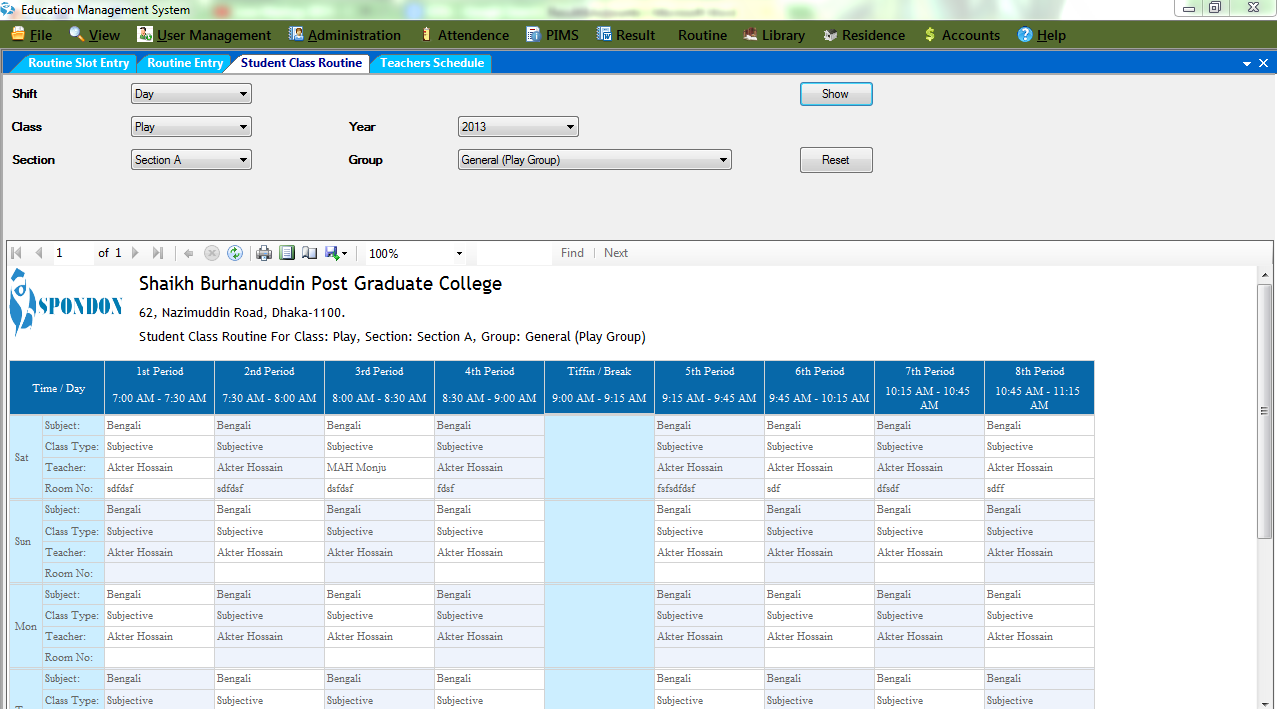
Routine Entry view’s as day and period wise routine for particular Class, Shift and Year. We just select Subject Name, Subject Type, Teacher’s Name and assigns room number for each and every period and individual day. It’s a weekly view. When all period’s and days are inputted correctly then save the data for next step.



**Figure 39:** Routine Entry

### Student Class Routine

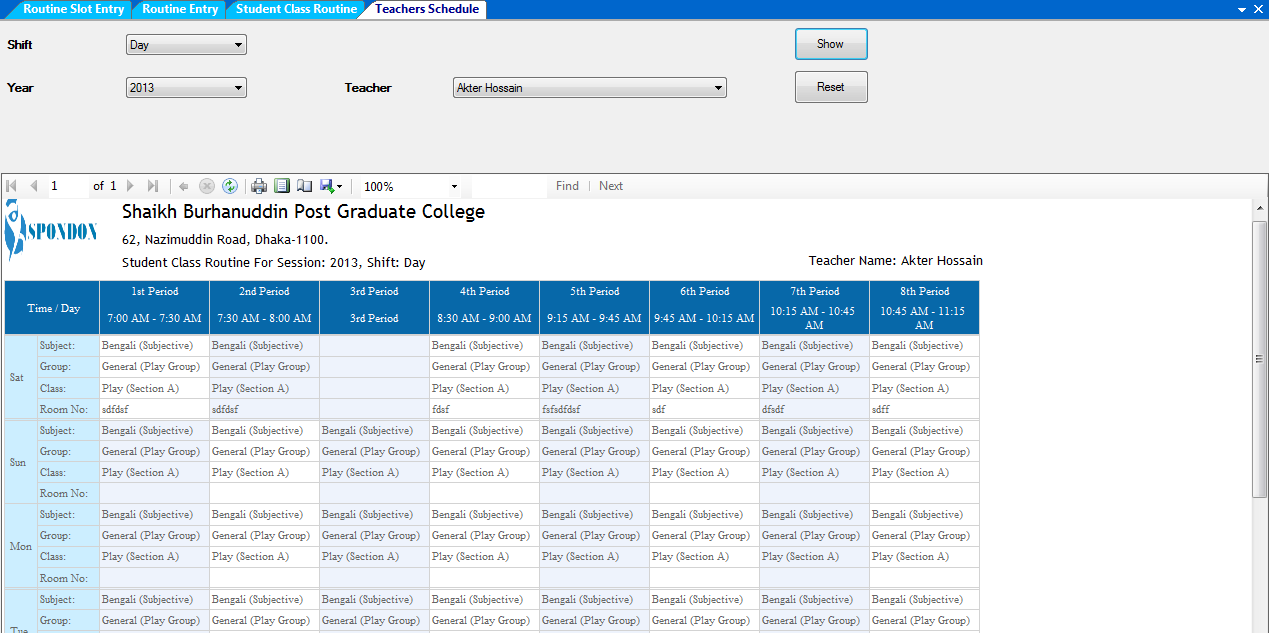
It generated the student Class Routine. Just simply select Shift, Class, Section, Year and Group and then show. We can print this report from here for student use and also export to pdf and excel.



**Figure 40:** Student Class Routine

### Teacher’s Schedule

This UI is generated Teacher’s Schedule. Just Select Shift, Year and Teacher Name and show the routine. It’s a class schedule for individual teachers with date –time and room number. Teacher’s can print their schedule for further use and also make pdf or excel file from here.

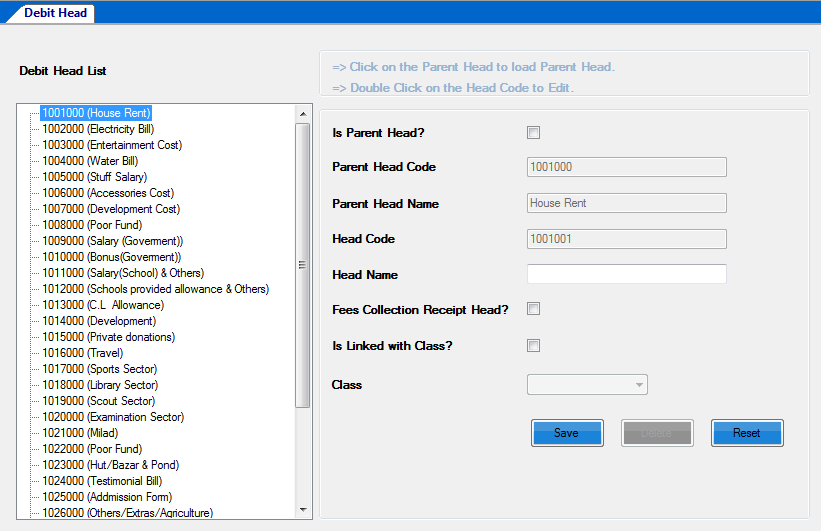


**Figure 41:** Teacher’s Schedule

## Accounts

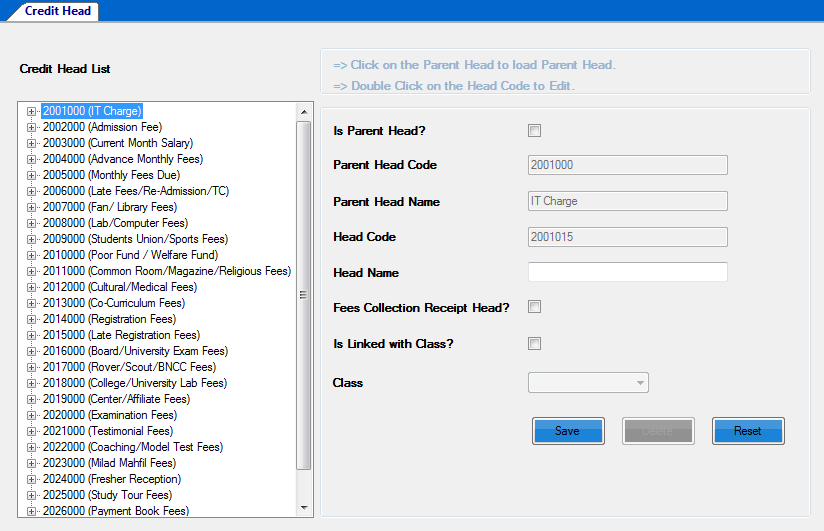
### Debit Head

We can create all expense head as Debit Head. We create debit head as parent head. If we have a child head under parent head then mark is parent head. If this head is linked with class then select linked with class and select class.

**Figure 42:** Debit Head

### Credit Head

We can create all income head as Credit Head. We create credit head as parent head. If we have a child head under parent head then mark is parent head and under parent head we create child head. If this head is linked with class then select linked with class and select class.

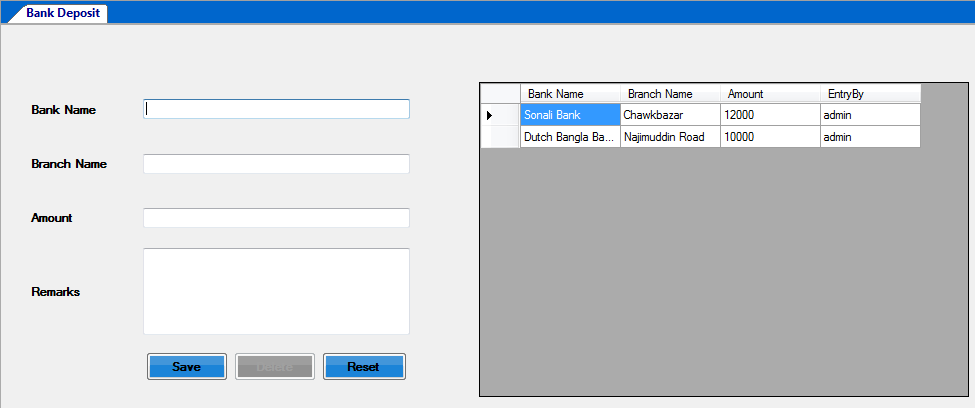


**Figure 43:** Debit Head

## 

### Bank Deposit

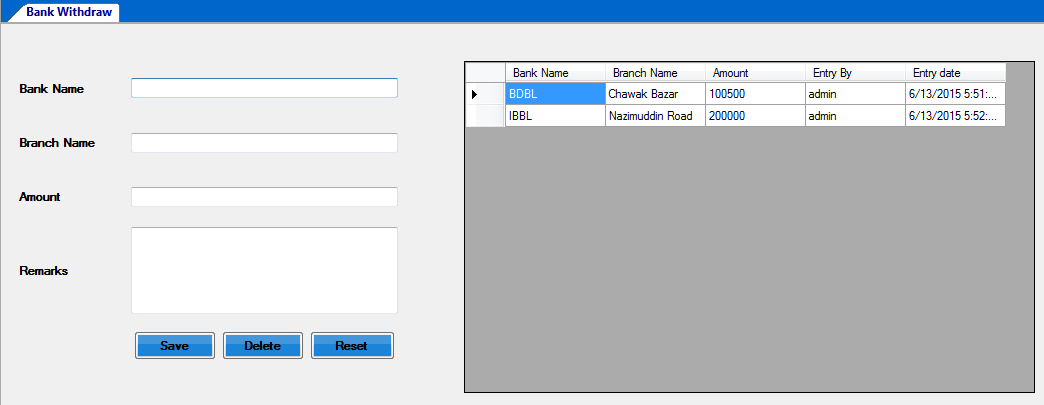
When an institute deposits any amount of money into a bank they can input deposit information into this system like bank name, branch name, deposit amount with time and date. Institute can use this information for any further query.



**Figure 44:** Bank Deposit

### Bank Withdraw

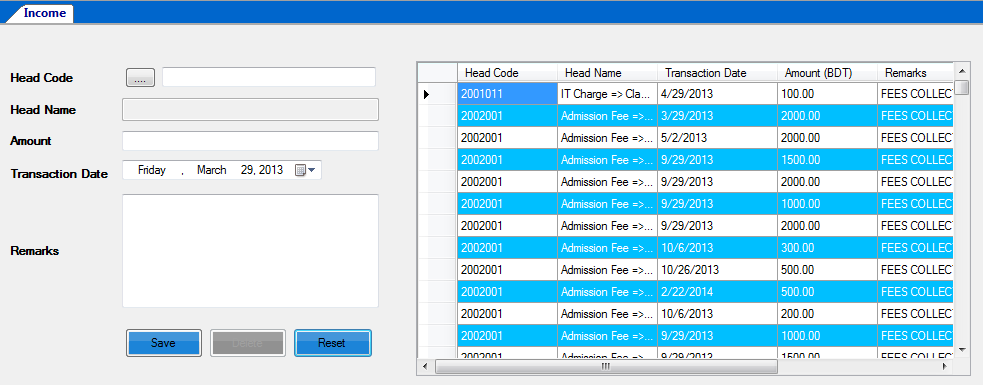
When an institute withdraws any amount of money from a bank they can input withdraw information into this system like bank name, branch name, deposit amount with time and date. Institute can use this information for any further query.



**Figure 45:** Bank Deposit

### Income

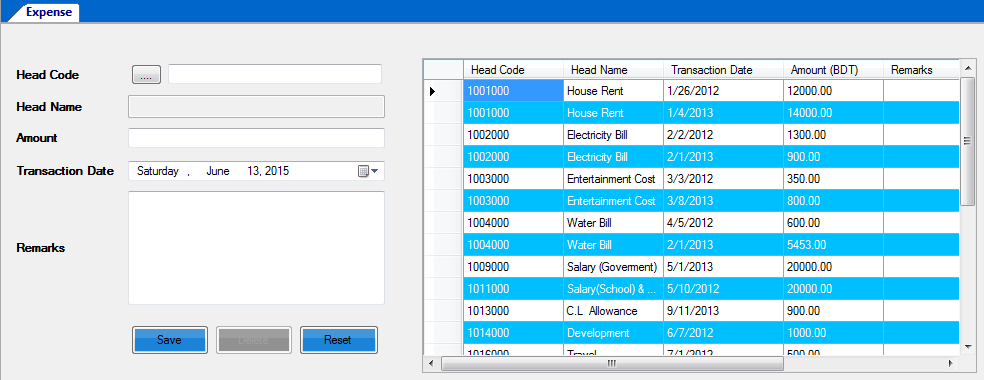
Income UI is most crucial in Accounts module. By income UI we save income source with credit head and amount of income. We simply click into head code and select credit head code for income source. Head name is automatically inputted into head name. Income is need for all accounts report.



**Figure 46:** Bank Deposit

### Expense

Expense UI is also crucial in Accounts module. By expense UI we save expense source with debit head and amount of income. We simply click into head code and select debit head code for income source. Head name is automatically inputted into head name. Expense is need for all accounts report.



**Figure 47:** Bank Deposit

# Chapter 9

# Future Works and conclusion

# Future Expandability

We are analysis User Management, Student Profile Management, Routine Management, Result Management, Employee Management and Accounts Management etc. We are complete software that has included this entire requirement.

All good software may have some limitations this software may be some limitations. We are trying to fix it as per possible.

## Conclusion

The application Education Management Software is the total package for the school or college management system requirements. With the help of menu bar users can interact with software very easily almost every object has been provided with tool bar.

This application is also supports terminal services so that database will be more secure by centralized the database.  
In fine we want to say we are not yet an experienced system analyst, but tried our best to fulfill this project.

# References

[www.w3school.com](http://www.w3school.com)

[www.wikipedia.org](http://www.wikipedia.org)

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

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

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

[www.click1me.com](http://www.click1me.com)