PROJECT ALLOCATION AND TRACKING SYSTEM

(PATS)

A PROJECT REPORT

Submitted by

SHALINI.M **71205104044**

SUDHARSHINI.I **71205104502**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY: CHENNAI 600025

APRIL 2009

ANNA UNIVERSITY: CHENNAI 600025

BONAFIDE CERTIFICATE

Certified that this project report **"PROJECTS ALLOCATION AND TRACKING SYSTEM"** is the bonafide work of **"SHALINI.M, SUDHARSHINI.I"** who carried out the project work under my supervision.

SIGNATURE

Dr.S THANGASAMY

DEAN,

Department of Computer Science

and Engineering,

Kumaraguru College of Technology

Coimbatore-641006

SIGNATURE

Mr. T SUDHAKAR

SUPERVISOR

Lecturer,

Department of Computer Science

and Engineering,

Kumaraguru college of Technology,

Coimbatore-641006

The candidates with University Register Nos. **71205104044**, **71205104502** were examined by us in the project viva-voce examination held on. 27-4-2009 .

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We are extremely grateful to **Prof. R.Annamalai, M.E.,** Vice Principal, Kuamaraguru College Of Technology for having given us this opportunity to embark on this project.

We are deeply obliged to **Dr. S. Thangasamy, Ph.d,** Dean of Computer Science and Engineering for his valuable guidance and useful suggestions during the course of this project.

We wish to express our heartiest thanks to **Mrs. P. Devaki, M.E.,** Assistant Professor and Project coordinator, CSE who have helped us to overcome the perplexity while choosing the project.

We thank our guide **Mr. T Sudhakar, M.E.** , Lecturer, Department of Computer Science and Engineering, for his excellent guidance in each and every step of our project and been with us to complete the project.

We thank the **teaching and non-teaching staff** of our Department for providing us the technical support in the duration of our project.

We also thank all our **parents and friends** who helped us to complete this project successfully.

TABLE OF CONTENTS

TITLE	PAGE NO.
ABSTRACT	**vi**
LIST OF FIGURES	**vii**
LIST OF TABLES	**vii**
LIST OF ABBREVIATIONS	**vii**
1. INTRODUCTION	**1**
1.1 Overview	1
1.2 About the system	1
2. LITERATURE REVIEW	**3**
2.1 Existing system	3
2.2 Proposed system	3
3. SYSTEM CONFIGURATION & REQUIRMENTS	**4**
3.1 SYSTEM SPECIFICATION	4
3.1.1 Software Specification	4
3.1.2 Hardware Specification	4
3.2 SYSTEM REQUIREMENTS	4
3.2.1 About ASP.NET	4
3.2.2 About SQL Server	9
4. SYSTEM ANALYSIS AND DESIGN	**17**
4.1 Modules	17
4.2 Database Design	20

5. SYSTEM DEVELOPMENT **24**

5.1 System Flow Diagram 24

5.2 Data Flow Diagram 25

6. SYSTEM TESTING **26**

6.1 Unit Testing 26

6.2 Module Testing 26

6.3 Sub-System Testing 26

6.4 System Testing 26

6.5 Accepting Testing 26

7. CONCLUSION & FUTURE ENHANCEMENT **27**

7.1 Conclusion 27

7.2 Future Enhancements 28

APPENDICES **29**

A.1 Sample Coding 29

A.2 Sample Output 29

REFERENCES **48**

ABSTRACT

The Project Allocation and Tracking System (PATS) is a system which basically acts like a management center, the main objective of PATS is to allocate Clients project thereby maintaining the database. It provides online facilities to the organization to store the details of the project, clients, employees, project status and the reports.

In current systems, the implementation is done in separate sections in a manual way. Due to this, more manpower is required to maintain those sections, and the current system seems to be very ambiguous and inefficient. This results in slow data retrieval and more time is consumed.

In order to overcome these problems we introduce PATS, which is completely computerized and maintains *n* number of database in a unique section composed with different modules. Hence PATS requires less manpower and it is highly accurate and efficient.

This system helps the organization to answer the queries of the clients very quickly and accurately. Therefore the System is developed for high security and reliability purpose.

LIST OF FIGURES:-

Si. No.	Figure no.	Description	Page no.
1	5.1	System Flow Diagram	24
2	5.2	Data Flow Diagram	25

LIST OF TABLES:-

Si. No.	Table no.	Table name	Page no.
1	4.2.1	Allocation	20
2	4.2.2	Client	20
3	4.2.3	Employee	21
4	4.2.4	Project Details	22
5	4.2.5	Project Environment	22
6	4.2.6	User Creation	23
7	4.2.7	Project Order Placing	23

LIST OF ABBREVIATIONS:-

1. PATS : Project Allocation and Tracking System.
2. dbConn : Database connection.

CHAPTER 1

1. INTRODUCTION

1.1 Overview

Generally in business, system analysis and design refers to the process of examining business situation with the aim of improving better procedures and methods.

Two major components are system analysis and system design. System analysis is the process of gathering and interpreting facts, diagnosing problems and using the facts to improve the system. System design is the process of planning a new business to replace or complement the old. Here the project is the new one and not based on the existing system.

1.2 About the system

The Project Allocation and Tracking System (PATS) - is a system which basically acts like a management center, it will provide facility to the clients of our company. Some of the activities involved in the system of storing the details about projects, clients, employee details and user information.

In this system will help the organization to answering queries from the client about their projects very quickly and accurately. This System was developed well secured and reliable.

The main aim of this project is to allocate the incoming projects to the respective staffs of the organization thereby checking the availability of the staffs. Each employee of the organization is noted with their specialization in the employee information module. By giving the environment ID in this module the concerned employee of that particular environment will be listed based upon their efficiency. The efficiency of the staff is identified completed project etc.

If the man power needed for the particular project is not sufficient then the efficient staff that has ranked first among all the areas will be assigned the project. Initially, when allocating the staff, the current status of the project has been saved as 0%.Depending upon the completion of the project each day the status should be updated by each employee and it is saved. Whenever it reaches 100%, the project is said to be completed. So, the employee who has been in the non availability list will be marked as available.It provides the overall reports about all the above mentioned modules such as projects, employees, clients, ongoing projects etc.

The modules in this project also describe what project has been allocated to the employee and how far the project has been completed. The employees in the organization may not be aware of their present status. By using this each employee can look at their status i.e. whether any new project has allotted. The new project allotted will be specified with the status 0%.the employees can also find their team members by giving the particular project id specified.

CHAPTER 2

2. LITERATURE REVIEW

2.1 Existing System

The existing system is the manual one. In the manual system there are a lot of difficulties faced by the Organization such as . To fulfill the drawbacks of existing manual system, the proposed system is developed.

Some of the drawbacks of the Existing System are:

- Not user friendly and Flexible
- Slow processing
- Duplicate entries may occur in the record keeping process.
- The Client may spend more time for getting their information.
- Not automatic.

2.2 Proposed System

PATS eliminate the difficulties that are faced in the existing system. It is aimed at simplifying the complex allocation of project, storage and retrieval of data in a user friendly and graphical user interface. It is proposed to develop the system using ASP.NET as front end tool and SQL Server as back end. The proposed system is attempted to the following aspects of PATS:

- More user friendly and Flexible
- Automatic allocation of staffs
- Fast retrieval of data
- Record keeping process becomes easier.
- The time spent by the client in getting their information is saved.

CHAPTER 3

3. SYSTEM CONFIGURATION & REQUIREMENTS

3.1 System Specification

3.1.1 Software Specification

- ASP.NET Environment 2005
- Microsoft SQL server 2005
- Operating system: Windows Xp Professional

3.1.2 Hardware Specification

- 128 MB RAM
- 40 GB of Hard Disk Space
- Pentium IV Processor
- 110 keys Logitech keyboard

3.2 System Requirements

3.2.1 about ASP.NET

ASP.NET is the latest version of Microsoft's Active Server Pages technology (ASP).ASP.NET is a part of the Microsoft .NET framework, and a powerful tool for creating dynamic and interactive web pages.

What is ASP?

ASP is a server side scripting technology that enables scripts (embedded in web pages) to be executed by an Internet server.

- ASP is a Microsoft Technology
- ASP stands for Active Server Pages
- ASP is a program that runs inside IIS
- IIS stands for Internet Information Services
- IIS comes as a free component with Windows 2000
- IIS is also a part of the Windows NT 4.0 Option Pack
- The Option Pack can be downloaded from Microsoft
- PWS can be found on your Windows 95/98 CD

What is an ASP File?

- PWS is a smaller - but fully functional - version of IIS
- An ASP file is just the same as an HTML file
- An ASP file can contain text, HTML, XML, and scripts
- Scripts in an ASP file are executed on the server
- An ASP file has the file extension ".asp"

How Does it Work?

- When a browser requests an HTML file, the server returns the file
- When a browser requests an ASP file, IIS passes the request to the ASP engine on the server
- The ASP engine reads the file, line by line, and executes the scripts in the file
- Finally, the ASP file is returned to the browser as plain HTML

What is ASP.NET?

ASP 3.0 is the latest version of ASP, but there will never be an ASP 4.0 version. ASP.NET is the next generation ASP, but it's not an upgraded version of ASP. ASP.NET

is an entirely new paradigm for server-side ASP scripting.ASP.NET is a part of the .NET Framework. Microsoft spent three years rewriting ASP.NET from the ground up, and ASP.NET is not fully backward compatible with ASP 3.0

.NET Framework

The .NET Framework is the infrastructure for the Microsoft .NET platform. The .NET Framework is an environment for building, deploying, and running Web applications and Web Services. The .NET Framework contains a common language runtime and common class libraries - like ADO.NET, ASP.NET and Windows Forms - to provide advanced standard services that can be integrated into a variety of computer systems. The .NET Framework provides a feature-rich application environment, simplified development and easy integration between a numbers of different development languages.

The .NET Framework is language neutral. Currently it supports C++, C#, Visual Basic, and JScript (Microsoft's version of JavaScript).Microsoft's Visual Studio.NET is a common development environment for the .NET Framework.

Differences between ASP and ASP .NET

ASP .NET has better language support, a large set of new controls and XML based components, and better user authentication.ASP .NET provides increased performance by running compiled code.

ASP .NET Controls

ASP .NET contains a large set of HTML controls. Almost all HTML elements on a page can be defined as ASP .NET control objects that can be controlled by scripts.ASP .NET also contains a new set of object oriented input controls, like programmable list boxes and validation controls. A new data grid control supports sorting, data paging, and everything you expect from a dataset control.

User Authentication

ASP .NET supports forms-based user authentication, including cookie management and automatic redirecting of unauthorized logins.

Easy Configuration

Configuration of ASP .NET is done with plain text files. Configuration files can be uploaded or changed while the application is running. No need to restart the server. No more metabase or registry puzzle.

How Does it Work?

Fundamentally an ASP.NET page is just the same as an HTML page. An HTML page has the extension .htm. If a browser requests an HTML page from the server, the server sends the page to the browser without any modifications. An ASP.NET page has the extension .aspx. If a browser requests an ASP.NET page, the server processes any executable code in the page, before the result is sent back to the browser. The ASP.NET page above does not contain any executable code, so nothing is executed. In the next examples we will add some executable code to the page to demonstrate the difference between static HTML pages and dynamic ASP pages.

ASP.NET - Web Server Controls

Web server controls are special ASP.NET tags understood by the server.Like HTML server controls, Web server controls are also created on the server and they require a runat="server" attribute to work. However, Web server controls do not necessarily map to any existing HTML elements and they may represent more complex elements.The syntax for creating a Web server control is:

<asp:control_name id="some_id" runat="server" />

In the following example we declare a Button server control in an .aspx file. Then we create an event handler for the Click event which changes the text on the button.

Maintaining the View State

When a form is submitted in classic ASP, all form values are cleared. Suppose you have submitted a form with a lot of information and the server comes back with an error. You will have to go back to the form and correct the information. You click the back button, and what happens. ALL form values are CLEARED, and you will have to start all over again! The site did not maintain your ViewState.

When a form is submitted in ASP .NET, the form reappears in the browser window together with all form values. How come? This is because ASP .NET maintains your ViewState. The ViewState indicates the status of the page when submitted to the server. The status is defined through a hidden field placed on each page with a <form runat="server"> control.

ASP.NET - Database Connection

ADO.NET is also a part of the .NET Framework. ADO.NET is used to handle data access. With ADO.NET you can work with databases.

What is ADO.NET?

- ADO.NET is a part of the .NET Framework
- ADO.NET consists of a set of classes used to handle data access
- ADO.NET is entirely based on XML
- ADO.NET has, unlike ADO, no Recordset object

Create a Database Connection

We are going to use the Northwind database in our examples.First, import the "System.Data.OleDb" namespace. We need this namespace to work with Microsoft Access and other OLE DB database providers. We will create the connection to the database in the

Page_Load subroutine. We create a dbconn variable as a new OleDbConnection class with a connection string which identifies the OLE DB provider and the location of the database.

Web Server Controls

Like HTML server controls, Web server controls are also created on the server and they require a runat="server" attribute to work. However, Web server controls do not necessarily map to any existing HTML elements and they may represent more complex elements.

The syntax for creating a Web server control is:

<asp:control_name id="some_id" runat="server" />

3.2.2 About SQL server

All the data in Microsoft® SQL Server 2000 databases is contained in objects called tables. Each table represents some type of object meaningful to the users. For example, in a school database you would find tables such as a class table, an instructor table, and a student table.SQL Server provides several data type synonyms to help support SQL-92 data type names not included as base data types, such as national character and character varying. When a synonym is specified in a CREATE TABLE statement, the column is assigned the base data type associated with the synonym. For more information, see Data Type Synonyms.

A domain is the set of all allowable values in a column. It includes not only the concept of enforcing data types, but also the values allowed in the column. For example, a part color domain would include both the data type, such as **char(6)**, and the character strings allowed in the column, such as Red, Blue, Green, Yellow, Brown, Black, White, Teal, Grey, and Silver. Domain values can be enforced through mechanisms such as CHECK constraints and triggers.

System Tables

SQL Server stores the data defining the configuration of the server and all its tables in a special set of tables known as system tables. SQL Server does not support direct updates to the system tables by users or applications. Only SQL Server should update the system tables in response to administration commands issued by users. Additionally, users should not query the system tables directly unless there is no other way to get the data required by the application. The system tables can change from version to version; applications referencing system tables directly may have to be rewritten before they can be upgraded to a newer version of SQL Server with a different version of the system tables.

Temporary Tables

SQL Server supports temporary tables. These tables have names that start with a number sign (#). If a temporary table is not dropped when a user disconnects, SQL Server automatically drops the temporary table. Temporary tables are not stored in the current database; they are stored in the **tempdb** system database.There are two types of temporary tables:

- Local temporary tables

  The names of these tables begin with one number sign (#). These tables are visible only to the connection that created them.

- Global temporary tables

  The names of these tables begin with two number signs (##). These tables are visible to all connections. If the tables are not dropped explicitly before the connection that created them disconnects, they are dropped as soon as all other tasks stop referencing them. No new tasks can reference a global temporary table after the connection that created it disconnects. The association between a task and a table is always dropped when the current statement completes executing; therefore, global temporary tables are usually dropped soon after the connection that created them disconnects.

Fundamentals of SQL Server 2000 Architecture

Microsoft® SQL Serve 2000 is a family of products that meet the data storage requirements of the largest data processing systems and commercial Web sites, yet at the same time can provide easy-to-use data storage services to an individual or small business.

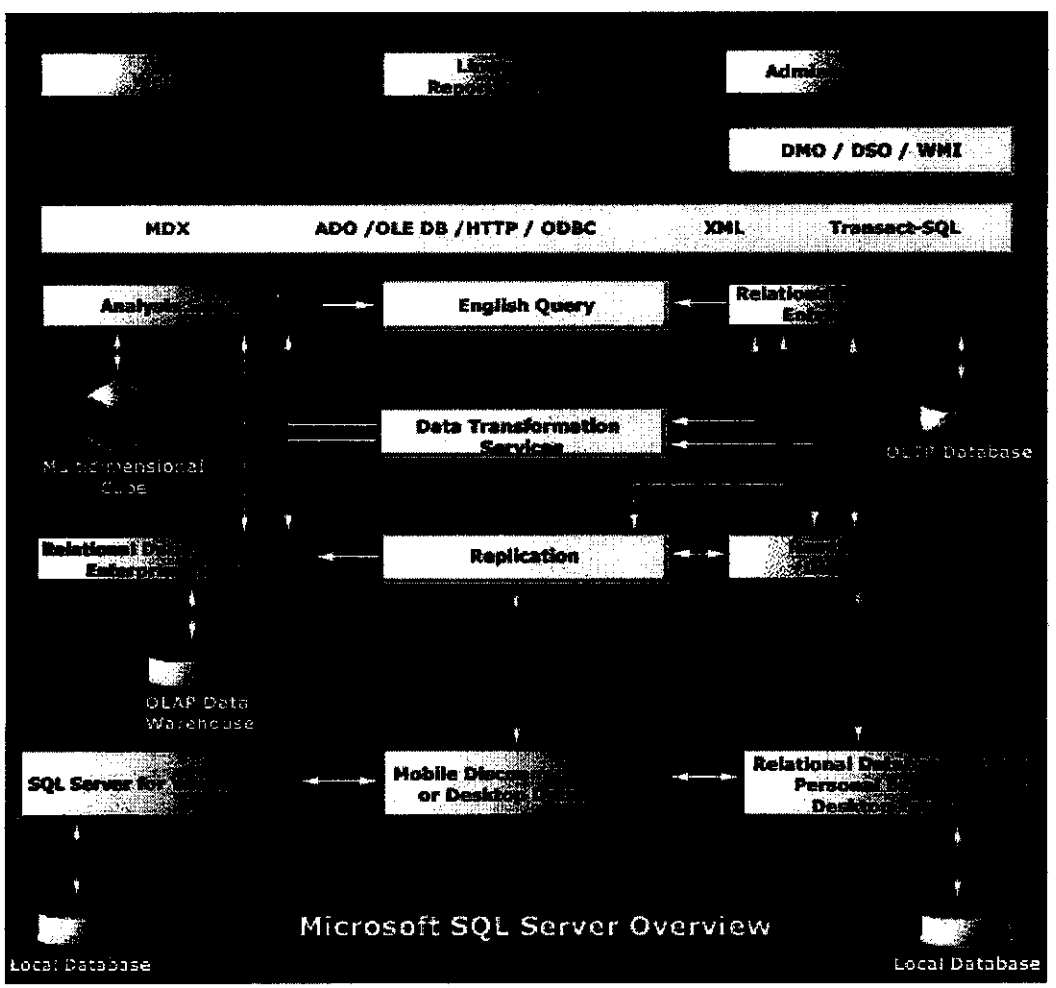
The data storage needs of a modern corporation or government organization are very complex. Some examples are:

- Online Transaction Processing (OLTP) systems must be capable of handling thousands of orders placed at the same time.
- Increasing numbers of corporations are implementing large Web sites as a mechanism for their customers to enter orders, contact the service department, get information about products, and for many other tasks that previously required contact with employees. These sites require data storage that is secure, yet tightly integrated with the Web.
- Organizations are implementing off-the-shelf software packages for critical services such as human resources planning, manufacturing resources planning, and inventory control. These systems require databases capable of storing large amounts of data and supporting large numbers of users.
- Organizations have many users who must continue working when they do not have access to the network. Examples are mobile disconnected users, such as traveling sales representatives or regional inspectors. These users must synchronize the data on a notebook or laptop with the current data in the corporate system, disconnect from the network, record the results of their work while in the field, and then finally reconnect with the corporate network and merge the results of their fieldwork into the corporate data store.
- Managers and marketing personnel need increasingly sophisticated analysis of trends recorded in corporate data. They need robust Online Analytical Processing (OLAP) systems easily built from OLTP data and support sophisticated data analysis.

- Independent Software Vendors (ISVs) must be able to distribute data storage capabilities with applications targeted at individuals or small workgroups. This means the data storage mechanism must be transparent to the users who purchase the application. This requires a data storage system that can be configured by the application and then tune itself automatically so that the users do not need to dedicate database administrators to constantly monitor and tune the application.

SQL Server 2000 Component Overview

This diagram is an illustration of the relationships between the major components:



ALTER TABLE

TABLE

Is the name of the table to be altered. If the table is not in the current database or owned by the current user, the database and owner can be explicitly specified.

ALTER COLUMN

Specifies that the given column is to be changed or altered. ALTER COLUMN is not allowed if the compatibility level is 65 or earlier.

The altered column cannot be:

- A column with a **text**, **image**, **ntext**, or **timestamp** data type.
- The ROWGUIDCOL for the table.
- A computed column or used in a computed column.
- A replicated column.
- Used in an index, unless the column is a **varchar**, **nvarchar**, or **varbinary** data type, the data type is not changed, and the new size is equal to or larger than the old size.
- Used in statistics generated by the CREATE STATISTICS statement. First remove the statistics using the DROP STATISTICS statement. Statistics automatically generated by the query optimizer are automatically dropped by ALTER COLUMN.
- Used in a PRIMARY KEY or [FOREIGN KEY] REFERENCES constraint.
- Used in a CHECK or UNIQUE constraint, except that altering the length of a variable-length column used in a CHECK or UNIQUE constraint is allowed.

- Associated with a default, except that changing the length, precision, or scale of a column is allowed if the data type is not changed.

ALTER TRIGGER

Alters the definition of a trigger created previously by the CREATE TRIGGER statement. For more information about the parameters used in the ALTER TRIGGER statement, see CREATE TRIGGER.

TRIGGER NAME

Is the existing trigger to alter.

TABLE|VIEW

Is the table or view on which the trigger is executed.

WITH ENCRYPTION

Encrypts the **syscomments** entries that contain the text of the ALTER TRIGGER statement. Using WITH ENCRYPTION prevents the trigger from being published as part of SQL Server replication.

Creating a Stored Procedure

You can create stored procedures using the CREATE PROCEDURE Transact-SQL statement. Before creating a stored procedure, consider that:

- CREATE PROCEDURE statements cannot be combined with other SQL statements in a single batch.
- Permission to create stored procedures defaults to the database owner, who can transfer it to other users.

- Stored procedures are database objects, and their names must follow the rules for identifiers.
- You can create a stored procedure only in the current database.

When creating a stored procedure, you should specify:

- Any input parameters and output parameters to the calling procedure or batch.
- The programming statements that perform operations in the database, including calling other procedures.
- The status value returned to the calling procedure or batch to indicate success or failure (and the reason for failure).

System Stored Procedures

Many of your administrative activities in Microsoft® SQL Server™ 2000 are performed through a special kind of procedure known as a system stored procedure. System stored procedures are created and stored in the **master** database and have the **sp_** prefix. System stored procedures can be executed from any database without having to qualify the stored procedure name fully using the database name **master**.

It is strongly recommended that you do not create any stored procedures using **sp_** as a prefix. SQL Server always looks for a stored procedure beginning with **sp_** in this order:

1. The stored procedure in the **master** database.
2. The stored procedure based on any qualifiers provided (database name or owner).
3. The stored procedure using **dbo** as the owner, if one is not specified.

Therefore, although the user-created stored procedure prefixed with **sp_** may exist in the current database, the **master** database is always checked first, even if the stored procedure is qualified with the database name.

SAVE TRANSACTION

Sets a savepoint within a transaction.

Syntax

SAVE TRAN [SACTION] { *savepoint_name* | *@savepoint_variable* }

savepoint_name

Is the name assigned to the savepoint. Savepoint names must conform to the rules for identifiers, but only the first 32 characters are used.

CHAPTER 4

4. SYSTEM ANALYSIS AND DESIGN

4.1 Modules

4.1.1 User Creation

It consists of two types of user named as administrator and general. Administrator has the rights to create the general user. The general users can entirely utilize this software in the absence of administrator with some restrictions. Both administrator and general users can login to this software with their respective ID and password.

4.1.2 Employee information

The entire employee information is stored into this software using this module of a particular organization. This module provides the features such as: The employee ID will be automatically generated, the information of the existing user can be found, the employee information can also be deleted.

4.1.3 Client information

The information about the new clients is stores in this module. It provides the similar features like employee information.

4.1.4 Project Environment details

In PATS, the different type of software development environment is stored using the environment ID and cost for modules is entered for the particular. Hence, for further incoming projects it is enough to mention the environment ID rather than entering the complete data once again.

4.1.5 Project Order placing

Here the actual order is placed with a creation of new project ID and the demand from the client, estimated date to finish is taken.

4.1.6 Project details Entry

Here the reusability concept is implemented. It is done in two ways:

- Same environment
- Same project title

If any project has been done in the same environment previously, then the system matches the environment and displays the project details. The environment is identified using the project ID.

The project for which the order has been placed is matched with the previously done project titles in order to implement the reusability concept.

4.1.7 Project Allocation

In this phase the project is allocated to the respective staffs of the organization thereby checking the availability of the staffs. Each employee of the organization is noted with their specialization in the employee information module. By giving the environment ID in this module the concerned employee of that particular environment will be listed based upon their efficiency. The efficiency of the staff is identified by certain resource modules for example allocating points based on their experience and completed project etc.

If the man power needed for the particular project is not sufficient then the efficient staff that has ranked first among all the areas will be assigned the project. Initially, when allocating the staff, the current status of the project has been saved as 0%.Depending upon the completion of the project each day the status should be updated by each employee and it is saved. Whenever it reaches 100%, the project is said to be completed. So, the employee who has been in the non availability list will be marked as available.

4.1.8 Cost estimation

In this module the total cost is estimated for the project by calculating the attributes like number of persons needed, number of reports, number of modules, development environment etc.

Cost Estimation of PATS

Estimating the cost of a software system is one of the most difficult and error prone tasks in software engineering. It is difficult to make and accurate cost estimate during the planning phase of software development, because of the large number of unknown factors at that time.

There are many factors that influence the cost software system. The effect of most of these factors and hence the cost of development and maintenance efforts are difficult to estimate Primary among the cost factors are individual abilities of personnel and the familiarity with the application area. Some of the major factors that influence the cost of the software products are Programmability, Product Complexity, Product Size, Available Time, Technology used. The availability, Familiarity and Stability of the system are used. Based on the above mentioned factors, the cost estimation of Software developed would be around Rs. 7000/-

4.1.9 Reports

It provides the overall reports about all the above mentioned modules such as projects, employees, clients, ongoing projects, completed projects.

4.1.10 status of employee and project

This module describes what project has been allocated to the employee and how far the project has been completed. The employees in the organization may not be aware of their present status. By using this each employee can look at their status i.e whether any new project has allotted. The new project allotted will be specified with the status 0%. The employees can also find their team members by giving the particular project id specified.

4.2 DATABASE DESIGN

4.2.1 ALLOCATION

ProjectID	nchar(6)
EmpID	nchar(6)
Module	nchar(50)
DateFinish	datetime
CStatus	nchar(10)
CSDate	smalldatetime

4.2.2 CLIENT

ClientID	nchar(6)
CName	nchar(50)
Address	nchar(50)
CPerson	nchar(30)
ContactNo	nchar(30)
Mailed	nchar(30)
Website	nchar(30)
NatWork	nchar(30)
RefBy	nchar(30)

4.2.3EMPLOYEE

Empno	nchar(6)
Ename	nchar(30)
FHname	nchar(30)
Address	nchar(60)
State	nchar(30)
Dob	datetime
Gender	char(1)
Qualify	nchar(50)
Doj	datetime
Pexp	float
Pdesig	nchar(30)
Cdesig	nchar(30)
ContactNo	nchar(30)
Emailed	nchar(30)
Dept	nchar(20)
Spec	nchar(6)
Availability	nchar(1)

4.2.4 PROJECT DETAILS

ProjectID	nchar(6)
PENVID	nchar(6)
Documents	nchar(30)
NPersons	int
EsCost	int
Demand	int
PStatus	nchar(5)
SDate	datetime

4.2.5 PROJECT ENVIRONMENT

PENVID	nchar(6)
FrontEnd	nchar(30)
BackEnd	nchar(30)
CPerModule	numeric(7, 0)
CPerSubModule	numeric(7, 0)
CPerReport	numeric(7, 0)
FVersion	nchar(20)
BVersion	nchar(20)

CPerson	numeric(7, 0)

4.2.6 USER CREATION

LoginID	nchar(10)
Empno	nchar(6)
Password	nchar(30)
UserType	char(1)
Doc	nchar(10)
CreatedBy	nchar(6)

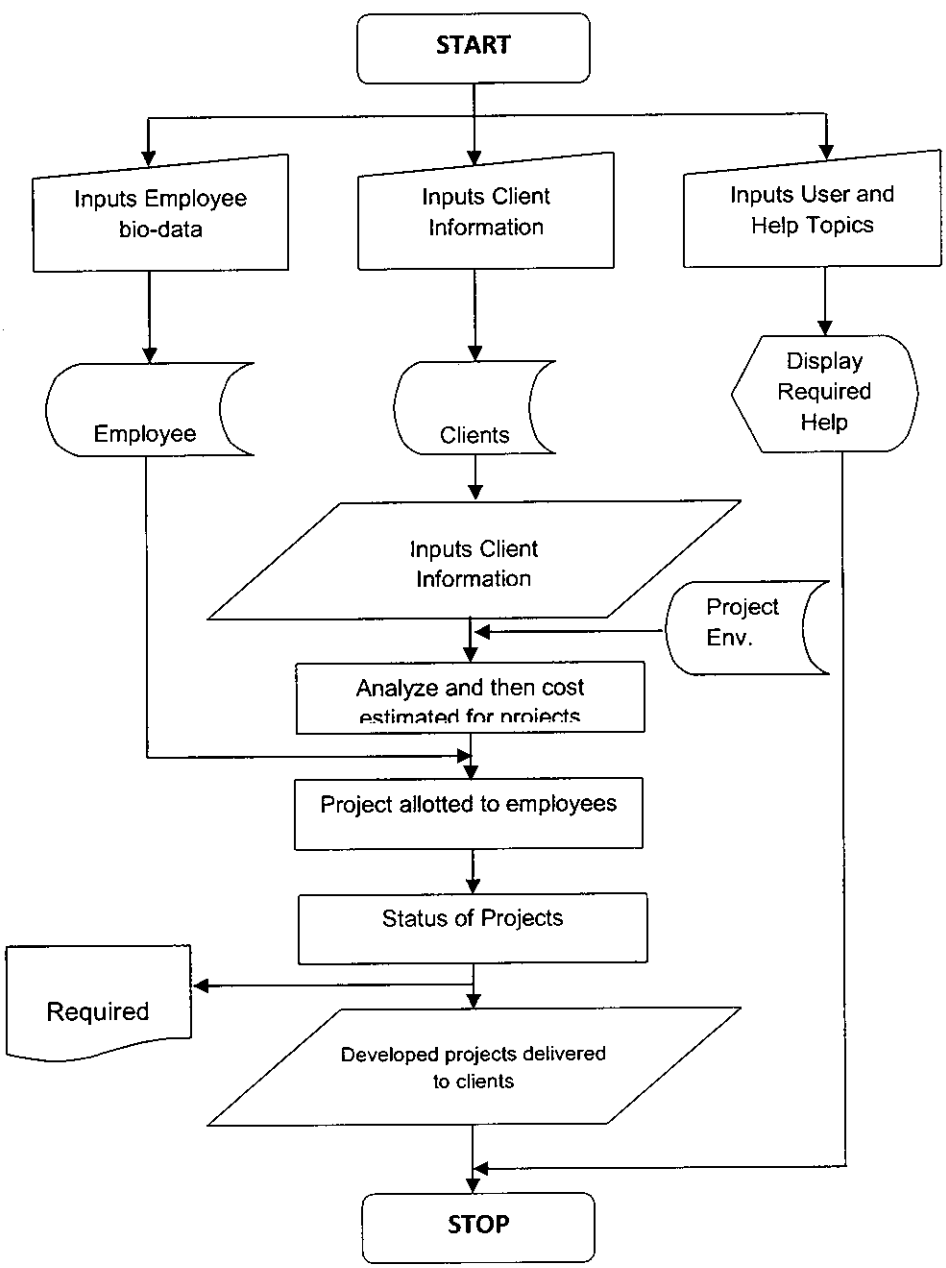
4.2.7 PROJECT ORDER PLACING

ProjectID	nchar(6)
ClientID	nchar(6)
ProjectTitle	nchar(50)
EsDate	datetime
ExDate	datetime
Remarks	nchar(50)
Dord	datetime

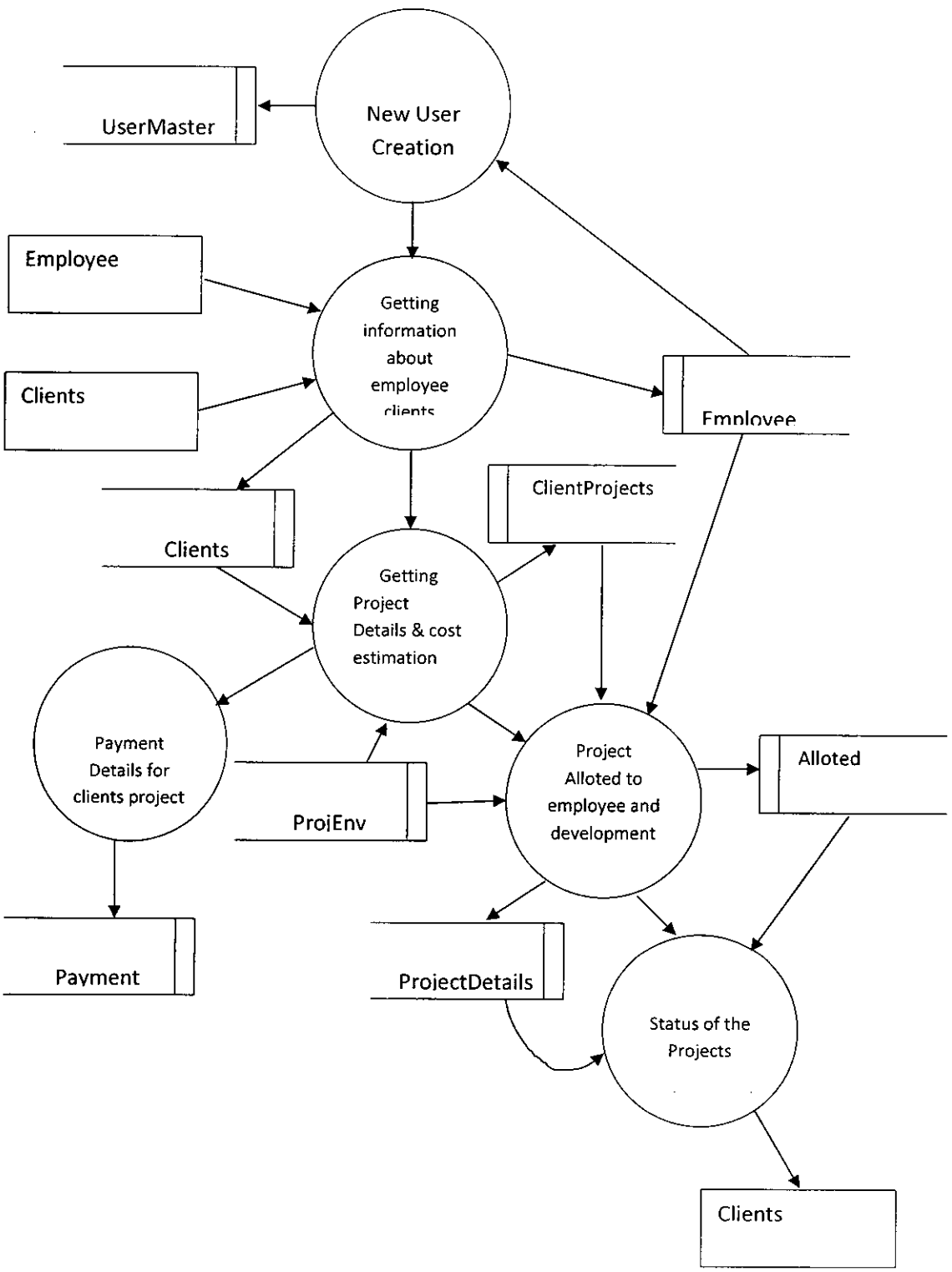
CHAPTER 5

5. SYSTEM DEVELOPMENT

System Flow Diagram (Fig. 5.1)



DATA FLOW DIAGRAM (Fig.5.2)



CHAPTER 6
6. SYSTEM TESTING

System testing is the implementation that is aimed at ensuring that system works accurately and efficiency, before live operation begins. System testing involves the capability of modules rather than integration of modules, Objective, current specification and system documentation.

6.1 Stages in the testing process:

6.1.1 **Unit testing**

In unit testing, individual components are tested to ensure that they operate correctly. Each component is tested independently, without any other system components.

6.1.2 **Module testing**

A module is the collection of dependent components such as object class, an abstract data type or some collection of Procedures and Functions. The module encapsulates related components that can be tested without other system modules.

6.1.3 **System testing**

The sub- systems are integrated to build up the entire system. The testing process is concerned with finding errors.

6.1.4 **Accepting testing**

This is the final stage in the testing process before the system is accepted for operational use.

CHAPTER 7

7. CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

ADVANTAGES:

- In an organization any one can be act as administrator and can get the order from the client.
- Availability of the staff can be automatically identified and allotted.
- Less communication is needed.
- Each employee can identify their status by themselves.
- Reusable modules from previously done projects can be easily identified.
- Reports of the overall project can be extracted in a fast way.
- This project provides an easy way of allocation, updation and searching.

DISADVANTAGES:

- This software is not entirely automated, certain manual entries are required.
- This software is completely online based for the employee organization and not for the clients.

7.2 FUTURE ENHANCEMENT

There is always a room for improvements in any sort of area. Even Microsoft always thinks of producing the enhanced version after developing software.

Here this software can be entirely automated in future. It can be modified to a higher level thereby providing rights for the clients to place the order online by providing their proof and also making the payments to be done by credit and debit cards.

APPENDICES

A.1 SAMPLE CODING

ALLOCATION

```
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;
using System.Data.SqlClient;

public partial class frmAllocation : System.Web.UI.Page
{
    SqlConnection mycon;
    SqlCommand cmd;
    SqlDataReader dread;
    String sql;
    Int32 np;
    String pen;
    String pd, en, am, df, cs, sd,rDt;
    protected void Page_Load(object sender, EventArgs e)
    {
        mycon = new SqlConnection("Data Source=HOME-
ADE326A655\\SQLEXPRESS;Initial Catalog=dbPATS;Integrated Security=True");
        mycon.Open();
    }
    protected void ddlProjectID_SelectedIndexChanged(object sender, EventArgs
e)
    {
        int ncount = 0;
        StatusGrid.Visible = false;
        OthersGrid.Visible = false;
        ClearFields();
        cmd = new SqlCommand("Select PenVID,NPersons From ProjectDetails
Where ProjectID='" + ddlProjectID.Text.Trim() + "'", mycon);
        dread = cmd.ExecuteReader();
        if (dread.HasRows == true)
        {
            dread.Read();
            pen = dread.GetString(0);
```

```
            np = dread.GetInt32(1);
            lblPEN.Text = pen;
        }
        dread.Close();
        sql = "Select * From EfMaster Where PENID='"+pen+"' and Avail='Y'
Order by Points Desc";
        cmd = new SqlCommand(sql, mycon);
        dread = cmd.ExecuteReader();
        if (dread.HasRows == true)
        {
            StatusGrid.Visible = true;
            StatusGrid.DataSource = dread;
            StatusGrid.DataBind();
        }
        dread.Close();
        sql = "Select * From EfMaster Where PENID='" + pen + "' and Avail='Y'
Order by Points Desc";
        cmd = new SqlCommand(sql, mycon);
        dread = cmd.ExecuteReader();
        ddlEmpID.Items.Clear();
        ddlEmpID.Items.Add("-Select-");
        while (dread.Read())
        {
            ddlEmpID.Items.Add(dread.GetString(0));
            ncount++;
        }
        dread.Close();
        if (ncount < np)
        {
            sql = "Select * From EfMaster Where PENID<>'" + pen + "' and
Avail='Y' Order by Points Desc";
            cmd = new SqlCommand(sql, mycon);
            dread = cmd.ExecuteReader();
            if (dread.HasRows == true)
            {
                OthersGrid.Visible = true;
                OthersGrid.DataSource = dread;
                OthersGrid.DataBind();
            }
            dread.Close();
            cmd = new SqlCommand(sql, mycon);
            dread = cmd.ExecuteReader();
            while (dread.Read())
            {
                ddlEmpID.Items.Add(dread.GetString(0));
                ncount--;
                if (ncount <= 0) break;
            }
            dread.Close();

            sql = "Select * From EfMaster Where Avail='N'  Order by Points
Desc";
            cmd = new SqlCommand(sql, mycon);
            dread = cmd.ExecuteReader();
            ddlUEmpID.Items.Clear();
```

```
            ddlUEmpID.Items.Add("-Select-");
            while (dread.Read())
               ddlUEmpID.Items.Add(dread.GetString(0));
            dread.Close();
        }
    }
    protected void btnList_Click(object sender, EventArgs e)
    {
        int percent;
        MoveData();
        sql = "Update Allocation Set CStatus='" + txtCStatus.Text +
"',CSDate='" + sd + "' Where ProjectID='" + pd + "' and
EmpID='"+ddlUEmpID.Text+"'";
        cmd = new SqlCommand(sql, mycon);
        cmd.ExecuteNonQuery();
        percent = Convert.ToInt32(txtCStatus.Text);
        ClearFields();
        if (percent >= 100)
        {
            cmd = new SqlCommand("Select PenVid From ProjectDetails Where
ProjectID='" + ddlProjectID.Text + "'", mycon);
            dread = cmd.ExecuteReader();
            if (dread.HasRows)
            {
                dread.Read();
                pen=dread.GetString(0);
            }
            dread.Close();
            cmd = new SqlCommand("Update EfMaster Set
Points=Points+10,Avail='Y' Where EmpID='"+ddlUEmpID.Text+"' and
Penid='"+pen+"'", mycon);
            cmd.ExecuteNonQuery();
        }

        cmd.Dispose();

        cmd = new SqlCommand("Select count(ProjectID) From Allocation Where
ProjectID='" + ddlProjectID.Text + "' and EmpID='"+ddlUEmpID.Text +"' and
CStatus<>'100'", mycon);
        dread = cmd.ExecuteReader();
        if (dread.HasRows)
        {
            dread.Read();
            if (dread.GetInt32(0) ==0)
            {
                dread.Close();
                cmd = new SqlCommand("Update ProjectDetails Set PStatus='100'
Where ProjectID='" + ddlProjectID.Text + "' and PenvID='" + pen + "'",
mycon);
                cmd.ExecuteNonQuery();
                cmd.Dispose();
            }
        }
        dread.Close();
        lblSTATUS.Text = "The Current Status has been updated";
```

```
    }

    protected void btnFind_Click(object sender, EventArgs e)
    {
        cmd = new SqlCommand("Select * From Allocation Where ProjectID='" +
ddlProjectID.Text + "' and EmpID='"+ddlUEmpID.Text +"'", mycon);
        dread = cmd.ExecuteReader();
        if (dread.HasRows == true)
        {
            dread.Read();
            DisplayData();
        }
        else lblSTATUS.Text = "The Given Employee No. is Not Found";
        dread.Close();
    }
    void SplitDate(String dt)
    {
        int s, e;
        String dd, mm, yy;
        s = dt.IndexOf(".");
        e = dt.LastIndexOf(".");
        mm = dt.Substring(s + 1, 2);
        dd = dt.Substring(0, s);
        yy = dt.Substring(e + 1, 4);
        rDt = mm + "." + dd + "." + yy;
    }
    void MoveData()
    {
        pd=ddlProjectID.Text;
        en = ddlEmpID.Text.Substring(0,4);
        am = txtAllModule.Text;
        SplitDate(txtDateFinish.Text);
        df = rDt;
        cs = txtCStatus.Text;
        SplitDate(txtStatusDate.Text);
        sd = rDt;
    }
    void DisplayData()
    {
        int s, e;
        String dd, mm, yy,dt;
        ddlProjectID.Text = dread.GetString(0);
        ddlUEmpID.Text = dread.GetString(1);
        txtAllModule.Text = dread.GetString(2);
        dt = dread.GetDateTime(3).ToShortDateString();
        s = dt.IndexOf("/");
        e = dt.LastIndexOf("/");
        mm = dt.Substring(s + 1, 2);
        dd = dt.Substring(0, s);
        yy = dt.Substring(e + 1, 4);
        //txtStatusDate.Text = dd + "." + mm + "." + yy;
        txtDateFinish.Text = dd + "." + mm + "." + yy;
        txtCStatus.Text = dread.GetString(4);
        dt = dread.GetDateTime(5).ToShortDateString();
```

```
        s = dt.IndexOf("/");
        e = dt.LastIndexOf("/");
        mm = dt.Substring(s + 1, 2);
        dd = dt.Substring(0, s);
        yy = dt.Substring(e + 1, 4);
        txtStatusDate.Text = dd + "." + mm + "." + yy;


    }
    void ClearFields()
    {
        txtAllModule.Text = "";
        txtDateFinish.Text = "";
        txtCStatus.Text = "";
        txtStatusDate.Text = "";
        StatusGrid.Visible = false;
        OthersGrid.Visible = false;
    }
    protected void btnSave_Click(object sender, EventArgs e)
    {
        MoveData();
        sql = "Insert into Allocation Values('" + pd + "','" + en + "','" +
am + "','" + df + "','" + cs + "','" + sd + "')";
        cmd = new SqlCommand(sql, mycon);
        cmd.ExecuteNonQuery();
        lblSTATUS.Text = "This Allocation is done Successfully";
        cmd.Dispose();
        cmd = new SqlCommand("Update EfMaster Set Avail='N' Where EmpID='" +
en + "' and Penid='" + lblPEN.Text  + "'", mycon);
        cmd.ExecuteNonQuery();
        cmd.Dispose();
        ClearFields();
    }
}
```

PROJECT DETAILS

```
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;
using System.Data.SqlClient;

public partial class frmAllocation : System.Web.UI.Page
{
```

```
    SqlConnection mycon;
    SqlCommand cmd;
    SqlDataReader dread;
    String sql;
    Int32 np;
    String pen;
    String pd, en, am, df, cs, sd,rDt;
    protected void Page_Load(object sender, EventArgs e)
    {
        mycon = new SqlConnection("Data Source=HOME-
ADE326A655\\SQLEXPRESS;Initial Catalog=dbPATS;Integrated Security=True");
        mycon.Open();
    }
    protected void ddlProjectID_SelectedIndexChanged(object sender, EventArgs
e)
    {
        int ncount = 0;
        StatusGrid.Visible = false;
        OthersGrid.Visible = false;
        ClearFields();
        cmd = new SqlCommand("Select PenVID,NPersons From ProjectDetails
Where ProjectID='" + ddlProjectID.Text.Trim() + "'", mycon);
        dread = cmd.ExecuteReader();
        if (dread.HasRows == true)
        {
            dread.Read();
            pen = dread.GetString(0);
            np = dread.GetInt32(1);
            lblPEN.Text = pen;
        }
        dread.Close();
        sql = "Select * From EfMaster Where PENID='"+pen+"' and Avail='Y'
Order by Points Desc";
        cmd = new SqlCommand(sql, mycon);
        dread = cmd.ExecuteReader();
        if (dread.HasRows == true)
        {
            StatusGrid.Visible = true;
            StatusGrid.DataSource = dread;
            StatusGrid.DataBind();
        }
        dread.Close();
        sql = "Select * From EfMaster Where PENID='" + pen + "' and Avail='Y'
Order by Points Desc";
        cmd = new SqlCommand(sql, mycon);
        dread = cmd.ExecuteReader();
        ddlEmpID.Items.Clear();
        ddlEmpID.Items.Add("-Select-");
        while (dread.Read())
        {
            ddlEmpID.Items.Add(dread.GetString(0));
            ncount++;
        }
        dread.Close();
        if (ncount < np)
```

```
        {
            sql = "Select * From EfMaster Where PENID<>'" + pen + "' and
Avail='Y' Order by Points Desc";
            cmd = new SqlCommand(sql, mycon);
            dread = cmd.ExecuteReader();
            if (dread.HasRows == true)
            {
                OthersGrid.Visible = true;
                OthersGrid.DataSource = dread;
                OthersGrid.DataBind();
            }
            dread.Close();
            cmd = new SqlCommand(sql, mycon);
            dread = cmd.ExecuteReader();
            while (dread.Read())
            {
                ddlEmpID.Items.Add(dread.GetString(0));
                ncount--;
                if (ncount <= 0) break;
            }
            dread.Close();

            sql = "Select * From EfMaster Where Avail='N'  Order by Points
Desc";
            cmd = new SqlCommand(sql, mycon);
            dread = cmd.ExecuteReader();
            ddlUEmpID.Items.Clear();
            ddlUEmpID.Items.Add("-Select-");
            while (dread.Read())
               ddlUEmpID.Items.Add(dread.GetString(0));
            dread.Close();
        }
    }
    protected void btnList_Click(object sender, EventArgs e)
    {
        int percent;
        MoveData();
        sql = "Update Allocation Set CStatus='" + txtCStatus.Text +
"',CSDate='" + sd + "' Where ProjectID='" + pd + "' and
EmpID='"+ddlUEmpID.Text+"'";
        cmd = new SqlCommand(sql, mycon);
        cmd.ExecuteNonQuery();
        percent = Convert.ToInt32(txtCStatus.Text);
        ClearFields();
        if (percent >= 100)
        {
            cmd = new SqlCommand("Select PenVid From ProjectDetails Where
ProjectID='" + ddlProjectID.Text + "'", mycon);
            dread = cmd.ExecuteReader();
            if (dread.HasRows)
            {
                dread.Read();
                pen=dread.GetString(0);
            }
            dread.Close();
```

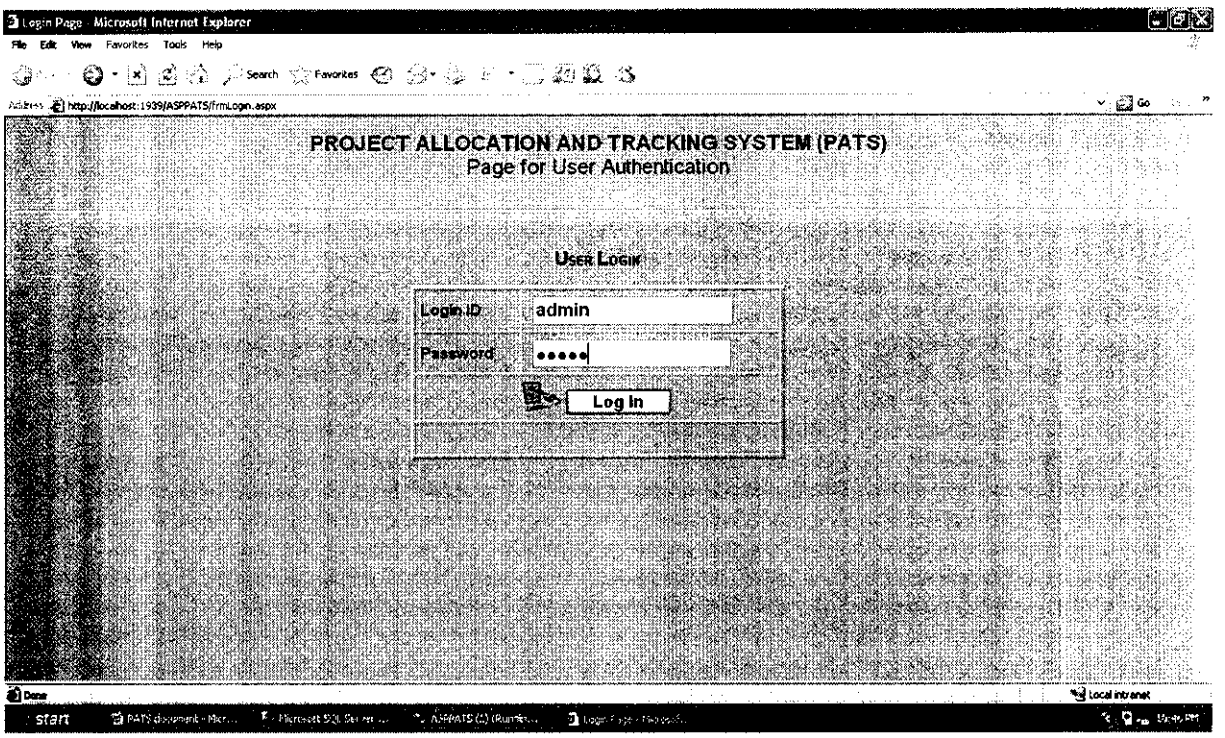
```
            cmd = new SqlCommand("Update EfMaster Set
Points=Points+10,Avail='Y' Where EmpID='"+ddlUEmpID.Text+"' and
Penid='"+pen+"'", mycon);
            cmd.ExecuteNonQuery();
        }

        cmd.Dispose();

        cmd = new SqlCommand("Select count(ProjectID) From Allocation Where
ProjectID='" + ddlProjectID.Text + "' and EmpID='"+ddlUEmpID.Text +"' and
CStatus<>'100'", mycon);
        dread = cmd.ExecuteReader();
        if (dread.HasRows)
        {
            dread.Read();
            if (dread.GetInt32(0) ==0)
            {
                dread.Close();
                cmd = new SqlCommand("Update ProjectDetails Set PStatus='100'
Where ProjectID='" + ddlProjectID.Text + "' and PenvID='" + pen + "'",
mycon);
                cmd.ExecuteNonQuery();
                cmd.Dispose();
            }
        }
        dread.Close();
        lblSTATUS.Text = "The Current Status has been updated";
    }


    protected void btnFind_Click(object sender, EventArgs e)
    {
        cmd = new SqlCommand("Select * From Allocation Where ProjectID='" +
ddlProjectID.Text + "' and EmpID='"+ddlUEmpID.Text +"'", mycon);
        dread = cmd.ExecuteReader();
        if (dread.HasRows == true)
        {
            dread.Read();
            DisplayData();
        }
        else lblSTATUS.Text = "The Given Employee No. is Not Found";
        dread.Close();
    }
    void SplitDate(String dt)
    {
        int s, e;
        String dd, mm, yy;
        s = dt.IndexOf(".");
        e = dt.LastIndexOf(".");
        mm = dt.Substring(s + 1, 2);
        dd = dt.Substring(0, s);
        yy = dt.Substring(e + 1, 4);
        rDt = mm + "." + dd + "." + yy;
    }
    void MoveData()
```

```
    {
        pd=ddlProjectID.Text;
        en = ddlEmpID.Text.Substring(0,4);
        am = txtAllModule.Text;
        SplitDate(txtDateFinish.Text);
        df = rDt;
        cs = txtCStatus.Text;
        SplitDate(txtStatusDate.Text);
        sd = rDt;
    }
    void DisplayData()
    {
        int s, e;
        String dd, mm, yy,dt;
        ddlProjectID.Text = dread.GetString(0);
        ddlUEmpID.Text = dread.GetString(1);
        txtAllModule.Text = dread.GetString(2);
        dt = dread.GetDateTime(3).ToShortDateString();
        s = dt.IndexOf("/");
        e = dt.LastIndexOf("/");
        mm = dt.Substring(s + 1, 2);
        dd = dt.Substring(0, s);
        yy = dt.Substring(e + 1, 4);
        //txtStatusDate.Text = dd + "." + mm + "." + yy;
        txtDateFinish.Text = dd + "." + mm + "." + yy;
        txtCStatus.Text = dread.GetString(4);
        dt = dread.GetDateTime(5).ToShortDateString();

        s = dt.IndexOf("/");
        e = dt.LastIndexOf("/");
        mm = dt.Substring(s + 1, 2);
        dd = dt.Substring(0, s);
        yy = dt.Substring(e + 1, 4);
        txtStatusDate.Text = dd + "." + mm + "." + yy;


    }
    void ClearFields()
    {
        txtAllModule.Text = "";
        txtDateFinish.Text = "";
        txtCStatus.Text = "";
        txtStatusDate.Text = "";
        StatusGrid.Visible = false;
        OthersGrid.Visible = false;
    }
    protected void btnSave_Click(object sender, EventArgs e)
    {
        MoveData();
        sql = "Insert into Allocation Values('" + pd + "','" + en + "','" +
am + "','" + df + "','" + cs + "','" + sd + "')";
        cmd = new SqlCommand(sql, mycon);
        cmd.ExecuteNonQuery();
        lblSTATUS.Text = "This Allocation is done Successfully";
        cmd.Dispose();
```

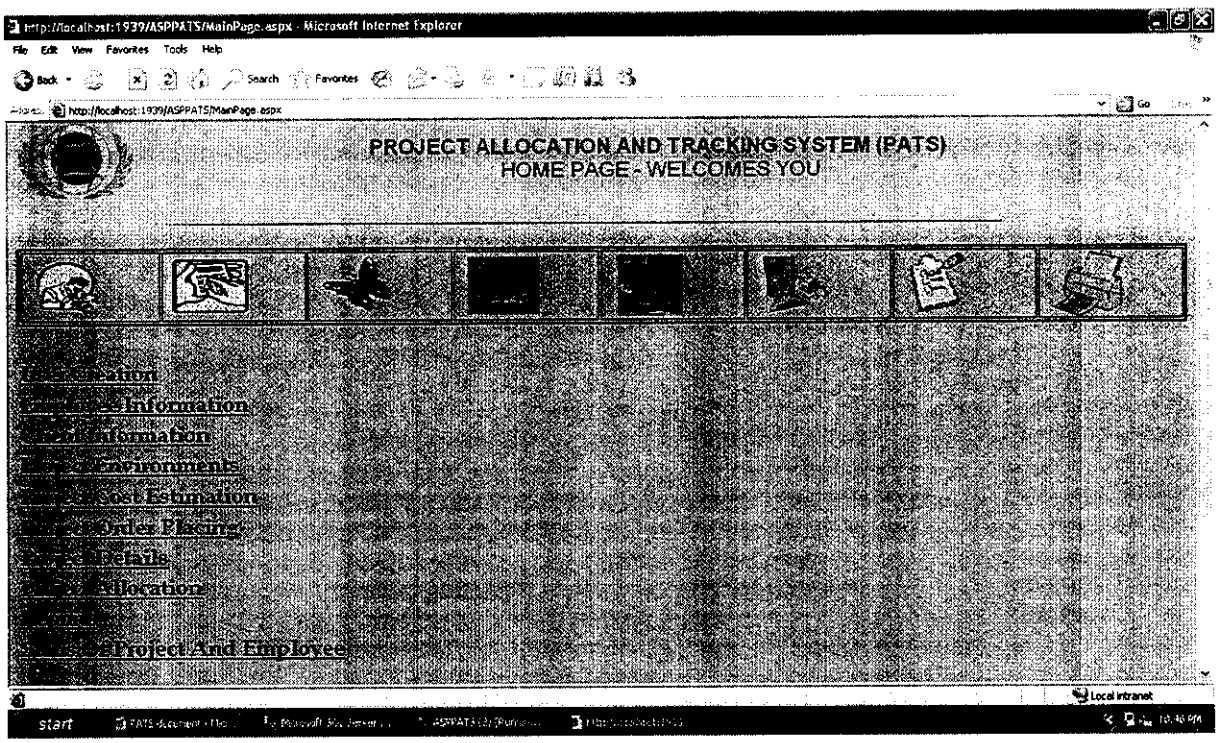
```
        cmd = new SqlCommand("Update EfMaster Set Avail='N' Where EmpID='" +
en + "' and Penid='" + lblPEN.Text  + "'", mycon);
        cmd.ExecuteNonQuery();
        cmd.Dispose();
        ClearFields();
    }
}
```

A.2 SAMPLE OUTPUT

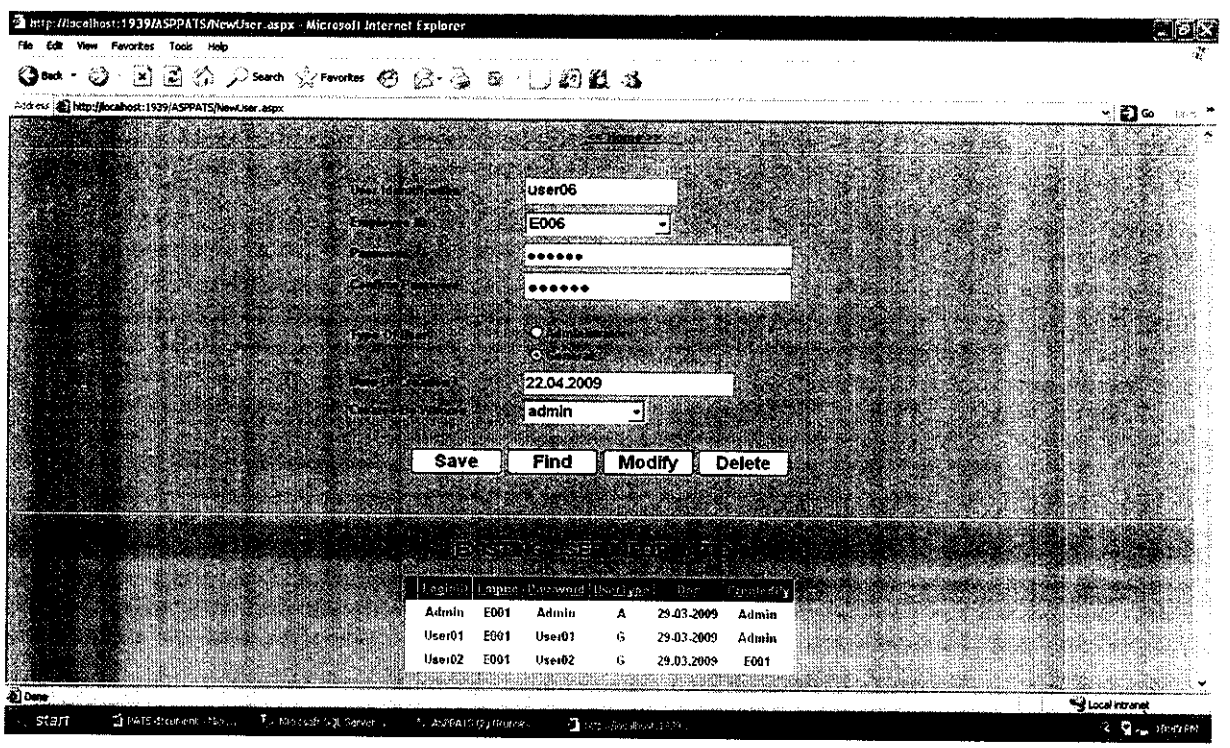
USER LOGIN



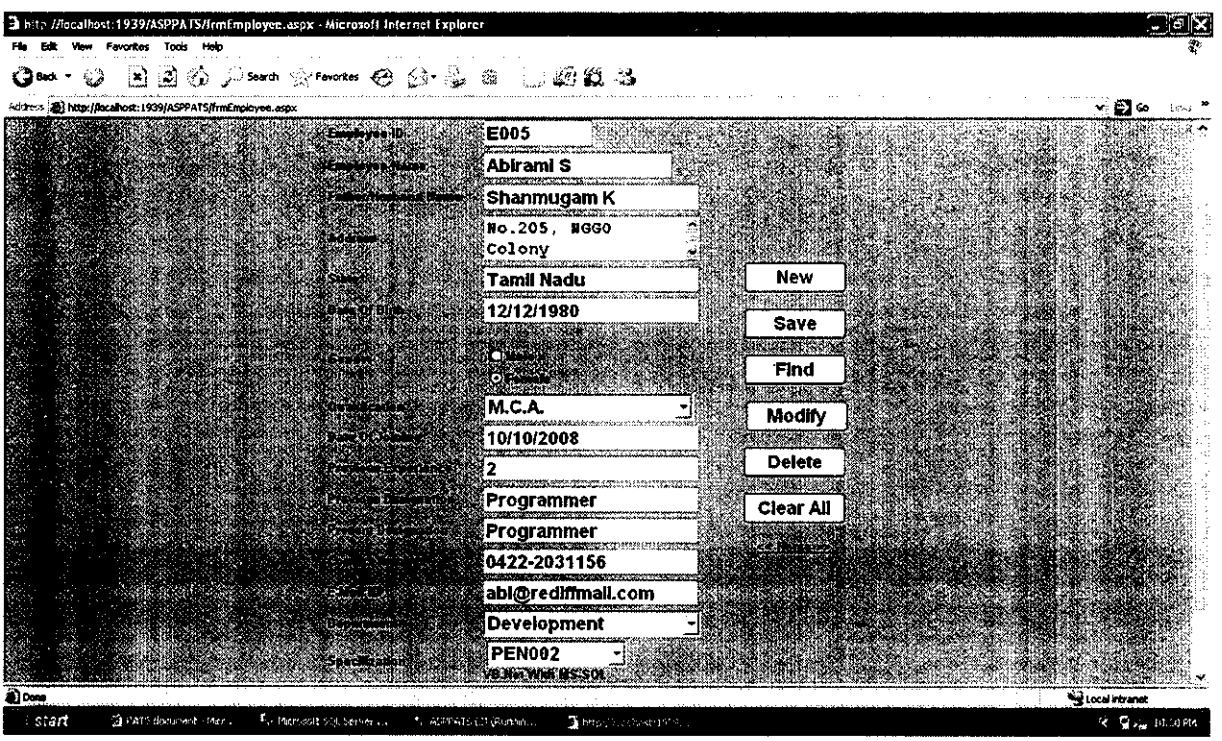
WELCOME PAGE



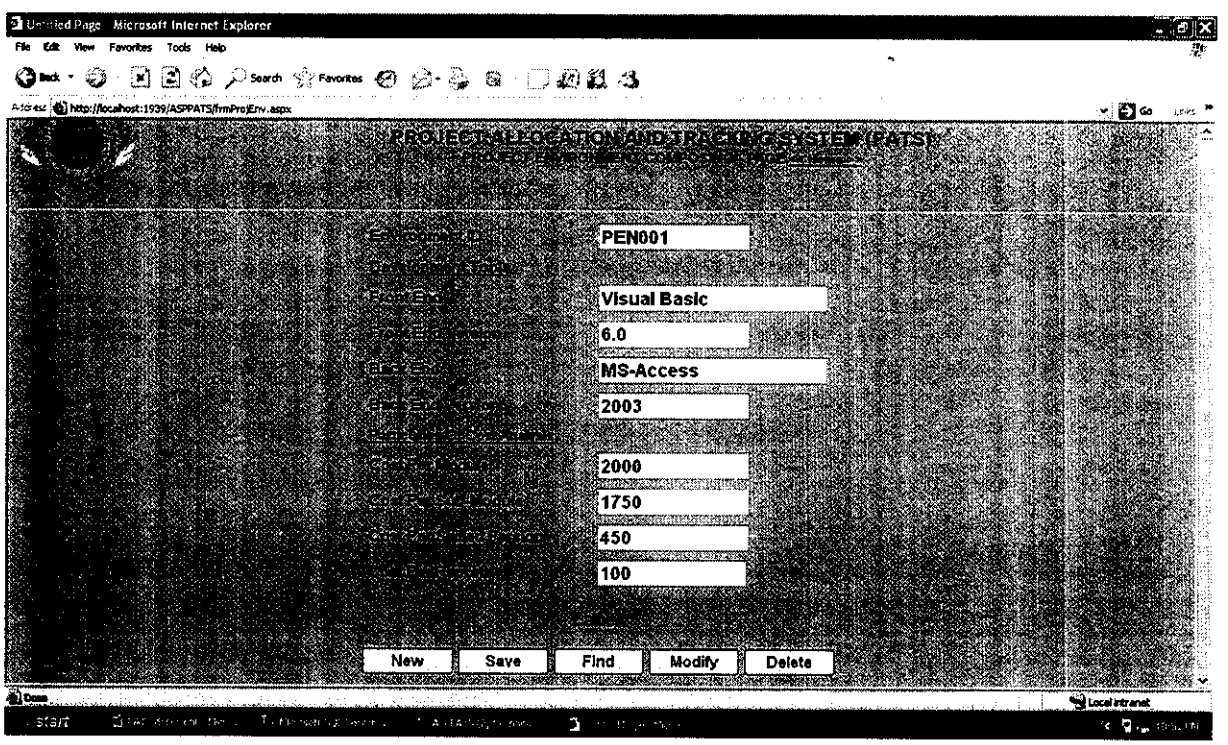
NEW USER CREATION



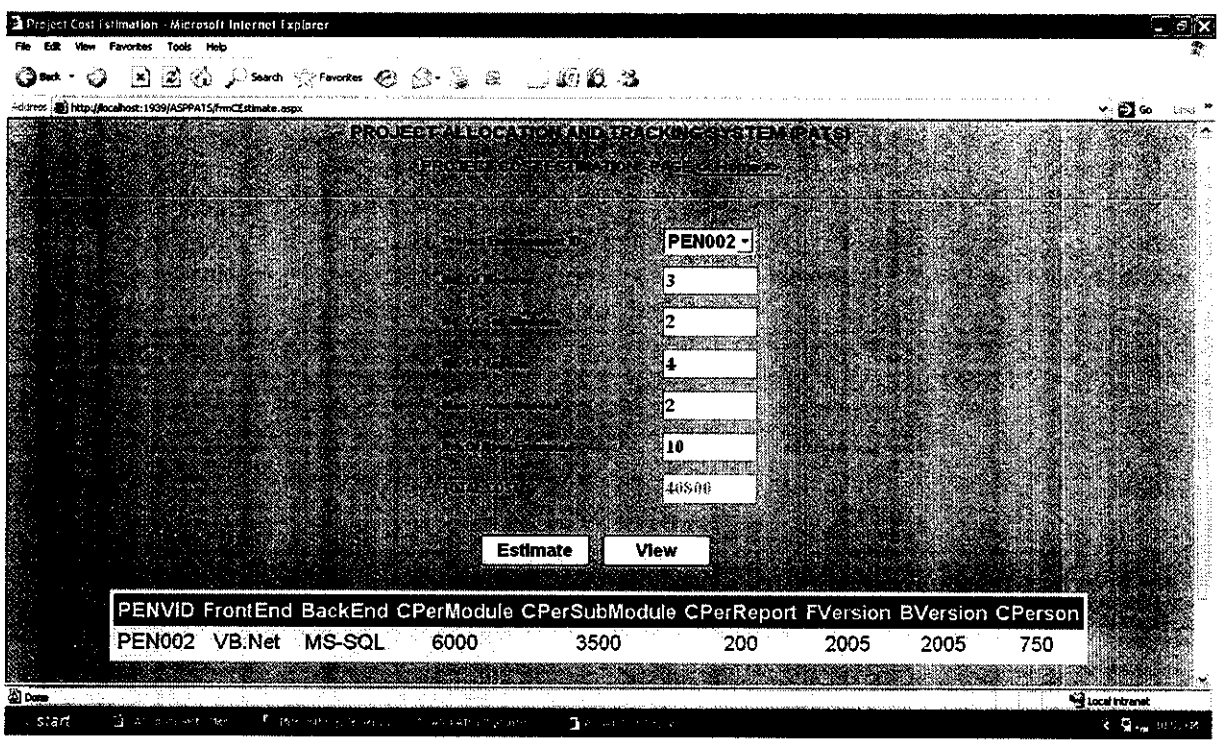
EMPLOYEE INFORMATION



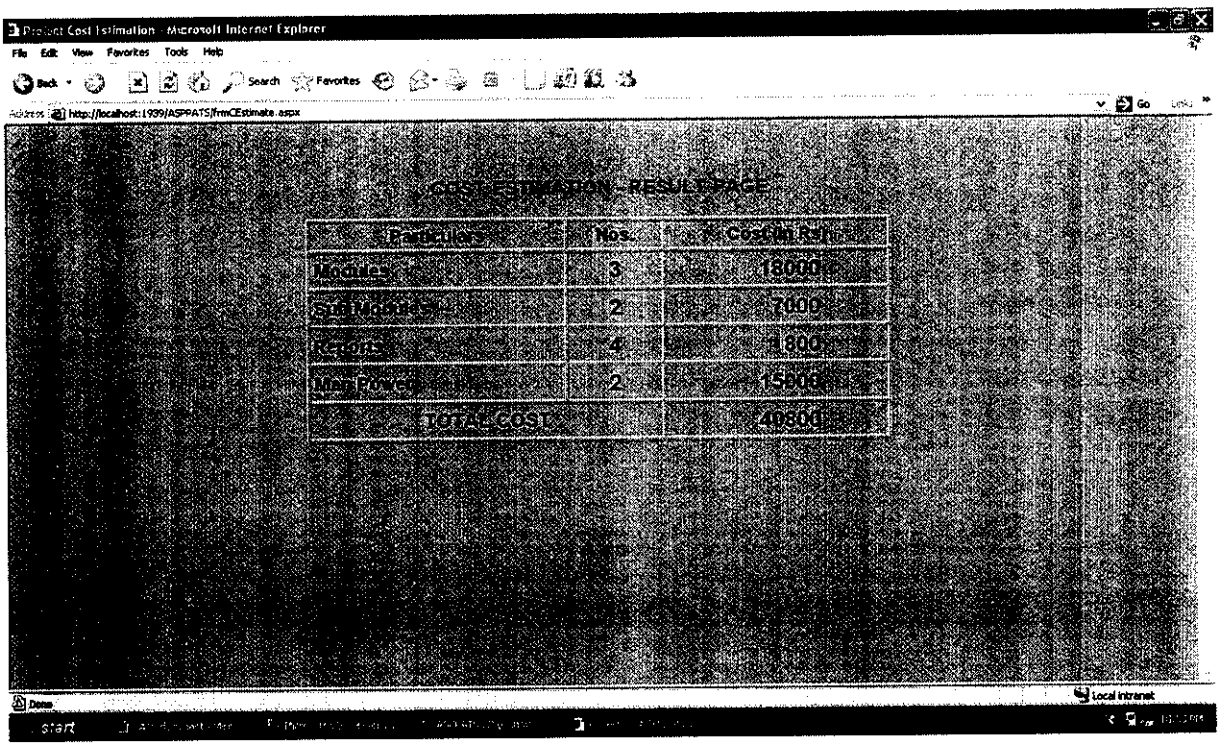
PROJECT ENVIRONMENT



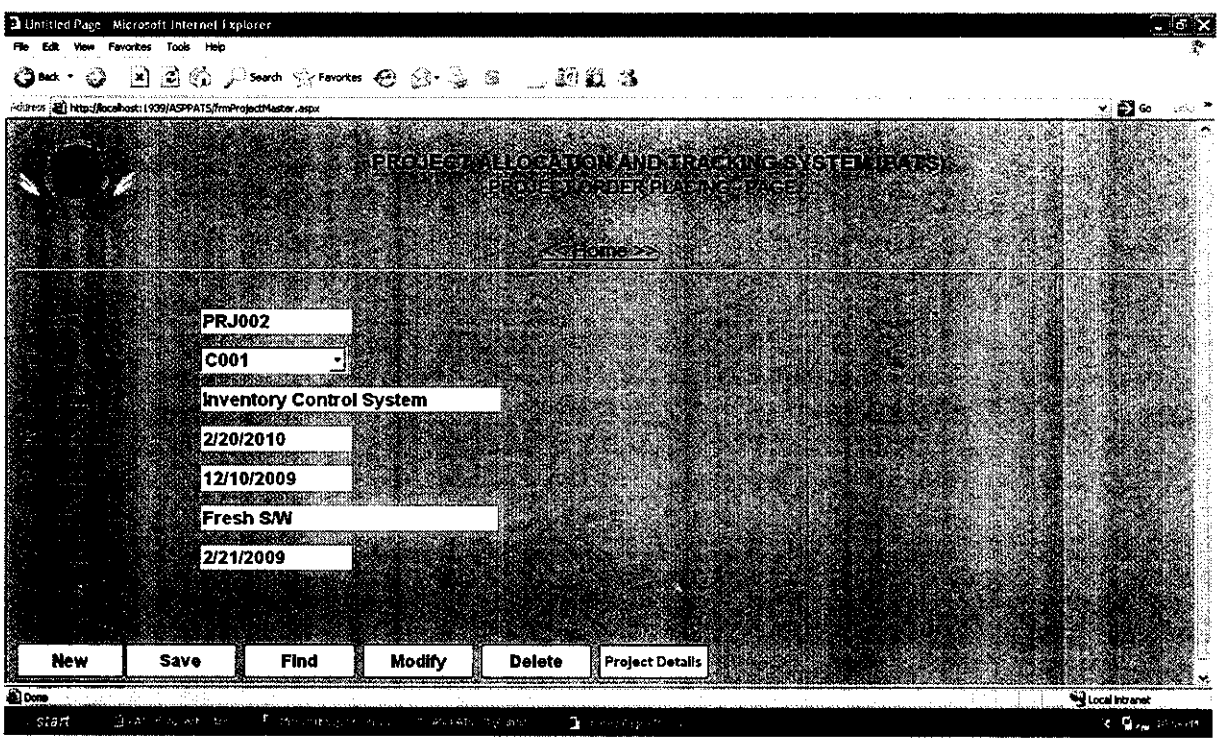
PROJECT COST ESTIMATIMATION



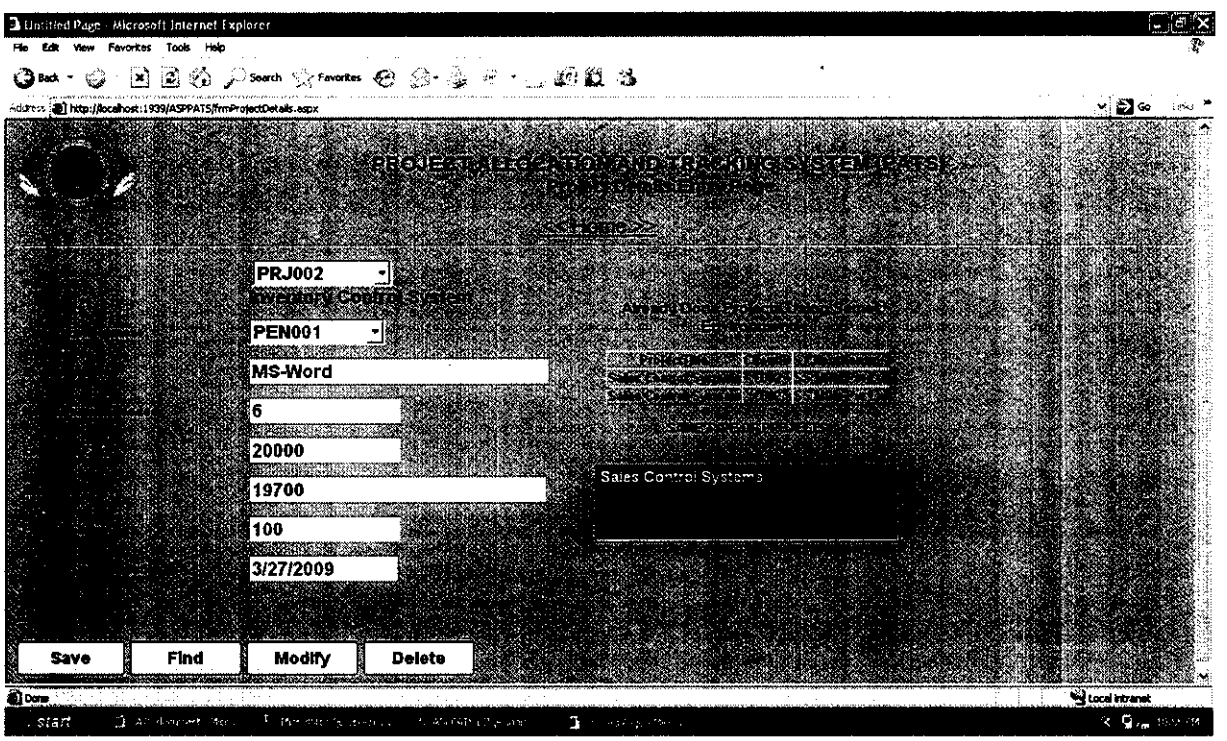
COST ESTIMATION RESULT PAGE



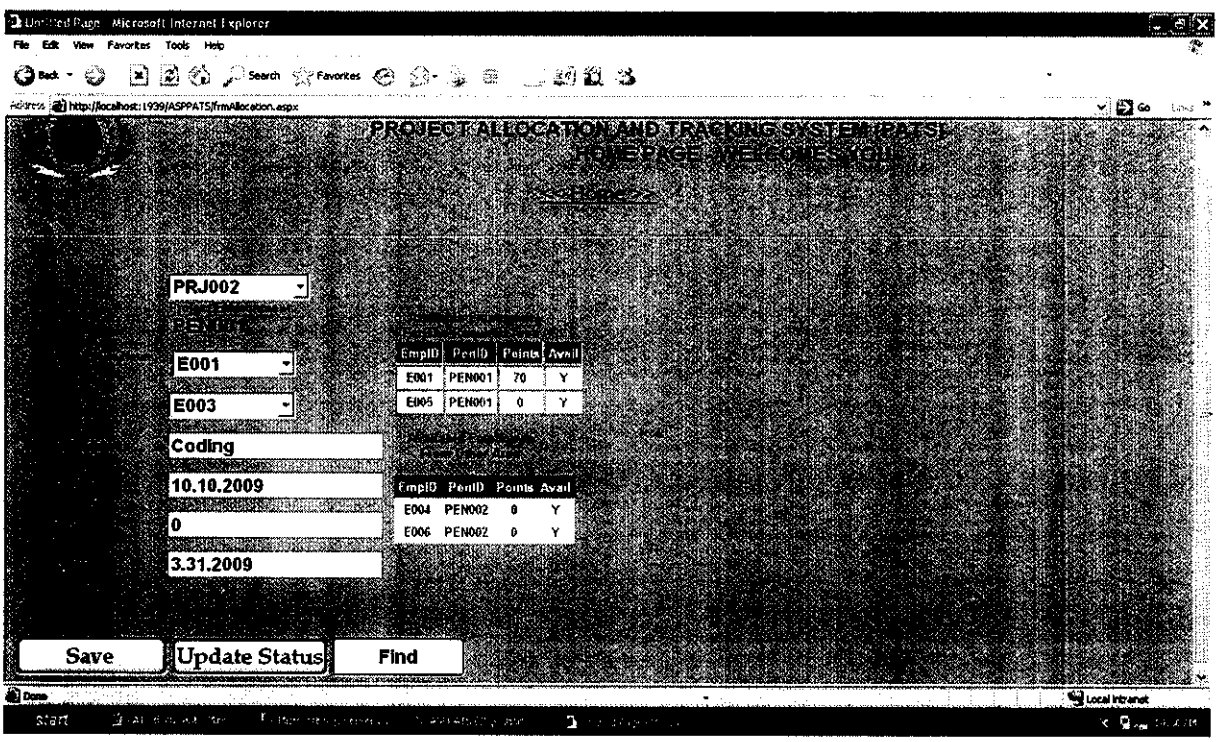
PROJECT ORDER PLACING



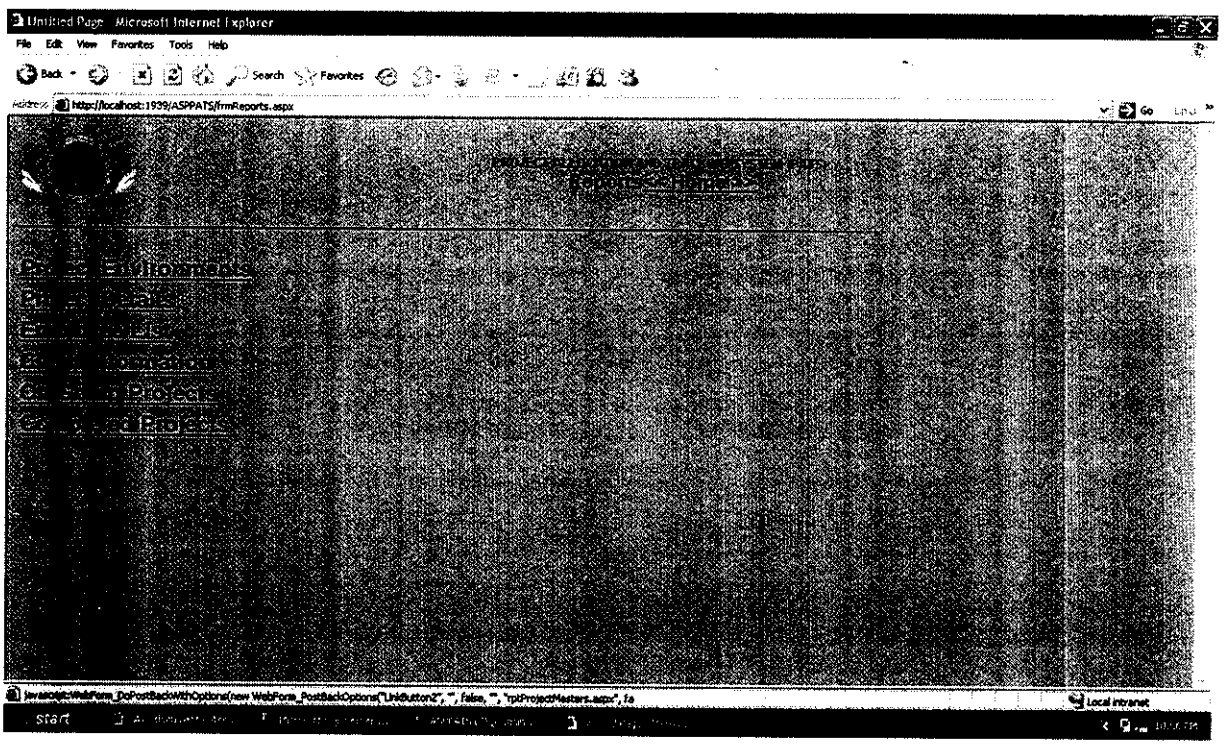
PROJECT DETALS ENTRY PAGE



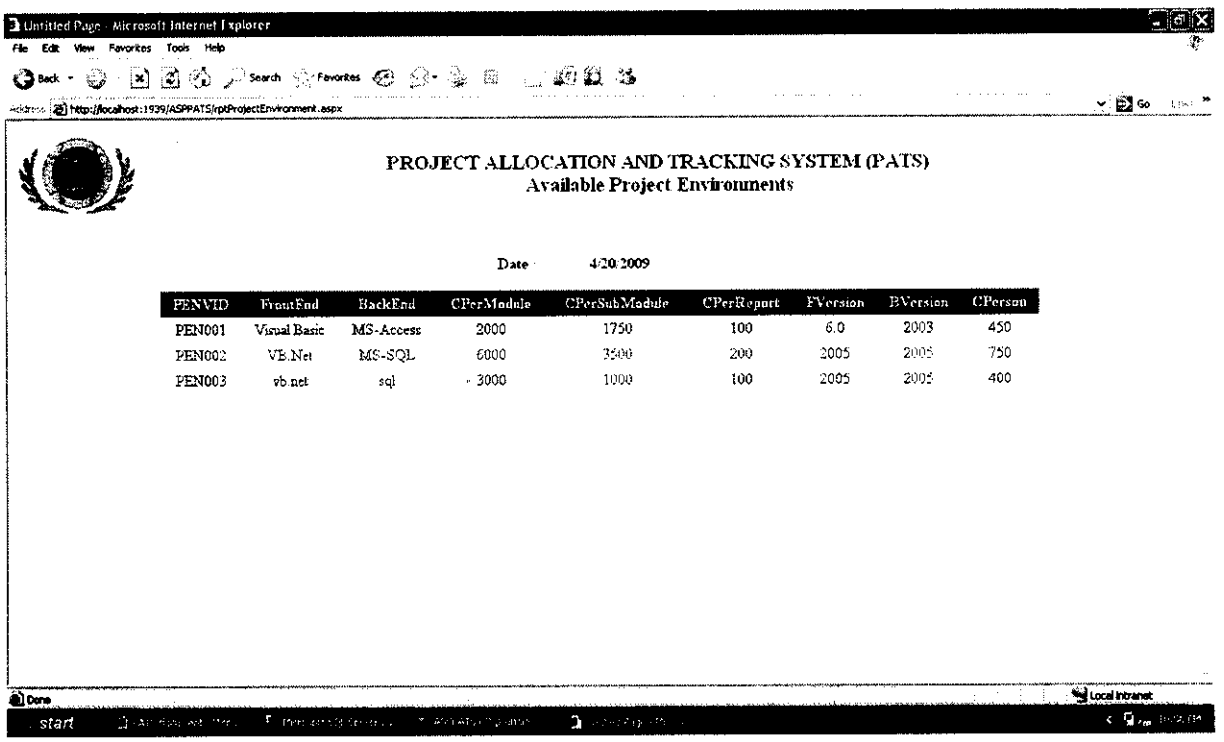
ALLOCATION PAGE



REPORTS


Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
http://localhost:1939/ASPPATS/frmReports.aspx
Go
Local intranet
Start

REPORTS FOR AVAILABLE PROJECT ENVIRONMENTS

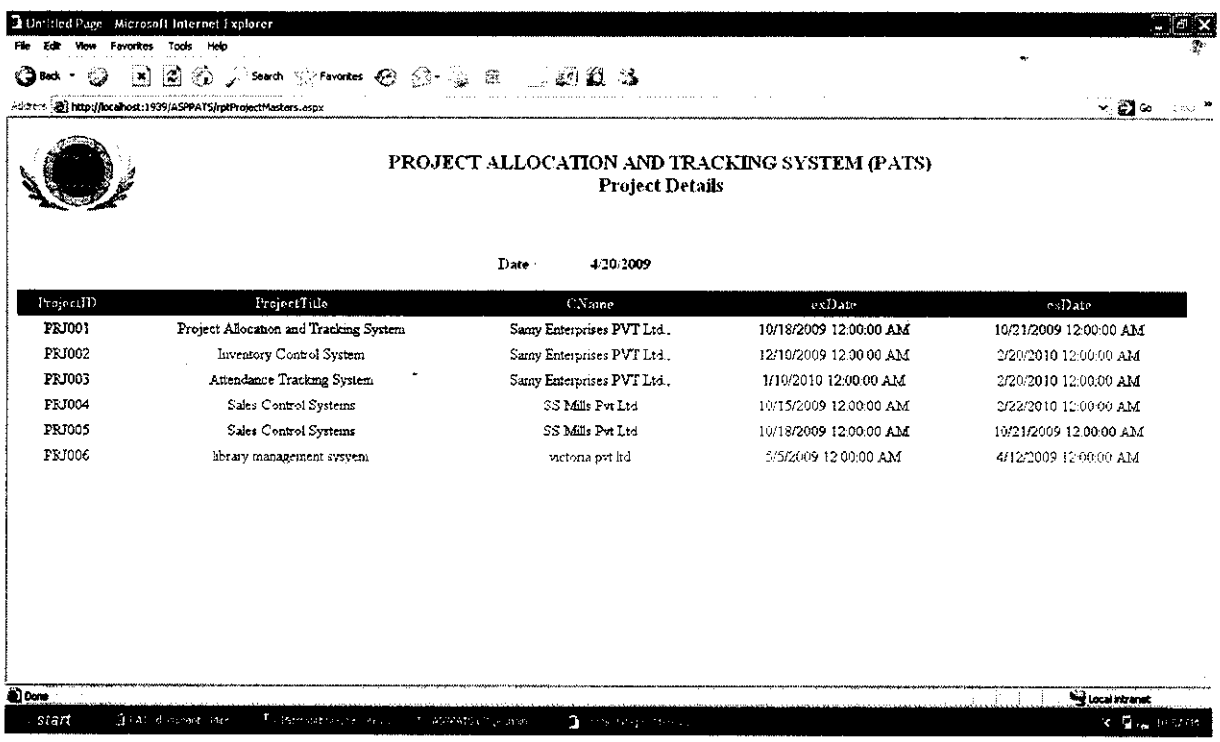


PROJECT ALLOCATION AND TRACKING SYSTEM (PATS)
Available Project Environments

Date 4/20/2009

PENVID	FrontEnd	BackEnd	CPerModule	CPerSubModule	CPerReport	FVersion	BVersion	CPerson
PEN001	Visual Basic	MS-Access	2000	1750	100	6.0	2003	450
PEN002	VB.Net	MS-SQL	6000	3500	200	2005	2005	750
PEN003	vb.net	sql	3000	1000	100	2005	2005	400

REPORT FOR PROJECT DETAILS

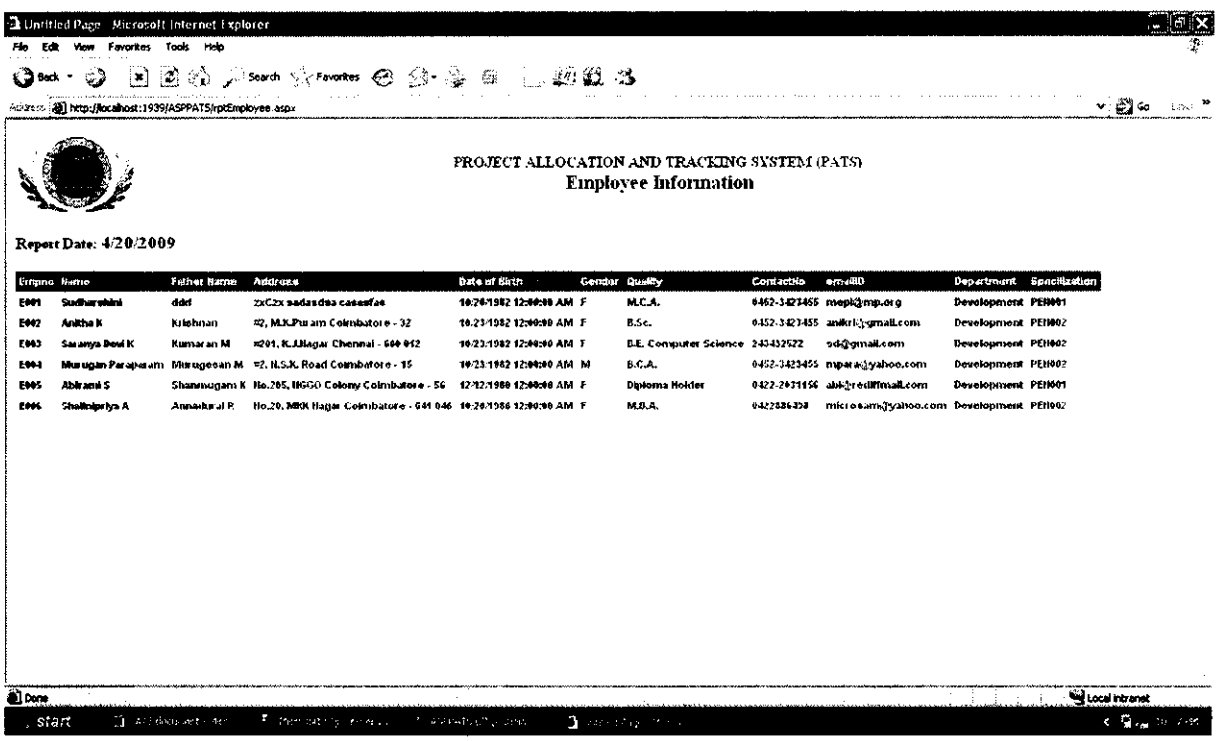


PROJECT ALLOCATION AND TRACKING SYSTEM (PATS)
Project Details

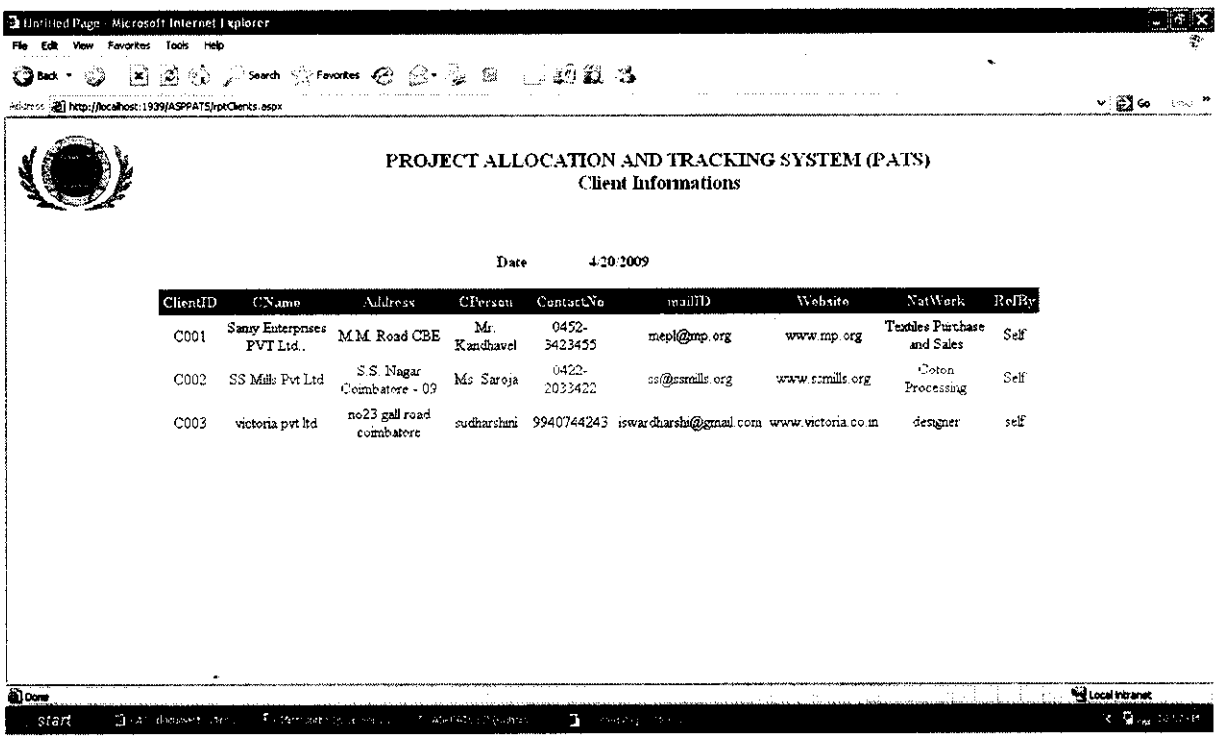
Date : 4/20/2009

ProjectID	ProjectTitle	CName	exDate	esDate
PRJ001	Project Allocation and Tracking System	Samy Enterprises PVT Ltd.,	10/18/2009 12:00:00 AM	10/21/2009 12:00:00 AM
PRJ002	Inventory Control System	Samy Enterprises PVT Ltd.,	12/10/2009 12:00:00 AM	2/20/2010 12:00:00 AM
PRJ003	Attendance Tracking System	Samy Enterprises PVT Ltd.,	1/10/2010 12:00:00 AM	2/20/2010 12:00:00 AM
PRJ004	Sales Control Systems	SS Mills Pvt Ltd	10/15/2009 12:00:00 AM	2/22/2010 12:00:00 AM
PRJ005	Sales Control Systems	SS Mills Pvt Ltd	10/18/2009 12:00:00 AM	10/21/2009 12:00:00 AM
PRJ006	library management sysyem	victoria pvt ltd	5/5/2009 12:00:00 AM	4/12/2009 12:00:00 AM

REPORT FOR EMPLOYEE INFORMATION



REPORT FOR CLIENT INFORMATION

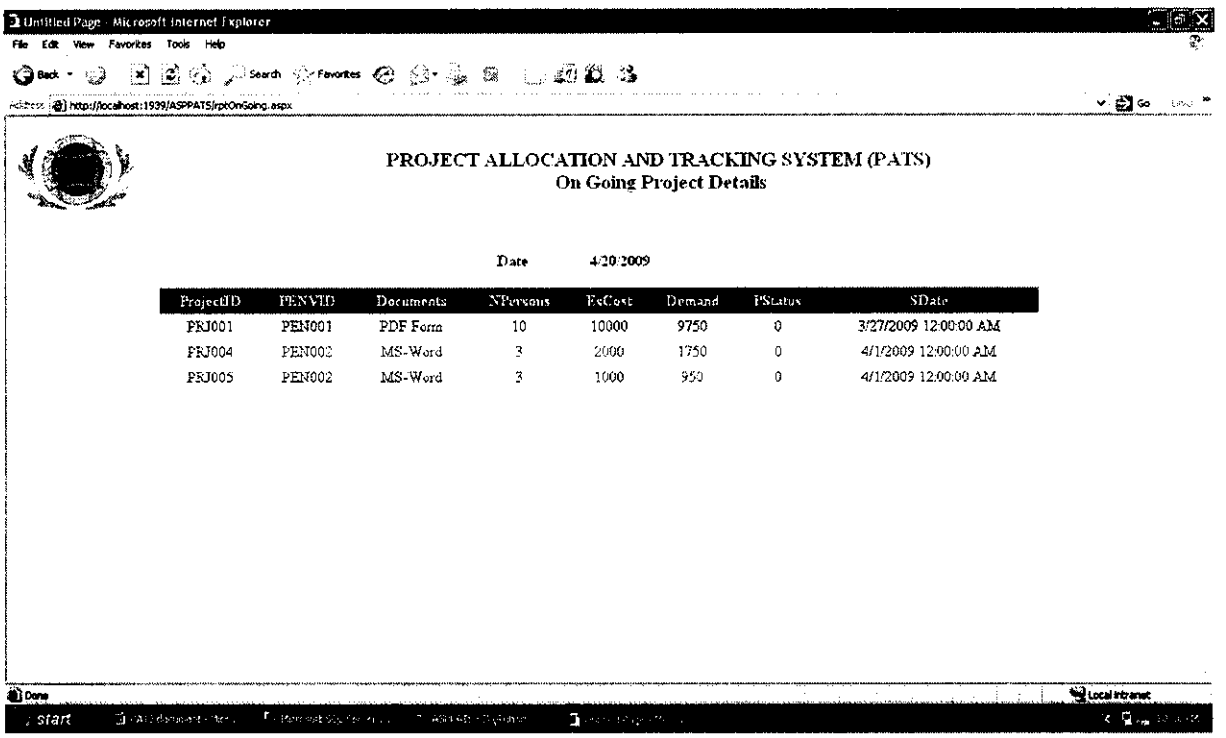


PROJECT ALLOCATION AND TRACKING SYSTEM (PATS)
Client Informations

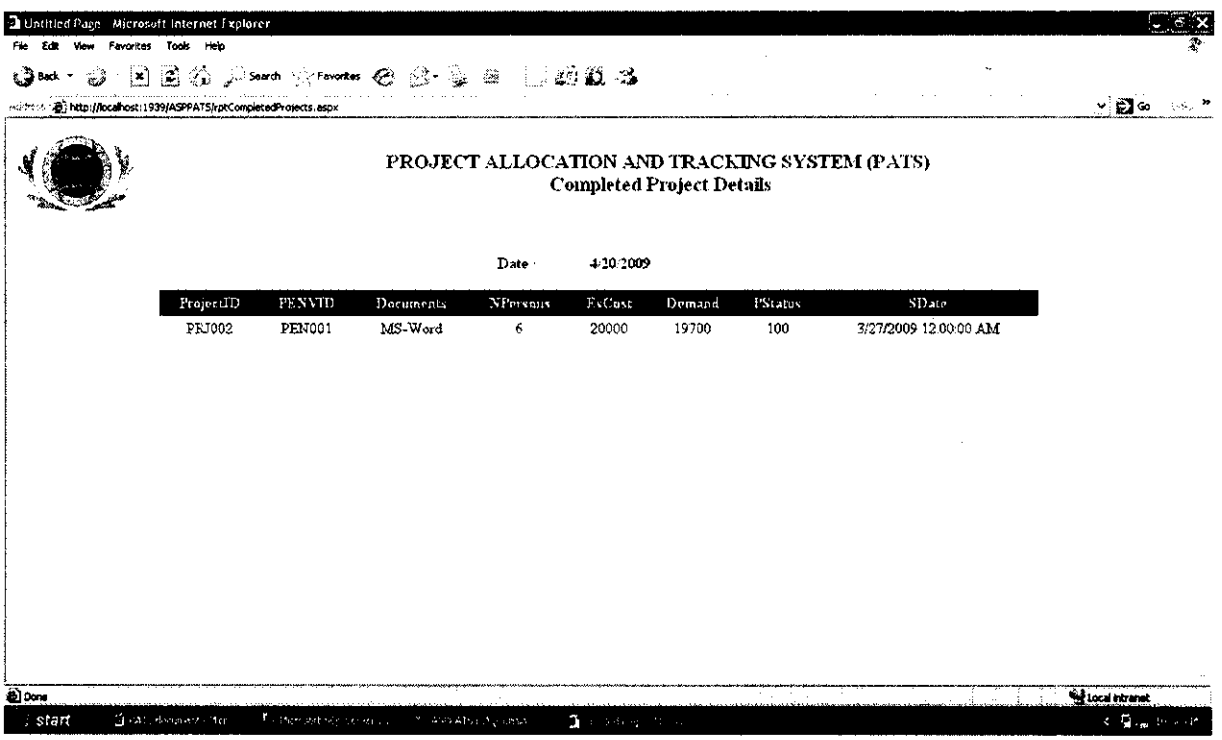
Date 4/20/2009

ClientID	CName	Address	CPerson	ContactNo	mailID	Website	NatWork	RefBy
C001	Sanny Enterprises PVT Ltd.,	M.M. Road CBE	Mr. Kandhavel	0452-3423455	mepl@mp.org	www.mp.org	Textiles Purchase and Sales	Self
C002	SS Mills Pvt Ltd	S.S. Nagar Coimbatore - 09	Ms. Saroja	0422-2033422	ss@ssmills.org	www.ssmills.org	Coton Processing	Self
C003	victoria pvt ltd	no23 gall road coimbatore	sudharshini	9940744243	iswardharshi@gmail.com	www.victoria.co.in	designer	self

REPORT FOR ON GOING PROJECT



REPORT FOR COMPLETED PROJECT



Untitled Page - Microsoft Internet Explorer

File Edit View Favorites Tools Help

http://localhost:1939/ASPPATS/rptCompletedProjects.aspx

PROJECT ALLOCATION AND TRACKING SYSTEM (PATS)
Completed Project Details

Date : 4/20/2009

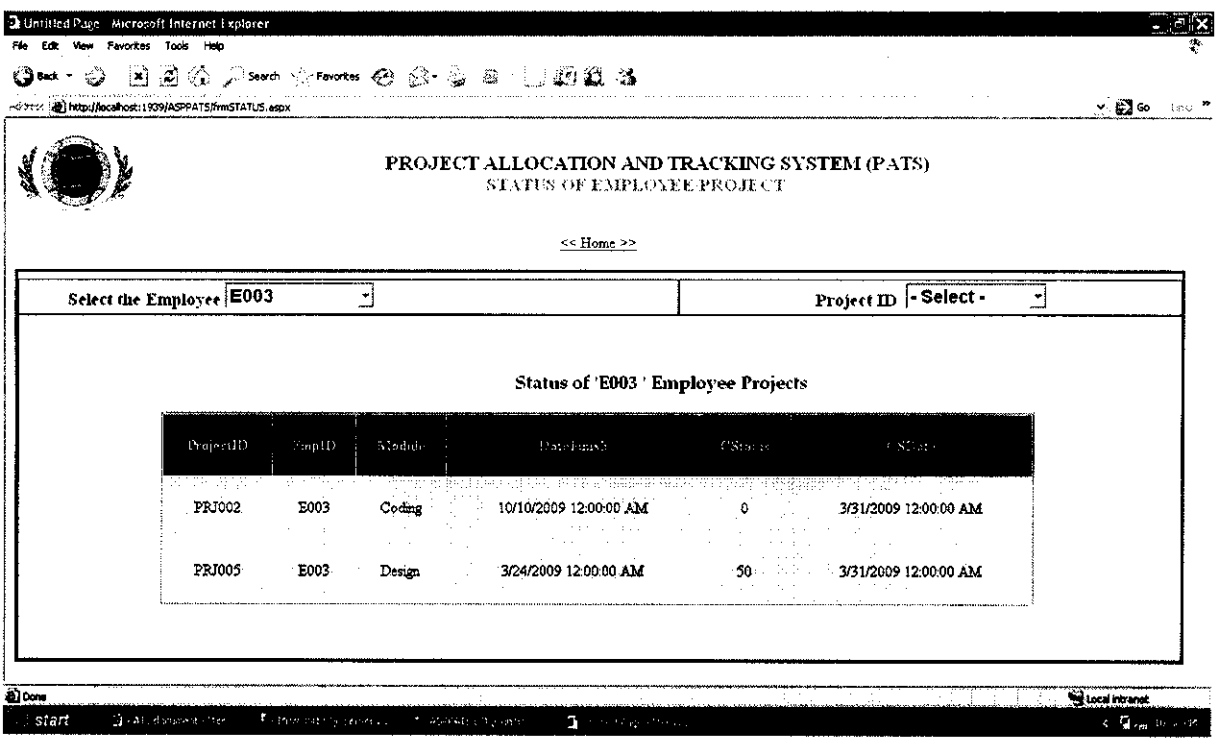
ProjectID	PENVID	Documents	NPersons	ExCost	Demand	PStatus	SDate
PRJ002	PEN001	MS-Word	6	20000	19700	100	3/27/2009 12:00:00 AM

Done

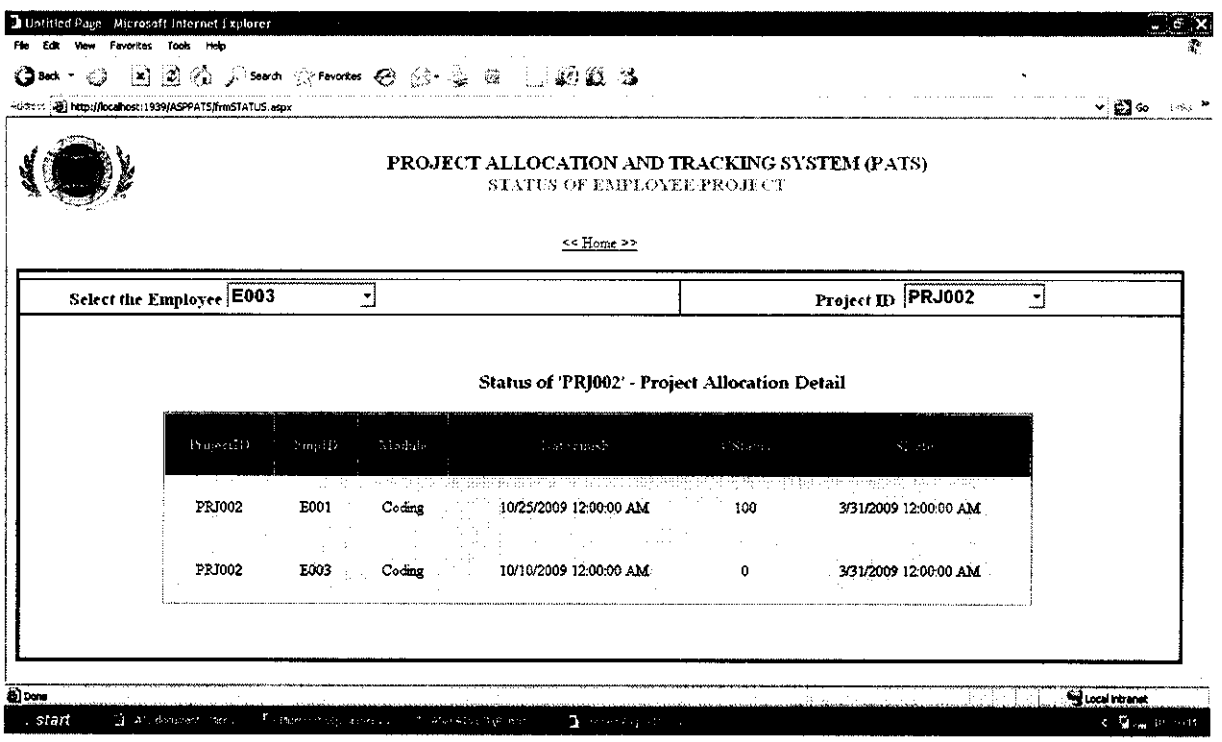
Local intranet

start

STATUS OF EMPLOYEE



STATUS OF PROJECT



REFERENCES

BOOKS

1. Kris Jamsa, "Visual Basic.Net".
2. Tony Martin, Dominic Selly, "Visual Basic.Net Projects", Tata McGraw Hill publications, 2003.
3. Gayle Coffman "SQL Server 7 the complete reference", Tata McGraw Hill Publications, 1999.

ONLINE REFERENCES

1. www.micrsoft.com
2. www.w3schools.com