**CONTENTS**

* **Introduction 02**
* **Objectives & Scope of the project 02**
* **Theoretical Background Definition of Problem 03**
* **System Analysis & design 12**
* **User Requirement s System Planning (PERT Chart) 16**
* **Methodology adopted 16**
* **System Implementation 17**
* **System Maintenance & Evaluation 32**
* **Cost & Benefit analysis 36**
* **Detailed Life Cycle of the Project**

**ERD, DFD 04**

**Input and output screen design 39**

**Process involved 41**

**Methodology used for design 50**

**Test Report, Printout of the Reports, Print out of code sheet 53**

* **User/Operational Manuals 63**
* **Conclusion 66**
* **Future enhancement 66**
* **References 66**

**Introduction**

This project is aimed at developing a web based leave management, which is of importance to any organization or a college

This application is used to automate the leave applications and their approvals. It mainly deals with details of all staffs both teaching and non-teaching. A staff can submit a leave by checking his current leave balances. The clerk verifies the submitted leave and checks the available leave balances for accepting or rejecting the leave and send it to HOD. There are features like notification, cancellation of leave, approval of leave, report generators etc in this tool

**Objective and Scope of the Project**

**Objective:**

* While taking leave staff can manually fill leave-request form and submit it to their higher authority
* At the end of each month clerk can check leave balances of all staff
* It increases the paper work and makes record maintenance tedious
* To overcome the old process we are going to implement the new application
* Stored information can be retrieved in a single click
* Easy to maintain the all staff records without handlingfiles

Hence Our Project “Leave Management System “has been developed to maintain and get records with a single click and this project has modules like:

* Admin
* Staff
* HOD
* Principal

These all users add their related information using the system and hence can maintain and fill the information whenever required.

1. ­Admin-Add staff, Add leave details, View status
2. Staff-Apply leave, View status
3. HOD-Apply leave, View status
4. Principal-Sanction and Cancel leave and View status

**Scope:**

The main scope of the application is used to automate the leave applications and their approvals.Where we can get required information on the finger tips.

**TheoreticalBackgroundDefinition of the Problem**

**Existing System**

In existing system, staff can manually fill a leave-request form and submit it to their respected higher authority.The clerk does the attendance and leave calculation manually at the end of each month and send it to HOD, this increases the paper work and makes the record maintenance tedious.

**Proposed System**

The proposed system automates the existing system. It reduces the paper work and easier record maintenance by having a database for leave maintenance. It tracks all the information of faculty, report etc and view leave transaction details, including leave balances and view approve/ refuse leave requests and easy updating of everyday leaves in the database system.

**Types of Leaves**

1. Casual leave
2. Duty leave
3. Earn leave
4. Restricted holiday
5. Commuted leave
6. Medical leave

**Hardware Requirements:**

* RAM: 1 GB
* HARD DISK: 80 GB
* PROCESSOR: PENTIUM IV OR ABOVE

**Software Requirements:**

* OS: WINDOWS 2000/XP OR ABOVE
* BACKEND:SQL SERVER 2005
* FRONT END: VISUAL STUDIO.NET 2008

**Data Flow Diagram**

The data flow diagram is an important modeling tool. It shows the use of data pictorially. DFD represents the flow of data between different transformations and processes in the system. The DFD shows logical flow of data.

**Different notations used in DFD are**

* **Process**

Processes show the system does. Each process has one or more inputs and none or more outputs. If a process does not have an output then it is considered to a Black Hole.

* **Functional Processing**

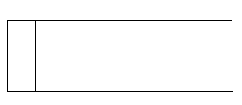
It is represented by an oval. The processing or main transactions are specified by this notation.

* **Data Flow**

It is represented by an arrow line and name of the data is specified by the side of the line as label. This is used for the data movement.

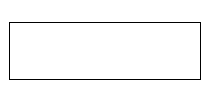
* **Data Store**

It is represented by anyone open-end rectangle. The database used in the system is specified by this notation.



* **Source or Sink**

It is represented by one open-end rectangle. It is used for specifying from where data comes and where it reaches.

****

**Context or Zero Level DFD**:

**First Level DFD**:

**High Level DFD:**



# Detailed Life Cycle of the Project

# Entity Relationship Diagram

* For database application, the entity relationship approach can be used effectively for modeling some part of the problem. The ER modeling approach is used to help design information system. The main focus of ER modeling is the data items in the system and relationship between them.
* An ER diagram consist of the following components

It represents entity sets

It represents attribute

It represents relationship set

**High Level Design:**

****

****

****

****

****

****

****

### Relational Model

Relational Schema is the core idea to describe a database as a collection of predicates over a finite set of predicate variables, describing constrains on the possible values and combinations of values. The content of the database at any given times is a finite model (logic) of the database i.e. a set of relations, one per predicate variables such that all predicates are satisfied.

The purpose of the relational model is to provide a declarative method for specifying data and queries we directly state what information the database contains and what information we want from it, and let the

Following are the steps which can be used to convert ER-Components to a relational model.

**Step 1: For regular strong entity type E in the ER-schema**

1. Create a relation R to represent strong entity E and include all simple attributes of E as attributes of R.
2. Include only simple component of composite attribute if exist in E as attribute of R.
3. Choose one of the key attribute of E as the primary key of R.
4. If the chosen key attribute of E is composite, the set of simple attribute that from it will together from the primary key of R.
5. If E contains attribute of type multivalved or complex then follow step no 6 to convert it into relational concepts.

**Step 2: For every weak entity ‘w’ in the ER schema with owner entity type ‘E’**

1. Create a new relation R to represent ‘W’. Include all simple attribute (simple components of composite attributes) of W as attribute of relation R.
2. Include as foreign key attributes in R the primary key attribute of relation that corresponds to the owner entity type.
3. The primary key of R is the combination of foreign key and the partial key of ‘W’.

**Step 3: For each binary 1:1 relationship type R in the ER relational model**

1. Identify relations S and T that corresponds to the entity types participating in R.
2. Choose one of the relation say S and include as foreign key in the primary key of T.
3. It’s better to choose an entity type with total participation in R in the role of S.
4. Include all simple attribute (or simple components of composite attributes) of the 1:1 relationship type R as attribute of ‘S’.

**Step 4: For each binary 1: N relationship type R in the ER schema or ER model**

1. Identify the relation ‘S’ that represents the participating entity type at the N side of the relationship type R.
2. Include the foreign key in S the primary key of the relation ‘T’ that represents the other entity type participating in the relationship type R.
3. Include all simple attributes (simple components of composite attributes) of the 1: N relationship type as the attribute of ‘S’.

**Step 5: For each binary M: N relationship type R in the ER schema**

1. Create a new relation ‘S’ to represent R.
2. Include as foreign key in ‘S’ the primary key of relations that represents that participating entity type in the relationship R.
3. Include any simple attribute of R as attributes of ‘S’.
4. The primary key of ‘S’ is the combination of foreign key including S.

**Step 6: For each multivalved attribute A**

1. Create a new relation R.
2. This relation R will include an attribute corresponding to A plus the primary key attribute K as foreign key in R of the relation that represents the entity type or relationship type has A as an attribute.
3. The primary key of R is the combination of A and K. If A is composite we include its simple component as attribute of R.

**System Analysis and Design**

**Use Case Model**

The use-case concept is used to present functionality of the system described in a number of use-cases, each of which represents a specific flow of events in the system. A use-case diagram is a graph of actors, a set of use cases enclosed by a system boundary, and generalization among the use-cases.

Use case diagram consists of use cases and actors and shows the interaction between them. The key points are:

* The main purpose is to show the interaction between the use cases and the actor
* To represent the system requirement from user’s perspective
* The use cases are the functions that are to be performed in the module
* An actor could be the end-user of the system or an external system



**DataTables:**

****

****

****

****

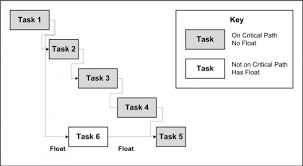
****

****

**User Requirements System Planning (PERT Chart)**

A PERT chart is a project management tool used to schedule, organize, and coordinate tasks within a project. PERT stands for Program Evaluation Review Technique.PERT charts allow the tasks in a particular project to be analyzed, with particular attention to the time required to complete each task, and the minimum time required to finish the entire project.

A PERT chart is a graph that represents all of the tasks necessary to a project's completion, and the order in which they must be completed along with the corresponding time requirements. Certain tasks are dependent on serial tasks, which must be completed in a certain sequence. PERT charts are preferable to Gantt charts because they more clearly identify task dependencies; however, the PERT chart is often more challenging to interpret. As such, project managers frequently employ both methodologies

****

**Methodologies Adopted**

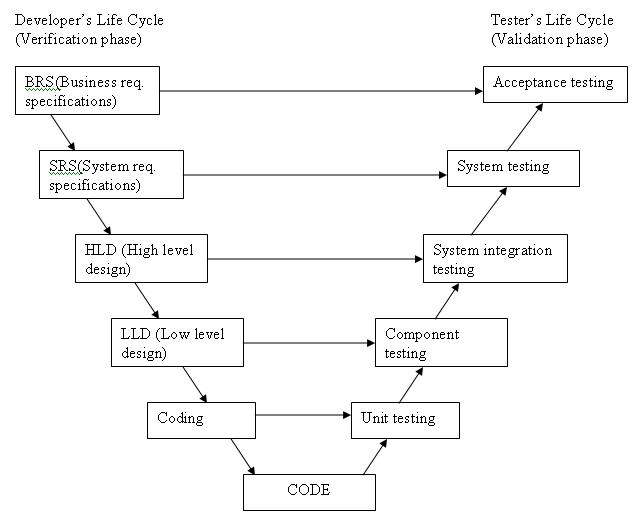
**V – MODEL:**

i) The V – Model splits testing into parts design and execution

ii) Test design is done early, while test execution done in the end

iii) There are different types of tests for each phase of life cycle

**V – Model Phases:**

****

### System Implementation

Implementation is the carrying out, execution, or practice of a plan, a method, or any design for doing something. As such, implementation is the action that must follow any preliminary thinking in order for something to actually happen. In an information technology context, implementation encompasses all the processes involved in getting new software or hardware operating properly in its environment, including installation, configuration, running, testing, and making necessary changes. The word deployment is sometimes used to mean the same thing. In this sense implementation is an ongoing process to implement this project “Leave Management System”.

## The features of the tool used to design the front end

Windows Form application for the Microsoft.NET framework using C#. The sample application demonstrates how to create and layout controls on a simple form and the handling of mouse click events. The application displays a form showing attributes of a file. This form is similar to the properties dialog box of a file (Right click on a file and Click on Properties menu item). Since attributes of a file will be shown, the sample will show how to use File IO operations in .NET framework.

Most of the C# core functionality is implemented in the System namespace. For forms application, the functionality is included in the System. Win Forms namespace. Therefore, right at the top of our source file we need to define these namespaces.

Using System;

Using System.WinForms;

Using System.Data.Sql;

These can be used to create an user friendly GUI, graphical user interface for this the application will use front end as C# .NET which involves the drag and drop options to design forms. The forms are design using labels, textboxes, buttons, image, image button, combo box, datagridview, radio buttons etc.

# The features of the db used to design the back end

For back end using MS SQL Server 2005 supports having a wide range of users access it at the same time. An instance of MS SQL Server 2005 includes the files that make up a set of database and copy of the DBMS software. Applications running on separate computer use a SQL Server 2005 communications component to transmit commands over a network to the SQL Server 2005; it can reference any of the databases in that instance that the user is authorized to access. The communication component also allows communication between an instance of SQL Server 2005 and an application running on the computer.

## The connectivity process, installation process, execution process

## Connectivity

* The MS SQL Server 2005 is used for provider for connection purpose
* Using System.Data.Client; is used for namespace operations
* Sqlconnection class is used for Connection to sqlserver

**How to load and Run**

* First we create exe file and we execute this execute this execute file in client machine
* Double click on the executable file the application will run

Software testing is a critical element of software quality assurance and represents the ultimate review of specification,design and code generation. Testing stratergies used are:

**Introduction to c#.NET**

C# programming language and .NET base class libraries using a friendly and approachable tone. The first edition of C# and .NET was published in conjunction with the release of .NET Beta 2, circa the summer of 2001.

Given that .NET is such a radical departure from the current thoughts of day, Microsoft has developed a new programming language (C#) specifically for this new platform. C# is a programming language that looks very similar (but not identical) to the syntax of java. For example, like java, a C# class definition is contained within a single-source code file (\*.cs) rather than the C++-centric view of splitting a class definition into discrete header (\*.h) and implementation (\*.cpp) files.

NET framework is completely new model for building systems on the Windows family of operating systems and in future.

The numerous benefits provided by C# and the .NET platform:

**Life As a C/Win32 API (Application Programming Interface) Programmer**

Developing software for the Windows family of operating systems involved using the C programming language in conjunction with the Windows API.

**Life As a C++/MFC (Microsoft Foundation Classes) Programmer**

C++ can be thought of as an object-oriented layer on top of c. Thus, even though C++ programmers benefit from the famed “pillars of OOP”. The main role of MFC is to wrap a “sane subset” of the raw Win32 API behind a number of classes, magic macros and numerous code wizards.

**Life As a Visual Basic 6.0 Programmer**

Many programmers have shifted away from the world of C (++)-based frameworks to kinder, gentler languages such as Visual Basics 6.0. VB is popular due to its ability to build complex user interfaces, code libraries (e.g., COM servers) and data access logic with minimal fuss and bother.

**Life As a COM (Component Object Model) Programmer**

COM is architecture that says in effect, “If you build your classes in accordance with the rules of COM, you end up with up with a block of reusable binary code”. The beauty of COM server is that it can be accessed in a language-independent manner.

**Standard controls:**

C# .NET has various drag and drop controls which are user friendly like textbox to enter the data, button is used to perform some action etc.

**Usage:**

* Text boxes are used in login page to fetch username and password from user
* Dropdown menu is used to select topic when an admin is necessary
* Button is used for save, update, close, delete etc

**Data bound controls:**

Data bound controls are used when information to be displayed is more than one row. Ex, search results then we go for data bound controls. It has many such controls like- grid view, repeater, data list, details view and form view. They all used to display information but not to insert it.

**Usage:**

* Grid view to display short all the details
* Grid view is also used here to show the all rows information

**Validation controls:**

C# .NET facilitates us with providing standard controls for validation. It provides validation controls like-required field validation, regular expression validation, range validation, compare validation &on.

**Usage:**

* In login page, user name and password are validated for required field’s validation
* In change password page required fields validation is kept for old password and new password in New profile page
* In Customer page, first name, middle name, last name, phone numbers are validated for required field’s validation
* For cell double click in data grid view in all pages are validated wherever it takes

# SQL Server 2005

C# .NET can interact with any database like Access, Oracle, MySQL, and SQL-Server. Etc, but the most popular in combination with ASP.NET is SQL-Server. This is because:

* In Visual Studio 2005 a built-in SQL-Server called SQL Express Edition is given hence we need not install any other database externally
* .NET with SQL-Server works 30% faster than other database

**Connectivity:**

For connectivity purpose SQL server needs 2 objects:

**Connection Object:** Connection object is used to connect any front-end application of .NET to a particular database like SQL Server.

For SQL Server provider is SQLOLEDB.

**Command Object:** It is use to execute any SQL Statement or stored procedures against database.

It has got 3 important methods-

i) **Execute Non-Query**- when we use commands like insert, update, Delete to add a record, to change a record or to remove a record then we should use this method along with command object.

ii) **Execute Reader**- When we use simple select statement to fetch one or more records. This method is used along with command object. To store the fetched rows we have to use an oledbreader object (dr). We can fetch first row by dr [0].

iii) **Execute Scalar**- when we use select statement with group Functions like Sum, Average, Min, Max, and count which return only one value from database.

**Usage:**

* Execute non query is used to perform insert, delete and update functions for all add, delete and edit forms respectively
* Execute reader is used to fetch values from table

**Microsoft.NET Framework**

The .NET Framework is a new computing platform that simplifies application development in the highly distributed environment of the Internet. The .NET Framework is designed to fulfill the following objectives:

* To provide a consistent object-oriented programming environment whether object code is stored and executed locally, executed locally but Internet-distributed, or executed remotely
* To provide a code-execution environment that minimizes software deployment and versioning conflicts
* To provide a code-execution environment that guarantees safe execution of code, including code created by an unknown or semi-trusted third party
* To provide a code-execution environment that eliminates the performance problems of scripted or interpreted environments

The .NET Framework has two main components: the common language runtime and the .NET Framework class library. The common language runtime is the foundation of the .NET Framework. You can think of the runtime as an agent that manages code at execution time, providing core services such as memory management, thread management, and remoting, while also enforcing strict type safety and other forms of code accuracy that ensure security and robustness. The class library, the other main component of the .NET Framework, is a comprehensive, object-oriented collection of reusable types that you can use to develop applications ranging from traditional command-line or graphical user interface (GUI) applications to applications based on the latest innovations provided by ASP.NET, such as Web Forms and XML Web services.

The following illustration shows the relationship of the common language runtime and the class library to your applications and to the overall system. The illustration also shows how managed code operates within a larger architecture.

**Features of the Common Language Runtime:**

The common language runtime manages memory, thread execution, code execution, code safety verification, compilation, and other system services. These features are intrinsic to the managed code that runs on the common language runtime.

The runtime also enforces code robustness by implementing a strict type- and code-verification infrastructure called the common type system (CTS). The CTS ensures that all managed code is self-describing. The various Microsoft and third-party language compilers generate managed code that conforms to the CTS. This means that managed code can consume other managed types and instances, while strictly enforcing type fidelity and type safety.

Finally, the runtime can be hosted by high-performance, server-side applications, such as Microsoft® MS Access™ and Internet Information Services (IIS). This infrastructure enables you to use managed code to write your business logic, while still enjoying the superior performance of the industry's best enterprise servers that support runtime hosting.

**.NET Framework Class Library**

The .NET Framework class library is a collection of reusable types that tightly integrate with the common language runtime. The class library is Object Oriented, providing types from which your own managed code can derive functionality. This not only makes the .NET Framework types easy to use, but also reduces the time associated with learning new features of the .NET Framework.

For example, the .NET Framework collection classes implement a set of interfaces that you can use to develop your own collection classes. Your collection classes will blend seamlessly with the classes in the .NET Framework.

In addition to these common tasks, the class library includes types that support a variety of specialized development scenarios. For example, you can use the .NET Framework to develop the following types of applications and services:

* Console applications
* Scripted or hosted applications
* Windows GUI applications (Windows Forms)
* ASP.NET applications
* XML Web services
* Windows services

For example, the Windows Forms classes are a comprehensive set of reusable types that vastly simplify Windows GUI development. If you write an ASP.NET Web Form application, you can use the Web Forms classes.

**Client Application Development**

Client applications are the closest to a traditional style of application in Windows-based programming. These are the types of applications that display windows or forms on the desktop, enabling a user to perform a task. Client applications include applications such as word processors and spreadsheets, as well as custom business applications such as data-entry tools, reporting tools, and so on. Client applications usually employ windows, menus, buttons, and other GUI elements, and they likely access local resources such as the file system and peripherals such as printers.

In the past, developers created such applications using C/C++ in conjunction with the Microsoft Foundation Classes (MFC) or with a rapid application development (RAD) environment such as Microsoft® Visual Basic®. The .NET Framework incorporates aspects of these existing products into a single, consistent development environment that drastically simplifies the development of client applications.

**ASP.NET**

ASP.NET is part of the whole. NET framework, built on top of the Common Language Runtime (also known as the CLR) - a rich and flexible architecture, designed not just to cater for the needs of developers today, but to allow for the long future we have ahead of us. What you might not realize is that, unlike previous updates of ASP, ASP.NET is very much more than just an upgrade of existing technology – it is the gateway to a whole new era of web development.

ASP.NET is a feature at the following web server releases

* Microsoft IIS 5.0 on WINDOWS 2000 Server
* Microsoft IIS 5.1 on WINDOWS XP

ASP.NET has been designed to try and maintain syntax and run-time compatibility with existing ASP pages wherever possible. The motivation behind this is to allow existing ASP Pages to be initially migrated ASP.NET by simply renaming the file to have an extension of .aspx. For the most part this goal has been achieved, although there are typically some basic code changes that have to be made, since VBScript is no longer supported, and the VB language itself has changed.

Some of the key goals of ASP.NET were to

* Remove the dependency on script engines, enabling pages to be type safe and compiled
* Reduce the amount of code required to develop web applications
* Make ASP.NET well factored, allowing customers to add in their own custom functionality, and extend/ replace built-in ASP.NET functionality
* Realize that bugs are a fact of life, as ASP.NET should be as fault tolerant as possible

**Benefits of ASP.NET**

The .NET Framework includes a new data access technology named ADO.NET, an evolutionary improvement to ADO. Though the new data access technology is evolutionary, the classes that make up ADO.NET bear little resemblance to the ADO objects with which you might be familiar. Some fairly significant changes must be made to existing ADO applications to convert them to ADO.NET. The changes don't have to be made immediately to existing ADO applications to run under ASP.NET, however.

**ASP.NET has several advantages over ASP**

 The following are some of the benefits of ASP.NET:

* Make code cleaner
* Improve deployment, scalability, and reliability
* Provide better support for different browsers and devices
* Enable a new breed of web applications

**ActiveX**

ActiveX is a specification develops by Microsoft that allows ordinary Windows programs to be run within a Web page. ActiveX programs can be written in languages such as Visual Basic and they are complied before being placed on the Web server.

ActiveX application, called controls, are downloaded and executed by the Web browser, like Java applets. Unlike Java applets, controls can be installed permanently when they are downloaded; eliminating the need to download them again. ActiveX’s main advantage is that it can do just about anything.

ActiveX has disadvantages

* It isn’t as easy to program as scripting language or Java
* ActiveX is proprietary
* It works only in Microsoft Internet Explorer and only Windows platforms

**ADO.NET**

ADO.NET provides consistent access to data sources such as Microsoft SQL Server, as well as data sources exposed via OLE DB and XML. Data-sharing consumer applications can use ADO.NET to connect to these data sources and retrieve, manipulate, and update data.

ADO.NET cleanly factors data access from data manipulation into discrete components that can be used separately or in tandem. ADO.NET includes .NET data providers for connecting to a database, executing commands, and retrieving results. Those results are either processed directly, or placed in an ADO.NET Dataset object in order to be exposed to the user in an ad-hoc manner, combined with data from multiple sources, or remote between tiers. The ADO.NET Dataset object can also be used independently of a .NET data provider to manage data local to the application or sourced from XML.

**Why ADO.NET?**

As application development has evolved, new applications have become loosely coupled based on the Web application model. More and more of today's applications use XML to encode data to be passed over network connections. Web applications use HTTP as the fabric for communication between tiers, and therefore must explicitly handle maintaining state between requests. This new model is very different from the connected, tightly coupled style of programming that characterized the client/server era, where a connection was held open for the duration of the program's lifetime and no special handling of state was required.

In designing tools and technologies to meet the needs of today's developer, Microsoft recognized that an entirely new programming model for data access was needed, one that is built upon the .NET Framework.

**Leverage Current ADO Knowledge**

Microsoft's design for ADO.NET addresses many of the requirements of today's application development model. At the same time, the programming model stays as similar as possible to ADO, so current ADO developers do not have to start from scratch in learning a brand new data access technology. ADO.NET is an intrinsic part of the .NET Framework without seeming completely foreign to the ADO programmer.

ADO.NET coexists with ADO. While most new .NET applications will be written using ADO.NET, ADO remains available to the .NET programmer through .NET COM interoperability services.

**XML Support**

XML and data access are intimately tied-XML is all about encoding data, and data access is increasingly becoming all about XML. The .NET Framework does not just support Web standards-it isbuilt entirely on top of them.

**Top-10 Features of SqlServer-2005**

1. **T-SQL (Transaction SQL) enhancements**

T-SQL is the native set-based RDBMS programming language offering high-performance data access. It now incorporates many new features including error handling via the TRY and CATCH paradigm, Common Table Expressions (CTE), which return a record set in a statement, and the ability to shift columns to rows and vice versa with the PIVOT and UNPIVOT commands.

2. **CLR (Common Language Runtime)**

The next major enhancement in SQL Server 2005 is the integration of a .NET compliant language such as C#, ASP.NET or VB.NET to build objects (stored procedures, triggers, functions, etc.). It is expected to replace extended stored procedures in the SQL Server 2000 environment as well as expand the traditional relational engine capabilities.

3. **Service Broker**

The Service Broker handles messaging between a sender and receiver in a loosely coupled manner. A message is sent, processed and responded to, completing the transaction. This greatly expands the capabilities of data-driven applications to meet workflow or custom business needs.

4. **Data encryption**

SQL Server 2000 had no documented or publicly supported functions to encrypt data in a table natively. Organizations had to rely on third-party products to address this need. SQL Server 2005 has native capabilities to support encryption of data stored in user-defined databases.

5. **SMTP mail**

Sending mail directly from SQL Server 2000 is possible, but challenging. With SQL Server 2005, Microsoft incorporates SMTP mail to improve the native mail capabilities. Say "see-ya" to Outlook on SQL Server!

6**. HTTP endpoints**

You can easily create HTTP endpoints via a simple T-SQL statement exposing an object that can be accessed over the Internet. This allows a simple object to be called across the Internet for the needed data.

7. **Multiple Active Result Sets (MARS)** MARS allow a persistent database connection from a single client to have more than one active request per connection. This should be a major performance improvement, allowing developers to give users new capabilities when working with SQL Server. For example, it allows multiple searches, or a search and data entry. The bottom line is that one client connection can have multiple active processes simultaneously.

8. **Dedicated administrator connection**

If all else fails, stop the SQL Server service or push the power button. That mentality is finished with the dedicated administrator connection. This functionality will allow a DBA to make a single diagnostic connection to SQL Server even if the server is having an issue.

9**. SQL Server Integration Services (SSIS)**

SSIS has replaced DTS (Data Transformation Services) as the primary ETL (Extraction, Transformation and Loading) tool and ships with SQL Server free of charge. This tool, completely rewritten since SQL Server 2000, now has a great deal of flexibility to address complex data movement.

10. **Database mirroring**

It's not expected to be released with SQL Server 2005 at the RTM in November, but I think this feature has great potential. Database mirroring is an extension of the native high-availability capabilities. So, stay tuned for more details.

**HTML**

HTML (Hyper Text Markup Language) is the language that is used to prepare documents for online publications. HTML documents are also called Web documents, and each HTML document is known as Web page.

A page is what is seen in the browser at any time. Each Web site, whether on the Internet or Intranet, is composed of multiple pages. And it is possible to switch among them by following hyperlinks. The collection of HTML pages makes up the World Wide Web.

A web pages is basically a text file that contains the text to be displayed and references of elements such as images, sounds and of course hyperlinks to other documents. HTML pages can be created using simple text editor such as Notepad or a WYSIWYG application such as Microsoft FrontPage.

"Hypertext" is the jumping frog portion. A hyperlink can jump to any place within your own page(s) or literally to anyplace in the world with a 'net address (URL, or Uniform Resource Locator.) It's a small part of the html language.

**INTERNET INFORMATION SERVER (IIS)**

A web server is a program connected to the World Wide Web (www) that furnishes resources from the web browser.

Tools and functionality, IIS also has built-in capabilities to help administer secure websites, and to develop server-intensive web application.

**FEATURES OF IIS**

IIS provides integrated security and access to a wide range of content, work seamlessly with COM components, and has a graphical interface-the Microsoft Management Console (MMC) –that you can use to create and manage your ASP application.

**IIS Provides Integrated Security:**

On the internet, most sites allow anybody to connect to the site. The exceptions are commercialists where you pay a onetime, monthly fee to access the site. Sites that are restrict the access called secured site. Secured site use either integrated security or login, password security. IIS support both of these methods.

**IIS provides Access to Content:**

All web servers can deliver HTML files, but they differ widely in how they treat other types of content. Most servers let you add and modify Multi-purpose Internet Mail Extensions (MMIE) types, but integrate directly into the windows registry. That means IIS natively understands how to treat most common windows file format, such as text (TXT) files, application initialization (INI) files, executable (EXE) files and many others

**IIS provides an Interface FOR COM:**

You can control many parts of IIS using COM>IIS exposes many of the server’s configuration settings via the IIS Admin objects. These objects are accessible from ASP and other languages. That means you can adjust server configuration and create virtual directories and webs programmatically. IIS 4 and higher store settings and web information in a spoil database called the Metaphase. You can use the IIS Admin objects to create new sites and virtual directories be alter the properties of existing sites and virtual directories.

**IIS ARCHITECTURES OVERVIEW:**

IIS is a core product, which means that it is designed to work closely with many other products, including all products in the Windows NT Server 4.0 Option pack. The following figure shows the relationship between IIS and other products installed as part of the Windows NT Server 4.0 Option pack.

**SECURITY FOR IIS APPLICATION**

IIS provides three authentication schemes to control access to ITS resources: Anonymous, Basic and Windows NT challenge/Response. Each of these schemes had different effect on the security context of an application launched by ITS.

**ACCESS PRIVIEGES**

IIS provides several new access levels. The following values can set the type of access allowed to specific directories:

* Read
* Write
* Script
* Execute
* Log Access
* Directory Browsing

**IIS WEBSITE ADMINISTRATION**

Administering websites can be time consuming and costly, especially for people who manage large internet Service Provider (ISP) Installations. But is there a cost-effective way to support both? The answer is yes; if you can automate administrative tasks and let users administer their own sites from remote computers. Microsoft Internet Information server (IIS) version 4.0 offers technologies to do this:

1. Windows scripting Host (WSH)

2. IIS Admin objects built on top of Active Directory service Interface(ADS))   
With these technologies working together behind the scenes, the person can administers sites from the command line of central computer and can group frequently used commands in batch files.

**System Implementation and Maintenance**

The design of a management information system may seem to management to be an expensive project, the cost of getting the MIS on line satisfactorily may often be comparable to that of its design, and the implementation has been accomplished when the outputs of the MIS are continuously utilized by decision makers.  
Once the design has been completed, there are four basic methods for implementing the MIS.   
These are as  
1. Install the system in a new operation or organization.   
2. Cut off the old system and install the new   
 3. Cut over by segments   
 4. Operate in parallel and cut over

**Plan the implementation:**  
The three main phases in implementation take place in series.   
These are   
1. The initial installation   
2. The test of the system as a whole   
3. The evaluation, maintenance and control of the system   
 Many implementation activities should be undertaken in parallel to reduce implementation time. Training of personnel and preparation of software may be in parallel with each other and with other implementation activities.

**Implementation Tasks**  
The major implementation tasks are consists of  
1. Planning the implementation activities   
2. Acquiring and laying out facilities and offices   
3. Organizing the personnel for implementation   
4. Developing procedures for installation and testing   
5. Developing the training program for operating personnel   
6. Completing the system’s software   
7. Acquiring required hardware   
8. Generating files   
9. Designing forms   
10. Testing the entire system   
11. Completing cutover to the new system   
12. Documenting the system   
13. Evaluating the MIS   
14. Providing system maintenance (debugging and improving)

**1. Planning the implementation activities**

**Establish Relationships among tasks**  
 For small projects, the order of performance may simply be described in text form. A Gantt chart or network diagram makes visualization of the plan and schedule much clearer.   
For large projects, many concurrent and sequential activities are interrelated so that a network diagram must be employed in any good plan.

**Establish a Schedule**  
 Schedule is prepared by having the system designers estimate the times between the events in the program network. The critical path (longest time through the network) can be calculated. After specifying the starting date, the end date is established.

**Cost Schedule to Tasks and Time**

The cost for completing each task required to complete is established as part of the plan; then the rate of expenditures should be budgeted.

Reporting and control of the work in progress may be obtained by weekly meetings. The financial personnel must make certain that report formats allow them to show cost and technical progress relationship as well as cost and time.

**2. Acquiring and laying out facilities and offices**  
For the installation of a new system to replace a current one may require a major revision of facilities as well as completely new office, computer room etc.

The MIS project manager must prepare rough layouts and estimates of particular floor areas that feel to be needed. The manager then prepares cost estimates.

Space planning must be done by the space to be occupied by people, the space occupied by equipment and the movement of people and equipment in the work progress.   
**3. Organizing the personnel for implementation**  
As the implementation tasks have been defined, management usually assigns a project manager to guide the implementation.   
Top management must make the middle managers for their involvement in implementation, besides these, systems specialists, computer programmer; top management should make sure that each people who will operate the system should have active parts in the implementation.   
**4. Developing procedures for installation and testing**  
After organizing the personnel for implementation the next task is to develop or prepare the procedures for implementation. As the project leader has the network plan for proceeding with the implementation, this leader calls the key people in the project to prepare more detailed procedures for system installation.   
**5. Developing the training program for operating personnel**  
A program is developed keeping in mind to impress management and support. After developing the program, it is necessary to train operating personnel in their new duties. They must have a thorough understanding of what the new MIS is like and what it is supposed to do. They must learn how it will operate. They are faced with many changes in their work and have to obtain acceptance of changes.   
**6. Completing the system’s software**  
As the software is developed internally or under contract, in both cases, the software development must take in mind the nature of the hardware required.   
As the system designers and programmers provide the flow diagrams and the block diagrams during the detailed design state. Some modification may be required, as the implementation stage progresses.

**7. Acquiring required hardware**  
This acquisition is usually the limiting factor in getting am MIS implementation. These tasks should be started during the design stage.   
The decision is to be needed, whether to buy or lease the hardware. Capital expenditure analysis is only one of many factors involved in this decision. Others are prestige, usage etc.   
**8. Generating files**  
In the implementation stage, the actual data must be obtained and recorded for the initial testing and operation of the system. This requires format of the data, storage form and format and remarks to indicate when the data have been stored.

**9. Designing forms**  
For controlling the marketing, a salesperson has to fill out the forms summarizing the day’s activities. The form ensures the right information to be supplied for computer storage.   
Forms are required not just for input and output but also for transmitting data at intermediate stages.   
**10. Testing the entire system**  
As the total system is installed; tests should be performed with the test specifications and procedure. A test during installation stage consists of component tests, subsystem tests and total system acceptance tests.   
Components may be equipment (that can be new or old), new software programs, new data collection methods, work procedures, reporting formats. Difficulties that occur during component tests may lead t design changes.   
System tests require verification of multiple inputs, complex logic systems, and timing aspects of many parts.

**11. Completing cutover to the new system**  
Cutover is a point at which the new component replaces the old component to the new system replaces the old system. This involves old forms, old files and old equipment being retried.   
The debugging proves associated with the cutover to the new system may extend for several months.   
**12. Documenting the system**  
Documentation of the MIS means preparation of written descriptions of the scope, purpose, information flow components, and operating procedures of the system.   
Documentation is a necessity for troubleshooting, for replacement of subsystems, for interfacing with other systems, for training new operating personnel and also for evaluating and upgrading the system.   
**13. Evaluating the system**  
After the MIS has been operating smoothly for a short period of time; an evaluation of each step in the design and of the final system performance should be made.   
Evaluation should not be delayed beyond the time when the system’s analysts have completed most of the debugging.

**14. Providing system maintenance**

Control and maintenance of the system are the responsibilities of the line managers.   
  
Maintenance is closely related to control. Maintenance is that ongoing activity that keeps the MIS at the highest levels of effectiveness and efficiency within cost constraints.   
  
Maintenance is directed towards reducing errors due to design, reducing errors due to environmental changes and improving the system’s scope and services.

# Cost &Benefits Analysis

# Cost Benefit Analysis Method (CBMA)

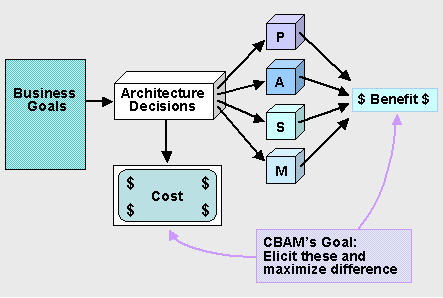
The Cost Benefit Analysis Method (CBAM) is an architecture-centric method for analyzing the costs, benefits, and schedule implications of architectural decisions. It also enables assessment of the uncertainty surrounding judgments of costs and benefits, thereby providing a basis for informed decision making about architectural design/upgrade. The CBAM builds on the [Architecture Tradeoff Analysis Method (ATAM)](http://www.sei.cmu.edu/architecture/tools/evaluate/atam.cfm), although an ATAM is not an absolute prerequisite.

## Challenges

* How do you go about taking economic considerations into account when designing or modifying system architecture?
* How do you account for the costs involved?
* How do you characterize and compare the benefits that will accrue to various architectural strategies?
* How can costs and benefits be traded off against quality attributes or functionality?
* How can you characterize the uncertainties involved in your estimates?

## Description

The creation and maintenance of a complex software-intensive system involves making a series of business-critical architecture design decisions. The SEI ATAM provides software architects with a framework for understanding the technical tradeoffs they face as they make design or maintenance decisions. But the biggest tradeoffs in large complex systems usually have to do with economics, and the ATAM does not provide any guidance for understanding these economic tradeoffs. Organizations need to know how to invest their resources to maximize their gains, meet their schedules, and minimize their risks. When economics have been addressed in the past, the focus has usually been on costs, and even then only the costs of building the system have been considered, not the long-term costs of maintenance and upgrade. Yet the benefits that an architectural decision may bring to an organization are as important-or perhaps even more important-than the costs.

Clearly we need to consider both, that is to consider the return on investment (ROI) of any architectural decision. Because the resources for building and maintaining a system are finite, there must be a rational process for choosing among architectural options, during the initial design phase and during subsequent periods of upgrade. These options will have different costs, consume different amounts of resources, implement different features (each bringing some benefit to the organization), and have some inherent risk or uncertainty. The ATAM uncovers the architectural decisions that are made (or are being considered) for the system, and links these decisions to business goals and quality attributes. The CBAM builds on this foundation, as exemplified by the cubes labeled P, A, S, and M, in this figure (representing performance, availability, security, and modifiability respectively).

The CBAM consists of the following steps:

1. choosing scenarios and architectural strategies
2. assessing quality attribute benefits
3. quantifying the benefits of architectural strategies
4. quantifying the costs and schedule implications of architectural strategies
5. calculating desirability
6. making decisions

Through the CBAM exercise, the CBAM team guides the stakeholders to determine a set of architectural strategies that address their highest priority scenarios. These chosen strategies furthermore represent the optimal set of architectural investments. They are optimal based on considerations of benefit, cost, and schedule, within the constraints of the elicited uncertainty of these judgments and the willingness of the stakeholders to withstand the risk implied by uncertainty.

## Benefits

The CBAM enables users to make informed decisions about software requirements and software investments based on an analysis of the economic and architectural implications of those decisions.

## Who Would Benefit

Software product or project managers, software product line managers, business analysts, software architects/senior designers.

### Input and Output Screen Design

This section is a description of the input media used by the operator for providing information to the system; For example, data entry screens, optical character readers, bar scanners, etc.

Provide the layout of all input data screens or graphical user interfaces (GUTs) (for example, windows). Provide a graphic representation of each interface. Define all data elements associated with each screen or GUI, or reference the data dictionary.

* Copies of form(s) if the input data are keyed or scanned for data entry from printed forms
* Description of any access restrictions or security considerations
* Each transaction name, code, and definition, if the system is a transaction-based processing system

## Outputs:

This section describes of the system output design relative to the user/operator; System outputs include reports, data display screens and GUIs, query results, etc. The following should be provided, if appropriate:

* Identification of codes and names for reports and data display screens
* Description of report and screen contents (provide a graphic representation of each layout and define all data elements associated with the layout or reference the data dictionary)
* Description of the purpose of the output, including identification of the primary users
* Report distribution requirements, if any (include frequency for periodic reports)

Description of any access restrictions or security considerations

**Types:**

* **Internal outputs** stay inside the system to support the system's users and managers
* **External outputs** leave the system to trigger actions on the part of their recipients or confirm actions to their recipients
  + **Turnaround outputs** are those which are typically implemented as a report eventually re-enters the system as an input
* **Detailed Reports:**
  + Present information with little or no filtering or restrictions
  + Some detailed reports are historical in nature
  + Detailed reports confirm and document the successful processing of transactions and serve as an audit trail for subsequent management inquiry
* **Exception Reports:**
  + Filter data before it is presented to the manager as information
  + Exception reports only report exceptions to some condition or standard

**Out Put Media:**

* Paper
* Screen
* Microfilm/Microfiche
* Video/Audio
* CDROM, DVD
* Other electronic media

**Process Involved**

**Introduction to System**

**Analyze and System**

Systems are created to solve problems. One can think of the systems approach as an organized way of dealing with a problem. In this dynamic world, the subject System Analysis and Design (SAD) mainly deals with the software development activities.

**Defining a System**

A collection of components that work together to realize some objectives forms a system. Basically there are three major components in every system, namely input, processing and output.

Input Output

Processing

**Fig. 1.1: Basic System Components**

In a system the different components are connected with each other and they are interdependent. For example, human body represents a complete natural system. The objective of the system demands that some output is produced as a result of processing the suitable inputs. A well-designed system also includes an additional element referred to as ‘control’ that provides a feedback to achieve desired objectives of the system.

**System Life Cycle**

System life cycle is an organizational process of developing and maintaining systems. It helps in establishing a system project plan, be-cause it gives overall list of processes and sub-processes required for developing a system.

**Following are the different phases of system development life cycle:**

* Preliminary study
* Feasibility study
* Detailed system study
* System analysis
* System design
* Coding
* Testing
* Implementation
* Maintenance

**The different phases of system development life cycle are shown in Fig. 1.2 below**

System Study

Maintenance Feasibility Study

Software

Development

Life Cycle

Implementation System Analysis

Testing System Design

Coding

**Fig. 1.2: Phases of System Development Life Cycle**

**Phases of System Development Life Cycle**

Let us now describe the different phases and related activities of system development life cycle.

**Preliminary System Study**

Preliminary system study is the first stage of system development life cycle. This is a brief investigation of the system under consideration and gives a clear picture of what actually the physical system is? In practice, the initial system study involves the preparation of a ‘System Proposal’ which lists the Problem Definition, Objectives of the Study, Terms of reference for Study, Constraints, and Expected benefits of the new system, etc. in the light of the user requirements.

In summary, we would say that system study phase passes through the following steps:

* problem identification and project initiation
* background analysis
* inference or findings (system proposal

**Feasibility Study**

In case the system proposal is acceptable to the management, the next phase is to examine the feasibility of the system. The feasibility study is basically the test of the proposed system in the light of its workability, meeting user’s requirements, effective use of resources and of course, the cost effectiveness. These are categorized as technical, operational, economic and schedule feasibility. The main goal of feasibility study is not to solve the problem but to achieve the scope. This also defines the resources needed to complete the de- tailed investigation.

**Detailed System Study**

The detailed investigation of the system is carried out in accordance with the objectives of the proposed system. This involves detailed study of various operations performed by a system and their relationships within and outside the system. During this process, data are collected on the available files, decision points and transactions handled by the present system. Interviews, onsite observation and questionnaire are the tools used for detailed system study. Using the following steps it becomes easy to draw the exact boundary of the new system under consideration:

* Keeping in view the problems and new requirements
* Workout the pros and cons including new areas of the system

All the data and the findings must be documented in the form of detailed data flow diagrams (DFDs), data dictionary, logical data structures and miniature specification. The main points to be discussed in this stage are:

* Specification of what the new system is to accomplish based on the user requirements
* Functional hierarchy showing the functions to be performed by the new system and their relationship with each other
* Functional network, which are similar to function hierarchy but they highlight the functions which are common to more than one procedure
* List of attributes of the entities – these are the data items which need to be held about each entity (record)

**System Analysis**

Systems analysis is a process of collecting factual data, understand the processes involved, identifying problems and recommending feasible suggestions for improving the system functioning. This involves studying the business processes, gathering operational data, understand the information flow, finding out bottlenecks and evolving solutions for overcoming the weaknesses of the system so as to achieve the organizational goals. System Analysis also includes sub- dividing of complex process involving the entire system, identification of data store and manual processes.

**System Design**

Based on the user requirements and the detailed analysis of the existing system, the new system must be designed. This is the phase of system designing. It is the most crucial phase in the developments of a system. Normally, the design proceeds in two stageges

* Preliminary or General Design
* Structured or Detailed Design
* Preliminary or General Design

In the preliminary or general design, the features of the new system are specified. The costs of implementing these features and the benefits to be derived are estimated.

**Structured or Detailed Design:**

In the detailed design stage, computer oriented work begins in earnest. At this stage, the design of the system becomes more structured. Structure design is a blue print of a computer system solution to a given problem having the same components and inter-relationships among the same components as the original problem. Input, output, databases, forms, codification schemes and processing specifications are drawn up in detail. In the design stage, the programming language and the hard-ware and software platform in which the new system will run are also decided.

There are several tools and techniques used for describing the system design of the system. These tools and techniques are:

* Flowchart
* Data flow diagram (DFD)
* Data dictionary
* Structured English
* Decision table
* Decision tree

**The system design involves:**

i.Defining precisely the required system output

ii.Determining the data requirement for producing the output

iii.Determining the medium and format of files and databases

iv.Devising processing methods and use of software to produce output

v.Determine the methods of data capture and data input

vi.Designing Input forms

vii.Designing Codification Schemes

viii.Detailed manual procedures

ix.Documenting the Design

**Coding**

The system design needs to be implemented to make it a workable system. This demands the coding of design into computer understandable language, i.e., programming language. This is also called the programming phase in which the programmer converts the pro-gram specifications into computer instructions, which we refer to as programs. It is an important stage where the defined procedures are transformed into control specifications by the help of a computer language. The programs coordinate the data movements and control the entire process in a system. It is generally felt that the programs must be modular in nature.

**Testing**

Before actually implementing the new system into operation, a test run of the system is done for removing the bugs, if any. It is an important phase of a successful system. After codifying the whole programs of the system, a test plan should be developed and run on a given set of test data. The output of the test run should match the expected results.

**Using the test data following test run are carried out:**

* Program test
* System test
* Program test

When the programs have been coded, compiled and brought to working conditions, they must be individually tested with the prepared test data. Any undesirable happening must be noted and debugged (error corrections).

**System Test**

After carrying out the program test for each of the programs of the system and errors removed, then system test is done. At this stage the test is done on actual data. The complete system is executed on the actual data. At each stage of the execution, the results or output of the system is analyzed. During the result analysis, it may be found that the outputs are not matching the expected output of the system.

**Implementation**

After having the user acceptance of the new system developed, the implementation phase begins. Implementation is the stage of a project during which theory is turned into practice. The major steps involved in this phase are:

* Acquisition and Installation of Hardware and Software
* Conversion
* User Training
* Documentation

The hardware and the relevant software required for running the system must be made fully operational before implementation. The conversion is also one of the most critical and expensive activities in the system development life cycle. The data from the old system needs to be converted to operate in the new format of the new system.

Main topics of such type of training are:

* How to execute the package
* How to enter the data
* How to process the data (processing details)
* How to take out the reports

After the users are trained about the computerized system, working has to shift from manual to computerized working. The process is called ‘Changeover’. The following strategies are followed for changeover of the system.

**Direct Changeover:**

This is the complete replacement of the old system by the new system. It is a risky approach and requires comprehensive system testing and training.

**Parallel run:**

In parallel run both the systems, i.e., computerized and manual, are executed simultaneously for certain defined period. The same data is processed by both the systems. This strategy is less risky but more expensive because of the following:

* Manual results can be compared with the results of the computerized system
* The operational work is doubled
* Failure of the computerized system at the early stage does not affect the working of the organization, because the manual system continues to work, as it used to do

**Pilot run:**

In this type of run, the new system is run with the data from one or more of the previous periods for the whole or part of the system. The results are compared with the old system results. It is less expensive and risky than parallel run approach.

There are generally two types of documentation prepared for any system. These are:

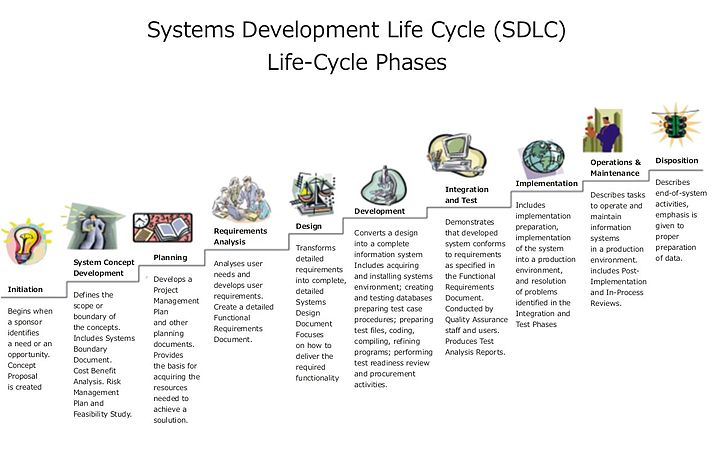
* User or Operator Documentation
* System Documentation

The user documentation is a complete description of the system from the users point of view detailing how to use or operate the system. The system documentation contains the details of system design, programs, their coding, system flow, data dictionary, process description, etc. This helps to understand the system and permit changes to be made in the existing system to satisfy new user needs.

**Maintenance**

Maintenance is necessary to eliminate errors in the system during its working life and to tune the system to any variations in its working environments. It has been seen that there are always some errors found in the systems that must be noted and corrected. It also means the review of the system from time to time. The review of the system is done for:

* Knowing the full capabilities of the system
* Knowing the required changes or the additional requirements
* Studying the performance.

[](http://commons.wikimedia.org/wiki/File:Systems_Development_Life_Cycle.jpg)

**System analysis**

The goal of system analysis is to determine where the problem is in an attempt to fix the system. This step involves breaking down the system in different pieces to analyze the situation, analyzing project goals, breaking down what needs to be created and attempting to engage users so that definite requirements can be defined.

Requirements analysis sometimes requires individuals/teams from client as well as service provider sides to get detailed and accurate requirements; often there has to be a lot of communication to and from to understand these requirements.

**Design**

In systems design the design functions and operations are described in detail, including screen layouts, business rules, process diagrams and other documentation. The output of this stage will describe the new system as a collection of modules or subsystems.

Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input design.

**Implementation**

Modular and subsystem programming code will be accomplished during this stage. Unit testing and module testing are done in this stage by the developers. This stage is intermingled with the next in that individual modules will need testing before integration to the main project.

**Testing**

The code is tested at various levels in software testing. Unit, system and user acceptance testing’s often performed. This is a grey area as many different opinions exist as to what the stages of testing are and how much if any iteration occurs. Iteration is not generally part of the waterfall model, but usually some occur at this stage. In the testing the whole system is test one by one,

Following are the types of testing:

* Defect testing
* Path testing
* Data set testing
* Unit testing
* System testing
* Integration testing
* Black box testing
* White box testing
* Regression testing
* Automation testing
* User acceptance testing
* Performance testing

**Operations and maintenance**

The deployment of the system includes changes and enhancements before the decommissioning or sunset of the system. Maintaining the system is an important aspect of SDLC. As key personnel change positions in the organization, new changes will be implemented, which will require system updates.

**Methodology Used for Testing**

Testing is a process, which reveals errors in the program. It is the major quality measure employed during software development. During testing, the program is executed with a set of conditions known as test cases and the output is evaluated to determine whether the program is performing as expected.

**Levels of Testing:**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  | | --- | --- | --- | |  |  |  |  |  |  |  | | --- | --- | --- | |  |  |  |  |  |  |  | | --- | --- | --- | |  |  |  |   Units Units Units  i/P integration o/p i/p integration o/P |   Module1 module2 module3 |   System Testing: Presentation +business + database  UAT: user acceptance testing |

**Types of Testing**

**Unit Testing**

Unit Testing is done on individual modules as they are completed and become executable. It is confined only to the designer's requirements.

**Each module can be tested using the following two strategies:**

**Black Box Testing**

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing has been uses to find errors in the following categories:

1. Incorrect or missing functions
2. Interface errors
3. Errors in data structure or external database access
4. Performance errors
5. Initialization and termination errors.

In this testing only the output is checked for correctness. The logical flow of the data is not checked.

**White Box testing**

In this the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases.

It has been uses to generate the test cases in the following cases:

1. Guarantee that all independent paths have been executed
2. Execute all logical decisions on their true and false sides
3. Execute all loops at their boundaries and within their operational bounds
4. Execute internal data structures to ensure their validity

**Integrating Testing**

Integration testing ensures that software and subsystems work together as a whole. It tests the interface ofall the modules to make sure that the modules behave properly when integrated together.

**System Testing**

Involves in-house testing of the entire system before delivery to the user. Its aim is to satisfy the user the system meets all requirements of the client's specifications.

**Acceptance Testing**

It is a pre-delivery testing in which entire system is tested at client's site on real world data to find errors.

**Validation**

The system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled. In case of erroneous input corresponding error messages are displayed.

**Compilation Test:**

It was a good idea to do our stress testing early on, because it gave us time to fix some of the unexpected deadlocks and stability problems that only occurred when components were exposed to very high transaction volumes.

**Execution Test:**

This program was successfully loaded and executed. Because of good programming there was no execution error.

##### **Output Test:**

The successful output screens are placed in the output screens section above.

## Test cases and results

**Test Case: 1**

**Description:**Admin has to login into login page.

|  |  |
| --- | --- |
| Input: | Correct username and password. |
| Desired output: | Admin login into his window login page. |
| On Error: | A message indicating invalid user. |
| If fields kept blank | A message indicating to fill the desired fields. |
| Remark | Ok. |

**Test case :2**

**Description:**Admin creates user and user has to login with valid username and password.

|  |  |
| --- | --- |
| Input: | Correct username and password. |
| Desired output: | User login into his window login page. |
| On Error: | A message indicating invalid user. |
| If fields kept blank | A message indicating to fill the desired fields. |
| Remark | Ok. |

**Project Code:**

**Admin Login:**

using System;

using System.Data;

using System.Configuration;

using System.Collections;

using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

using System.Web.UI.HtmlControls;

publicpartialclassAdmin\_frmAdminLogin : System.Web.UI.Page

{

clsAdminLogin\_Logic objAmin = newclsAdminLogin\_Logic();

protectedvoid Page\_Load(object sender, EventArgs e)

{

}

protectedvoid btnSubmit\_Click(object sender, EventArgs e)

{

try

{

objAmin.AdminLoginId = txtLoginId.Text.Trim();

objAmin.Password = txtPassword.Text.Trim();

DataSet dsAdminLoginDetail = objAmin.GetAdminLoginDetails();

DataRowCollection drc = dsAdminLoginDetail.Tables[0].Rows;

if (drc.Count > 0)

{

lblError.Visible = false;

DataRow dr = drc[0];

Session["AdminId"] = dr["AdminLoginId"].ToString();

Response.Redirect("frmAdminHome.aspx");

}

else

{

lblError.Visible = true;

lblError.Text = "Invalid Login ID/Password";

}

}

catch (Exception ex)

{

lblError.Visible = true;

lblError.Text = ex.Message.ToString();

}

}

protectedvoid Button1\_Click(object sender, EventArgs e)

{

Response.Redirect("~/Default.aspx");

}}

**Add Staff:**

using System;

using System.Data;

using System.Configuration;

using System.Collections;

using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

using System.Web.UI.HtmlControls;

publicpartialclassAdmin\_frmAddEmployee : System.Web.UI.Page

{

clsEmployee\_Logic objEmployee = newclsEmployee\_Logic();

clsCommon\_Logic objCommon = newclsCommon\_Logic();

protectedvoid Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

BindDept();

BindCountry();

//BindStatus();

BindState();

BindCity();

BindDesignation();

}

}

void BindCountry()

{

//DataSet dsCountry = objCommon.GetCountryName();

//ddlCountryName.DataSource = dsCountry.Tables[0];

//ddlCountryName.DataTextField = "CountryName";

//ddlCountryName.DataValueField = "CountryId";

//ddlCountryName.DataBind();

//ddlCountryName.Items.Insert(0, "Select");

}

void BindState()

{

//DataSet dsState = objCommon.GetStateName();

//ddlStateName.DataSource = dsState.Tables[0];

//ddlStateName.DataTextField = "StateName";

//ddlStateName.DataValueField = "StateId";

//ddlStateName.DataBind();

//ddlStateName.Items.Insert(0, "Select");

}

void BindCity()

{

//DataSet dsCity = objCommon.GetCityName();

//ddlCityName.DataSource = dsCity.Tables[0];

//ddlCityName.DataTextField = "CityName";

//ddlCityName.DataValueField = "CityId";

//ddlCityName.DataBind();

//ddlCityName.Items.Insert(0, "Select");

}

//void BindStatus()

//{

// DataSet dsStatus = objCommon.GetStatusName();

// ddlStatus.DataSource = dsStatus.Tables[0];

// ddlStatus.DataTextField = "StatusName";

// ddlStatus.DataValueField = "StatusId";

// ddlStatus.DataBind();

// ddlStatus.Items.Insert(0, "Select");

//}

void BindDept()

{

DataSet dsDept = objCommon.GetDeptName();

ddlDept.DataSource = dsDept.Tables[0];

ddlDept.DataTextField = "DeptName";

ddlDept.DataValueField = "DeptId";

ddlDept.DataBind();

ddlDept.Items.Insert(0, "Select");

}

void BindDesignation()

{

DataSet dsDesgn = objCommon.GetDesignation();

ddlDesig.DataSource = dsDesgn.Tables[0];

ddlDesig.DataTextField = "DesigType";

ddlDesig.DataValueField = "DesigId";

ddlDesig.DataBind();

ddlDesig.Items.Insert(0, "Select");

}

protectedvoid btnSubmit\_Click(object sender, EventArgs e)

{

objEmployee.EmpName = txtEmpName.Text.Trim();

objEmployee.Address = txtAddress.Text.Trim();

//objEmployee.CountryId = Convert.ToInt32(ddlCountryName.SelectedItem.Value);

//objEmployee.StateId = Convert.ToInt32(ddlStateName.SelectedItem.Value);

// objEmployee.CityId = Convert.ToInt32(ddlCityName.SelectedItem.Value);

objEmployee.ContactNo = txtContactNo.Text.Trim();

objEmployee.DeptId = Convert.ToInt32(ddlDept.SelectedItem.Value);

objEmployee.EmailId = txtEmailId.Text.Trim();

objEmployee.UserName = txtUserName.Text;

objEmployee.Password = txtPassword.Text.Trim();

objEmployee.Designation = ddlDesig.SelectedItem.Text;

objEmployee.Role = "Employee";

int i=objEmployee.AddEmployeeDetails();

if (i == 1)

{

Response.Redirect("frmManageEmployee.aspx");

}

if (i == -2)

{

lblError.Visible = true;

lblError.Text = "Sorry This user name is already existing please choose another username";

}

}

protectedvoid btnBack\_Click(object sender, EventArgs e)

{

Response.Redirect("frmManageEmployee.aspx");

}

protectedvoid ddlCountryName\_SelectedIndexChanged(object sender, EventArgs e)

{

//if (ddlCountryName.SelectedItem.Text != "Select")

//{

// objCommon.pro\_CountryId = Convert.ToInt32(ddlCountryName.SelectedItem.Value);

// BindState();

//}

}

//protected void ddlStateName\_SelectedIndexChanged(object sender, EventArgs e)

//{

// if (ddlStateName.SelectedItem.Text != "Select")

// {

// objCommon.pro\_StateId = Convert.ToInt32(ddlStateName.SelectedItem.Value);

// BindCity();

// }

//}

//protected void ddlCityName\_SelectedIndexChanged(object sender, EventArgs e)

//{

//}

}

**Leave Application:**

using System;

using System.Data;

using System.Configuration;

using System.Collections;

using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

using System.Web.UI.HtmlControls;

publicpartialclassEmployee\_frmLeaveApplication : System.Web.UI.Page

{

clsLeave\_Logic objLeave = newclsLeave\_Logic();

clsApplication\_Logic objApplicaion = newclsApplication\_Logic();

protectedvoid Page\_Load(object sender, EventArgs e)

{

if (Session["UserName"] == null)

{

Response.Redirect("frmEmployeeLogin.aspx");

}

else

txtUserName.Text = Session["UserName"].ToString();

if (!IsPostBack)

{

BindLeavetype();

}

GMDStartDate.Attributes.Add("readonly", "readOnly()");

}

void BindLeavetype()

{

//DataSet dsLeavetype = objLeave.GetLeaveType();

//ddlLeaveType.DataSource = dsLeavetype.Tables[0];

//ddlLeaveType.DataTextField = "LeaveTypeName";

//ddlLeaveType.DataValueField = "LeaveTypeId";

//ddlLeaveType.DataBind();

//ddlLeaveType.Items.Insert(0, "Select");

}

protectedvoid ddlLeaveType\_SelectedIndexChanged(object sender, EventArgs e)

{

if (ddlLeaveType.SelectedItem.Text != "Select")

{

objLeave.UserName = Session["UserName"].ToString();

objLeave.LeaveTypeId = Convert.ToInt32(ddlLeaveType.SelectedItem.Value);

int BalaceLeaves = objLeave.GetBalanceDays();

txtBalanceLeave.Text = BalaceLeaves.ToString();

}

else

txtBalanceLeave.Text = "";

}

protectedvoid btnApplyLeave\_Click(object sender, EventArgs e)

{

if (Convert.ToInt32(txtNoOfdays.Text) <= Convert.ToInt32(txtBalanceLeave.Text))

{

objApplicaion.UserName = Session["UserName"].ToString();

objApplicaion.LeaveTypeId = Convert.ToInt32(ddlLeaveType.SelectedItem.Value);

objApplicaion.StartingDate = GMDStartDate.DateString;

objApplicaion.ApplaiyingDate = System.DateTime.Now.ToString();

objApplicaion.NoOfDays = Convert.ToInt32(txtNoOfdays.Text);

//string.Format("{0:yyyy-MM-dd}", System.DateTime.Now.AddDays(1))

DateTime s = GMDStartDate.Date;

objApplicaion.EndingDate = (s.AddDays(Convert.ToInt32(txtNoOfdays.Text))).ToString();

objApplicaion.LeavePurpose = txtPurpose.Text;

objApplicaion.AddLeaveApplication();

Response.Redirect("frmLeaveApplicationSuccess.aspx");

}

else

{

lblError.Visible = true;

lblError.Text = "No of Days should not be more than Balalnce days";

}

}

protectedvoid btnBack\_Click(object sender, EventArgs e)

{

Response.Redirect("frmEmployeeHome.aspx");

}}

### Leave Sanction:

using System;

using System.Data;

using System.Configuration;

using System.Collections;

using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

using System.Web.UI.HtmlControls;

publicpartialclassPrincipal\_frmLeaveSanction : System.Web.UI.Page

{

int LeaveTypeId, BalanceLeave;

clsApplication\_Logic objSanction = newclsApplication\_Logic();

clsCommon\_Logic objCommon = newclsCommon\_Logic();

clsLeave\_Logic objLeave = newclsLeave\_Logic();

protectedvoid Page\_Load(object sender, EventArgs e)

{

if (Request["Id"] == null)

Response.Redirect("frmLeaveChecking.aspx");

else

ViewState["AppNo"] = Request["Id"].ToString();

if (!IsPostBack)

{

BindStatus();

Binddata();

}

GMAppDate.Attributes.Add("readOnly", "readOnly()");

GMStartDate.Attributes.Add("readOnly", "readOnly()");

}

void Binddata()

{

objSanction.ApplicationNo = Convert.ToInt32(ViewState["AppNo"].ToString());

DataSet ds = objSanction.GetApplicationLeaveDetails();

DataRowCollection drc = ds.Tables[0].Rows;

if (drc.Count > 0)

{

txtAppNo.Text = drc[0]["ApplicationNo"].ToString();

GMAppDate.DateString = drc[0]["ApplyingDate"].ToString();

txtUserName.Text = (drc[0]["UserName"].ToString());

txtDays.Text = (drc[0]["NoOfDays"].ToString());

txtLeaveType.Text = drc[0]["LeaveTypeName"].ToString();

//ddlLeaveType.SelectedItem.Text = drc[0]["LeaveTypeName"].ToString();

txtPurpose.Text = drc[0]["LeavePurpose"].ToString();

GMStartDate.DateString = drc[0]["StartingDate"].ToString();

//ddlStatus.SelectedItem.Text = drc[0]["StatusName"].ToString();

int Index = ddlStatus.Items.IndexOf(ddlStatus.Items.FindByText(drc[0]["StatusName"].ToString()));

if (Index >= 0)

ddlStatus.Items[Index].Selected = true;

ViewState["LeaveTypeId"] = drc[0]["LeaveTypeId"].ToString();

//UsedLeaves = Convert.ToInt32(drc[0]["UsedLeaves"].ToString());

//BalanceLeave = Convert.ToInt32(drc[0]["BalanceLeave"].ToString());

}

}

void BindStatus()

{

DataSet dsStatus = objCommon.GetStatusName();

ddlStatus.DataSource = dsStatus.Tables[0];

ddlStatus.DataTextField = "StatusName";

ddlStatus.DataValueField = "StatusId";

ddlStatus.DataBind();

ddlStatus.Items.Insert(0, "Select");

}

protectedvoid btnSubmit\_Click(object sender, EventArgs e)

{

objLeave.ApplicationNo = Convert.ToInt32(ViewState["AppNo"].ToString());

if (objLeave.CheckAppNoInLeaveDetails())

{

if (ddlStatus.SelectedItem.Text != "Select")

{

objSanction.ApplicationNo = Convert.ToInt32(ViewState["AppNo"].ToString());

objSanction.AppStatusId = Convert.ToInt32(ddlStatus.SelectedItem.Value);

objSanction.UpdateApplicationData();

if (ddlStatus.SelectedItem.Text == "Accepted")

{

objLeave.ApplicationNo = Convert.ToInt32(ViewState["AppNo"].ToString());

objLeave.AppStatusId = Convert.ToInt32(ddlStatus.SelectedItem.Value);

objLeave.UserName = txtUserName.Text;

objLeave.LeaveTypeId = Convert.ToInt32(ViewState["LeaveTypeId"]);

BalanceLeave = objLeave.GetBalanceDays() - Convert.ToInt32(txtDays.Text);

objLeave.BalanceLeaves = BalanceLeave;

objLeave.AddLeaveDetails();

Response.Redirect("frmLeaveChecking.aspx");

}

}

}

else

{

lblError.Visible = true;

lblError.Text = "Sorry you already given status as Accepted";

}}}

**Balance Leave:**

using System;

using System.Data;

using System.Configuration;

using System.Collections;

using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

using System.Web.UI.HtmlControls;

publicpartialclassAdmin\_frmFindBalanceLeaves : System.Web.UI.Page

{

clsLeave\_Logic objLeave = newclsLeave\_Logic();

clsEmployee\_Logic objEmp = newclsEmployee\_Logic();

privatestring strError = "No Data Available";

protectedvoid Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

BindData();

}

}

void BindData()

{

try

{

objEmp.Sort\_On = "";

if (ViewState["Sort\_On"] != null)

objEmp.Sort\_On = ViewState["Sort\_On"].ToString() + " " + ViewState["Sort\_By"].ToString();

lblError.Visible = false;

DataSet dsTemp = objEmp.GetEmployee();

DataTable dtTemp = dsTemp.Tables[0];

if (dtTemp.Rows.Count > 0)

{

lblError.Visible = false;

//btnDelete.Visible = true;

}

else

{

lblError.Visible = true;

lblError.Text = strError;

//btnDelete.Visible = false;

}

if (this.txtPageSize.Text != "")

{

if (System.Convert.ToInt32(this.txtPageSize.Text) > 0)

{

this.gvLeave.PageSize = System.Convert.ToInt32(this.txtPageSize.Text);

}

}

gvLeave.DataSource = dtTemp;

gvLeave.DataBind();

if (dtTemp.Rows.Count == 0)

{

this.Lbl\_Pageinfo.Visible = false;

}

else

{

Int16 intTo;

Int16 intFrom;

if (gvLeave.PageSize \* (gvLeave.PageIndex + 1) < dtTemp.Rows.Count)

{

intTo = System.Convert.ToInt16(gvLeave.PageSize \* (gvLeave.PageIndex + 1));

}

else

{

intTo = System.Convert.ToInt16(dtTemp.Rows.Count);

}

intFrom = System.Convert.ToInt16((gvLeave.PageSize \* gvLeave.PageIndex) + 1);

this.Lbl\_Pageinfo.Text = "Record(s) " + intFrom + " to " + intTo + " of " + dtTemp.Rows.Count;

this.Lbl\_Pageinfo.Visible = true;

}

}

catch (Exception ex)

{

lblError.Text = ex.Message;

lblError.Visible = true;

}

}

protectedvoid btnSearch\_Click(object sender, EventArgs e)

{

strError = "No data matching with your searching criteria";

gvLeave.PageIndex = 0;

objEmp.UserName = txtUserName.Text.Trim();

BindData();

}

protectedvoid gvLeave\_PageIndexChanging(object sender, GridViewPageEventArgs e)

{

if (ViewState["Sort\_On"] != null)

objEmp.Sort\_On = ViewState["Sort\_On"].ToString();

else

objEmp.Sort\_On = "";

gvLeave.PageIndex = e.NewPageIndex;

BindData();

}

protectedvoid gvLeave\_RowCommand(object sender, GridViewCommandEventArgs e)

{

if (e.CommandName.ToUpper() == "UPDATE")

{

Response.Redirect("frmEmpBalanceLeaveDetails.aspx?Id=" + e.CommandArgument.ToString());

}

}

protectedvoid gvLeave\_Sorting(object sender, GridViewSortEventArgs e)

{

objEmp.Sort\_On = e.SortExpression;

ViewState["Sort\_On"] = objEmp.Sort\_On;

if (ViewState["Sort\_By"] == null)

ViewState["Sort\_By"] = "Asc";

if (ViewState["Sort\_By"].ToString() == "Asc")

{

ViewState["Sort\_By"] = "Desc";

}

else

{

ViewState["Sort\_By"] = "Asc";

}

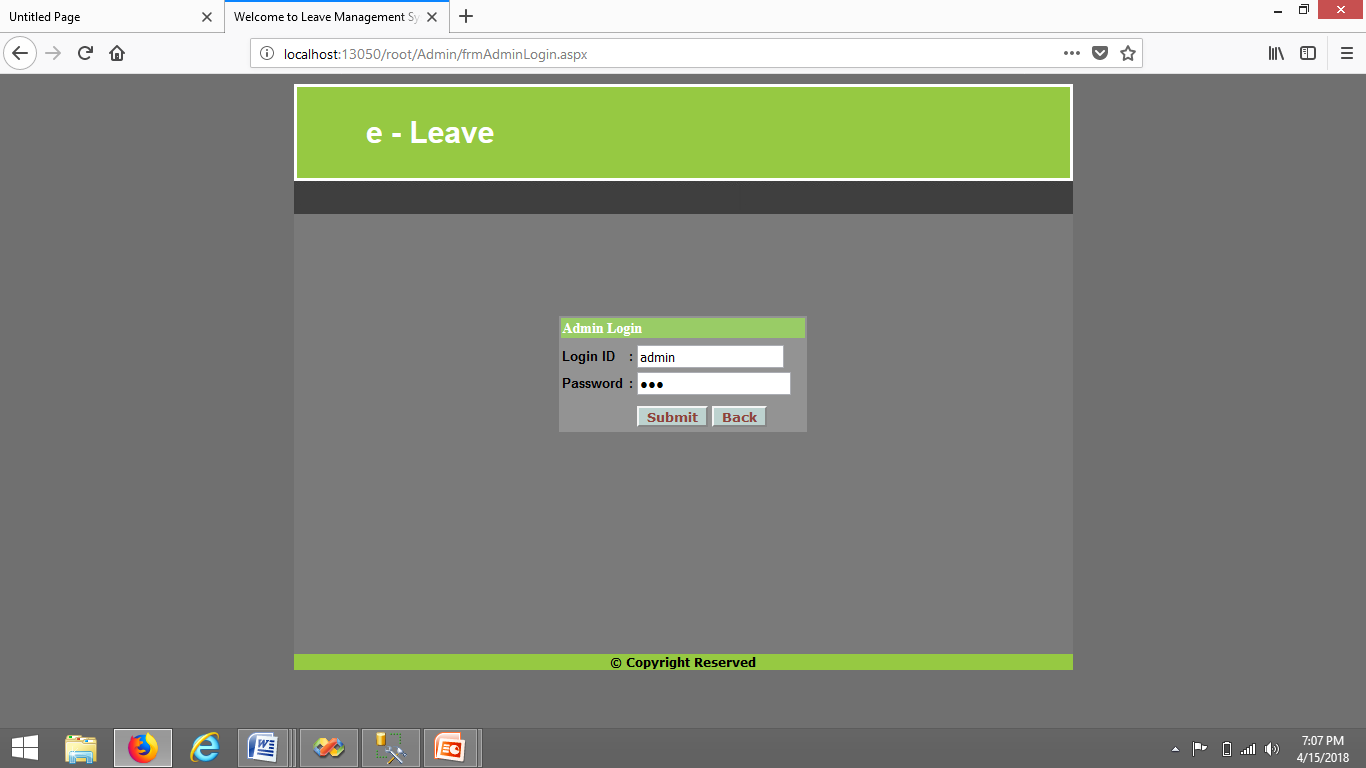
BindData();

}

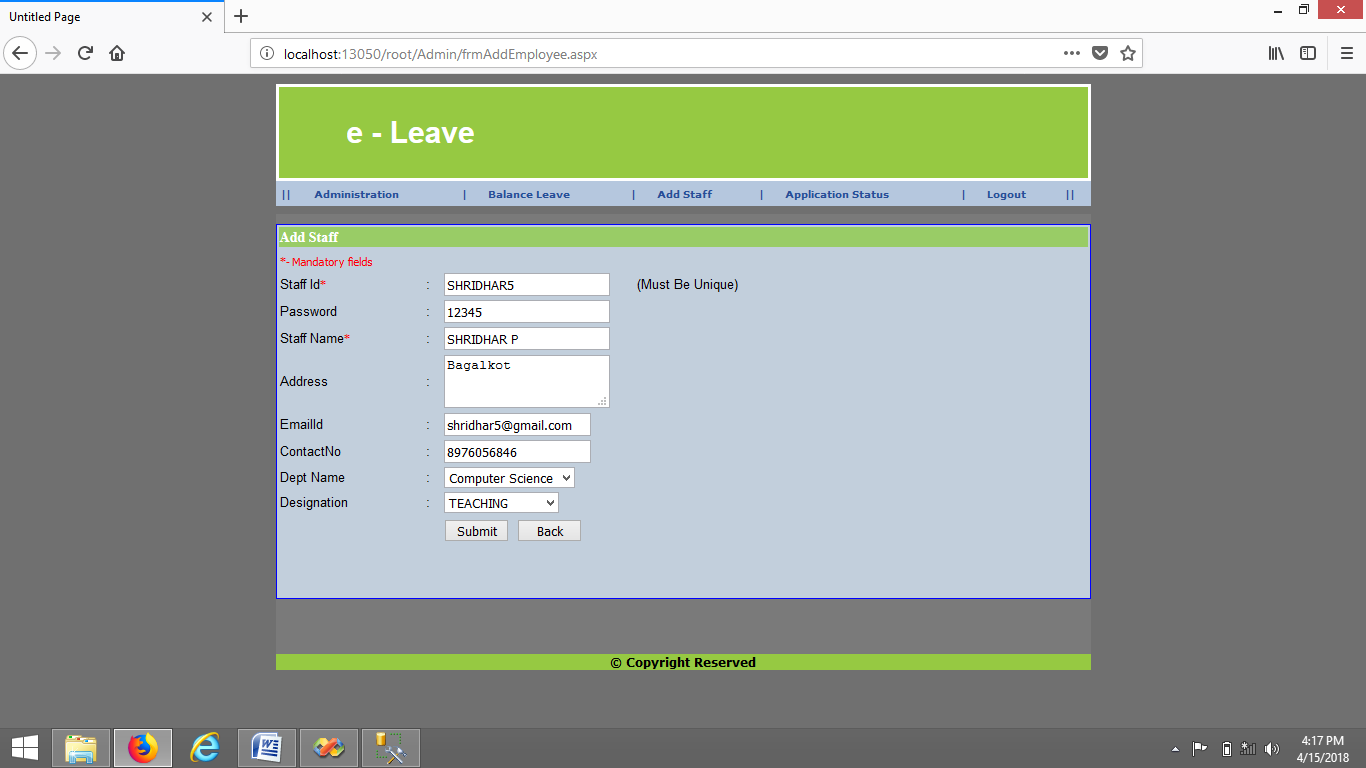
}

### User Manuals

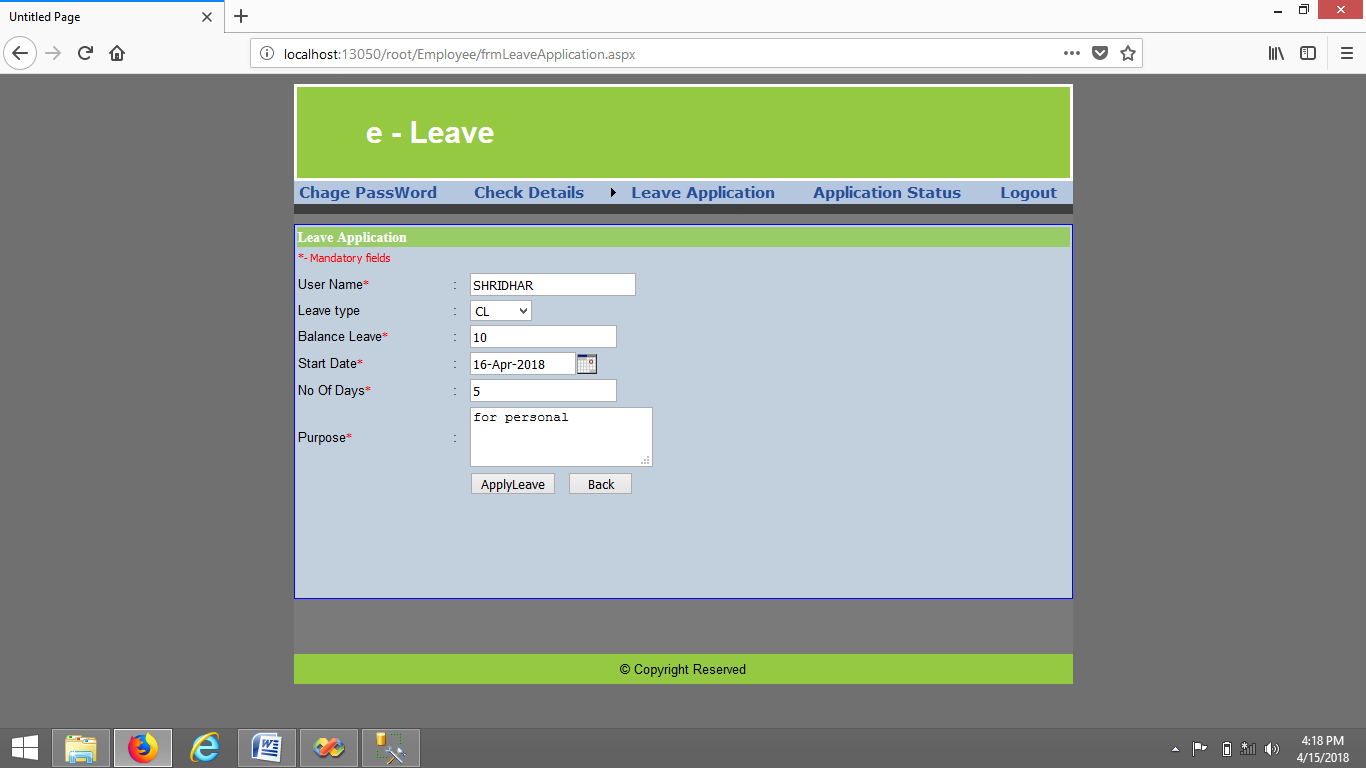
**Admin Login:**

****

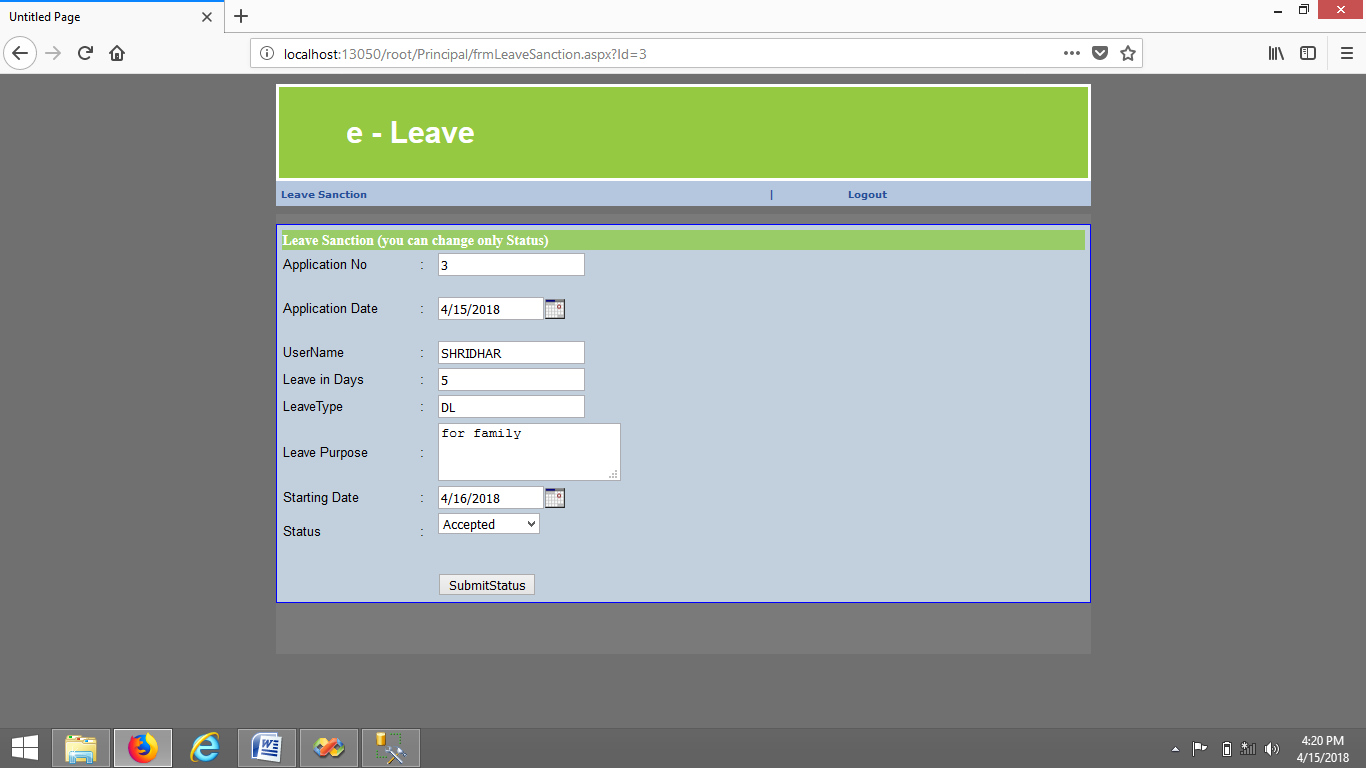
**Add Staff:**



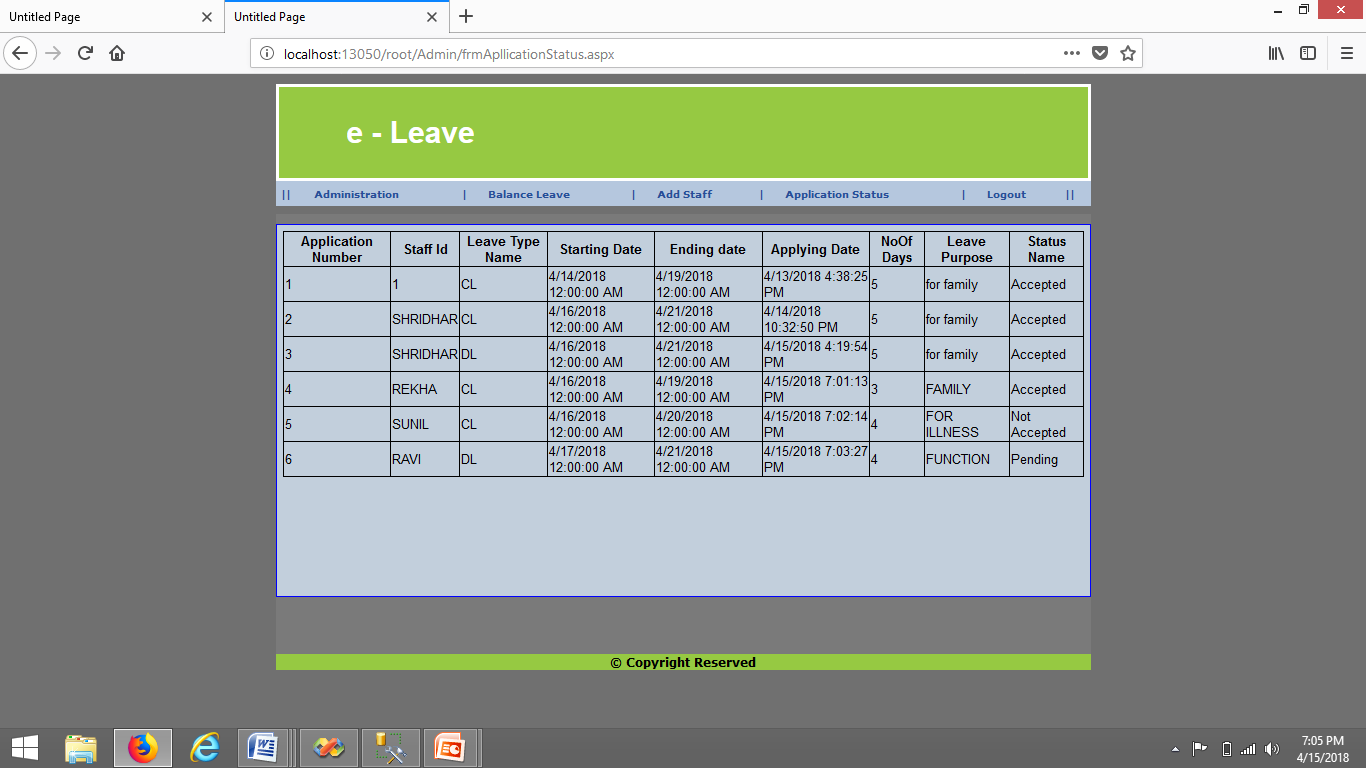
**Leave Application:**

****

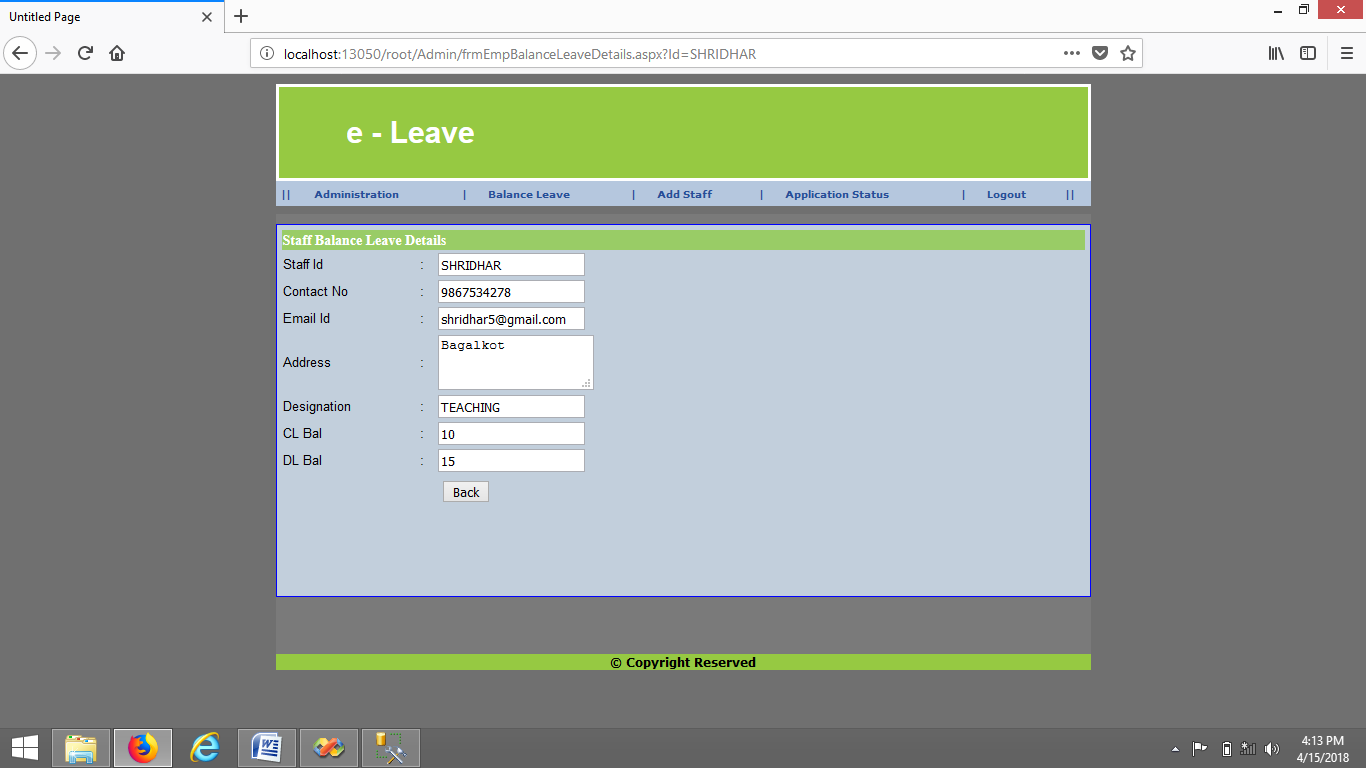
**Sanction Leave:**

****

**Application Status:**

****

**Balance Leave:**

****

**Conclusion**

e-leave is very useful for college to maintain the leave records and leave application of the staff. The higher authority may accept or reject the leave applications requested by the staff. Thus this system maintains the excess amount of job done by college to maintain the leaves.

**Future Enhancement**

* In future application should be updated with earn leave, restricted holiday, commuted leave, medical leave
* And also add SMS facility whenever we apply leave to know leave sanction and remaining leave

**References**

**Books:**

* C# and the .NET Platform Second Edition 2005 by Andrew Troelsen
* Pro ASP.NET 4 in C# 2010 by Matthew MacDonald 4th Edition

**Sites:**

* [www.Knowdotnet.com](http://www.Knowdotnet.com)
* [www.triconsole.com](http://www.triconsole.com)
* [www.stackoverflow.com](http://www.stackoverflow.com)
* [www.dotnetspider.com](http://www.dotnetspider.com)