**CHAPTER 1**

**INTRODUCTION**

# 1.1 Computer Applications and Importance

We know that this is advance era of computers. Computers are used in almost every sphere of human life. For example computers are used in the field of science, medical entertainment, space, weather, education etc.

The Computer was not much popular until the introduction of the Graphical User Interface (GUI) .GUI provides very interactive Interface between the user & Computer. The Computer had not been strange for end user after introduction of GUI. The rate of Development of Software increased & Computers interfered in almost every sphere of life. There are several Softwares, which are supposed to exist without GUI, but with GUI it produces several other facilities.

Information is crucial to the Management of any enterprise. The recent information explosion has greatly increased the information requirements of editing texts. Today’s users require flexible reporting tools to extract the information and they need it without depending on manual system.

Considering the present information scenario, the existing text editing system and other conventional methods of information storage and retrieval not only seem to be grossly inadequate but also time consuming and error prone. It seems very appropriate that computerization is the need of the hour.

**Literature Survey**

**a) Previous related work in chronological order**

In a given day, number of patients visits a hospital or a clinic. Many hospitals in India still manage the patient data manually. Hospitals will be able to save money and time if they have a good software program for managing patient’s data. The idea is to develop web based patient management software that can be used to keep track of the patients registering in a hospital or clinic. Doctors and the rooms available in a hospital can be managed using this system. Also, this system should support accessing the previous visit histories of any patient, search for patients by name etc.

A few points to be noted about the system we are developing here

* A patient can be categorized as “ In patient” or “Out Patient”. If patient type is "In Patient", a bed will be assigned to the patient.
* A doctor will be assigned to each patient before the patient meets the doctor. Only one doctor can be assigned to a patient at a given time.
* A patient can visit the hospital any number of times.

**b) Client/Company profile**

# One of the leading gynecologists of the city, Dr. Uroos Fatima (Aligarh Hospital) in Kela Nagar Crossing has established the clinic and has gained a loyal clientele over the past few years and is also frequently visited by several celebrities, aspiring models and other honorable clients and international patients as well. They also plan on expanding their business further and providing services to several more patients owing to its success over the past few years. The efficiency, dedication, precision and compassion offered at the clinic ensure that the patient's well-being, comfort and needs are kept of top priority. The clinic is equipped with latest types of equipment and boasts highly advanced surgical instruments that help in undergoing meticulous surgeries or procedures.

# 1.2 Present State of art and its shortcomings

The Aligarh Hospital started with few a workers and less number of patient comes there for the checkup, but as the time goes by today it has been joined by workers and the number of Patients increased drastically over the period of time hence information management of patient and workers has become a tedious task as everything is done on registers and manual files.

# 1.3 Realization of the Problem

In the previous system all the work has been registered on file manually and there are many other problems such as:

* Cannot Upload and Download the latest updates.
* No use of Web Services and Remoting.
* Risk of mismanagement and of data when the project is under development.
* Less Security.
* No proper coordination between different Applications and Users.
* Fewer Users - Friendly.

# 1.4 Introduction of the Problem

Hospital administrators are often inundated with information about a large number of patients and their visits to the hospital that need to be organized and kept up-to-date. The patient management system is a web based application that is designed and developed for hospital administrators and doctors to organize information on patient visits. The system intends to facilitate several steps in the process from the patient registration and to the patient evaluation. During this process, there will be many tasks that have to be handled by this system including maintaining complete information. The main objective of the system is to provide the administration staff and doctors with an easily maintainable information system for patient registration, visit scheduling and patient tracking with latest information.

**CHAPTER 2**

**PROBLEM FORMULATION**

# 2.1 Problem Definition detailed description

* The Aligarh Hospital started with few a workers and less number of patient comes there for the checkup, but as the time goes by today it has been joined by workers and the number of Patients increased drastically over the period of time hence information management of patient and workers has become a tedious task.
* Keeping track of Patients has become cumbersome with time and hence Report generation for patients becomes a difficult task.

# 2.2 Various aspects of problem

* With increase in patients information management has become a tedious task.
* Keeping track of Patients has become cumbersome with time and hence Report generation for patients becomes a difficult task.
* Keeping a track of doctors, workers, fetching and updating their records becomes difficult with the enlargement in Aligarh Hospital family.

# 2.3 Present System critical view

* Everything is being done manually, from record maintenance to report generation. Updating of records become very tedious as in some cases updating is needed at many other location of other files as well.
* Fetching details is also a very time taking task, as it has to be done sequentially, hence it wastes time.
* Maintaining the records of patients and their reports, test details etc also is done manually, hence generating a receipt of payment is to be done manually as well.
* Report generation is a very tedious task as it needs a lot of hard work by fetching data manually into the system in MS Excel.

**2.4 Scientific Novelty and need of work**

* The Aligarh Hospital was started with a small staff but today it has been joined by many humble people and hence information management has become a tedious task, so there is a need of computer based information management system.
* Keeping track of Patients has become cumbersome with time and hence Report generation for patients becomes a difficult task, hence a software which can fulfill this task with ease is a must have.
* Managing records on files and other database doesn’t give a very user friendly touch and only such records can be managed by two trained personnel, hence a user friendly software system is required so that in case of unavailability of those two resources, other people may also work on it.

**2.5 Proposed System/method of solution**

The development of the new system contains the following activities, which try to automate the entire process keeping in view of the database integration approach.

* User friendliness is provided in the application with various controls.
* The system makes the overall project management much easier and flexible.
* Readily upload the latest updates, allows user to download the alerts by clicking the URL.
* There is no risk of data mismanagement at any level while the project development is under process.
* It provides high level of security with different level of authentication.
* Updating is very much user friendly and easy.
* It reduces data redundancy.
* Saves time and efforts.
* Generates reports in one click.
* Paper work is drastically reduced.

**CHAPTER 3**

**SYSTEM ANALYSIS & DESIGN**

**3.1 Software Development tools**

**Visual Studio-2010**

Microsoft Visual Studio is an Integrated Development Environment (IDE) from Microsoft. It can be used to develop console and graphical user interface applications along with Windows Forms applications, web sites, web applications, and web services in both native code together with managed code for all platforms supported by Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework and Microsoft Silver-light.

Visual Studio supports languages by means of language services, which allow any programming language to be supported (to varying degrees) by the code editor and debugger, provided a language-specific service has been authored. Built-in languages include C/C++ (via Visual C++), VB.NET (via Visual Basic .NET), and C# (via Visual C#). Support for other languages such as Chrome, F#, Python, and Ruby among others has been made available via language services which are to be installed separately. It also supports XML/XSLT, HTML/XHTML, JavaScript and CSS. Language-specific versions of Visual Studio also exist which provide more limited language services to the user. These individual packages are called Microsoft Visual Basic, Visual J#, Visual C#, and Visual C++.

**Features**

* **Code editor**

Visual Studio, like any other IDE, includes a code editor that supports syntax highlighting and code completion using IntelliSense for not only variables, functions and methods but also language constructs like loops and queries. IntelliSense is supported for the included languages, as well as for XML and for Cascading Style Sheets and JavaScript when developing web sites and web applications.

* **Debugger**

Visual Studio includes a debugger that works both as a source-level debugger as well as machine-level debugger. It works with both managed code as well as native code and can be used for debugging applications written in any language supported by Visual Studio.

* **Designer**

**o Web designer**

Visual Studio also includes a web site editor and designer that allow web pages to be authored by dragging and dropping widgets. It is used for developing ASP.NET applications and supports HTML, CSS and JavaScript.

* **Class designer**

The Class Designer is used to author and edit the classes (including its members and their access) using UML modeling. The Class Designer can generate C# and VB.NET code outlines for the classes and methods. It can also generate class diagrams from hand-written classes.

* **Data designer**

The data designer can be used to graphically edit database schemas, including typed tables, primary and foreign keys and constraints. It can also be used to design queries from the graphical view.

* **Mapping designer**

The mapping designer is used by LINQ to SQL to design the mapping between database schemas and classes that encapsulate the data.

# 3.2 Information Collection

Techniques for collecting information are:

***3.2-a Interviewing***

***3.2-b Questionnaire***

***3.2-c Observation***

These techniques are not mutually exclusive indeed in practice more than one technique will be employed to establish the facts. During the course of an interview records may be inspected and a questionnaire may be completed.

**3.2-a INTERVIEWING**

The interview is a fact-to-fact exchange of information. It is a communication channel between the analyst and the organization. Interviewing is used to gain in support or understanding from the user for a new idea or method. The interview provides an excellent opportunity for the analyst to establish report with user personnel.

After the interview

* Record the data at the earliest opportunity.
* Check the facts as soon as possible.
* Get agreements to the facts.

**3.2-b QUESTIONNAIRE**

The questionnaire can be used best as a fact finding tool when the recipient is physically removed from the analyst and travel is prohibited for person, where there are many potential recipient and when the information is intended to verify similar information gathered from other sources.

**3.2-c OBSERVATION**

Observation can bring in missed facts, new ways to improve the existing procedures, duplicate work done inadvertently, etc, can bring in what other fact finding methods cannot! But this task is delicate because people do not like to be observed when they work.

It is not the quantity of time observed is important but the unorthodox angles of observation of the work content and methods are going to be rewarding.

Observation gives analyst the opportunity to go behind the scenes in a BCA Information System to learn inside story to discover how things work in new areas of information.

For this project, interview technique is used, interview is taken of the administrator (who is going to be the primary end user of the system, 1 local admin), managing team and doctor.

Two sets of questions have been prepared:

1. **For Administrators**

* What is being done in the organization?
* How it is being done?
* How much data is updated roughly on monthly basis?
* What are the problems that may arise?
* If a problem arises, how will it be solved?
* What could cause such a problem?

CONCLUSION:

The organization serves as an Hospital and everything is maintained on registers, hence a low budget computer based system is required by them to cater their needs like adding new patients, updating existing ones, for searching information about them etc.

1. **For Managing Team and their admin**

* How often new doctors joins?
* What problems arise in delegating tasks manually?
* What problems arise in recording reports manually?
* What must be done for proper management of patients?
* What can be done for better record keeping?

CONCLUSION:

Feeding the records of patient manually becomes tedious job, also managing test records becomes cumbersome, hence they need a computer based system where all records can be maintained, and previous records can be seen from history.

**3.3 Requirement Specification**

Requirements specification is a software engineering task that bridges the gap between the system level requirements engineering and software design. Requirements engineering activities result in the specification of software’s optional characteristics (function, data and behavior), indicate software’s interface with other system elements, and establish constraints that software must meet. The most commonly used requirements technique is to conduct a meeting or interview. The first meeting between a software engineer (the analyst) and the employee can be linked to the awkwardness of a first date between two adolescents. Neither person knows what to say or ask, both are worried that they do say will be misinterpreted, both are thinking about where it might lead (both likely have radically different expectations here), both want to get the thing over with, but at the same time, both want to be a success.

The goal of the requirements gathering activity is to collect all relevant information from the employee regarding the product to be developed with a view to clearly understanding the employee requirements and weeding out the incompleteness and inconsistencies in these requirements. The requirements analysis activity is begun by collecting all relevant data regarding the product to be developed from the users of the product and from the employee through interviews and discussions.

**3.4 Analysis and Development of Actual Solution**

The analysis phase is the detailed understanding of all important facts of the business area under investigation. The relationships of the various system components among themselves and with environment are studied and understood. This requires data collection from a variety of sources.

He/She has to understand about the whole details about the problem. He should also identify the reasons for the problems that have occurred and the preventive measures to avoid them.

The analyst must have a detailed study of the manuals and reports about the organization. Further he should have a direct observation of the activities in the organization and collect a sample of the forms and documents to understand the whole system.

**3.4.1 General Objective**

The purpose of this proposal is to design and develop Patient Montitoring system for the Aligarh Hospital.

**3.4.2 Specific Objectives**

*Patient Monitoring* Systemis an easy to use, web based application that enables organization to improve all aspects of task execution and success. To improve the quality of life of the underprivileged through care services, rehabilitation which is provided by the Aligarh Hospital, hence a requirement of an information management system to speed up their work is indeed a must have thing.

* This project ensures the organization to keep track of patients, doctors, medicine recommended , expenses and much more.
* The burden of manual work is reduced because of the computerization of the system.
* The system is user friendly.
* Updating is very much user friendly and easy.
* It reduces data redundancy.

**3.5 Module Description**

**Administrator:-**

In this module Administrator will have complete control of the system. She/he can Add/Edit/Delete patients, Add/Edit/Delete Doctors, Add/Edit/Delete Beds, Search for patients, Assign patients to doctors. He can search all the info about the Admitted Patient, Discharged Patient, Doctors, Medicine, Test, and Room.

**Employee:-**

This is Module is for employees who are working in that particular hospital. Admin will assign them user name and password by this they can enter in to their related page. An employee can enter the information about the Admitted Patient, he can add all type of charges like Room, Medicine, Test etc to particular Patient, and he can also maintain the information of the Patient who has discharged.

**Doctor:-**

Doctors who are related to that hospital can enter to their related page by their login name and password. They can see their information, change their account, they can see patients whom they checked, and Doctor can access a patient’s record and update his observations about the patient in that particular visit

**Reports:-**

This module contains all the information about the reports generated by the admin of the patient admitted, discharged, medicine charges, room charges, test charges for a particular patient and can be seen in the form of In-Patient and Out-Patient only by the employee.

**Authentication:-**

This module contains all the information about the authenticated user. User without his username and password can’t enter into the login if he is only the authenticated user then he can enter to his login.

**3.6 Choice of language**

***ASP.NET (As a Front-End tool)***

ASP.NET is the hosting environment that enables developers to use the .NET Framework to target Web-based applications. However, ASP.NET is more than just a runtime host; it is a complete architecture for developing Web sites and Internet-distributed objects Framework. using managed code. Both Web Forms and XML Web services use IIS and ASP.NET as the publishing mechanism for applications, and both have a collection of supporting classes in the .NET

Some features of ASP.NET

* Full support for object oriented programming.
* Structured error handling capabilities.
* Access to .NET Framework.
* Powerful unified Integrated Development Environment (IDE).
* Inherent support for XML & Web Services.
* New Web capabilities with Web Forms.
* Immense power of tools & controls (including Server Controls).

***Microsoft SQL Server (As a back end tool):***

* Microsoft SQL Server, a relational database server from Microsoft.
* It eliminates all parent-child relationships and instead represented all data in the database as simple row/column tables of data values.
* Flexibility in data modeling.
* Easy for accessing data.
* Each table is an independent entity and there is no physical relationship between tables.
* Reduce data storage and redundancy.
* Independent of physical storage and logical data design.
* It has wild level data manipulation language (SQL).
* Deferred writing at commits to improve transaction performance.

**3.7 Choice of system for implementation**

System Configuration required at client’s side must match below

**Operating System** : Windows 2000 and above

**CPU type** : Intel dual core (or higher versions)

**Micro processor** : 500 MHz or above

**RAM** : 256 MB and above

**Hard disk** : 10 GB and above

**Printer**  : DeskJet, Laser

**Machine type** : Desktop pc/Laptop

**CHAPTER 4**

**SYSTEM IMPLEMENTATION**

**4.1 Minimum Hardware Requirements**

Minimum hardware Configuration required at client’s side:

**CPU type** : Intel dual core (or higher versions)

**Micro processor** : 500 MHz or above

**CD- Drive** : CD-ROM

**RAM** : 256 MB and above

**Hard disk** : 10 GB and above

**Printer**  : DeskJet, Laser

* 1. **Software Requirements**

**Development Environment** : Visual Studio 2010

**Framework** : .NET 4.1

**Editor** : VS

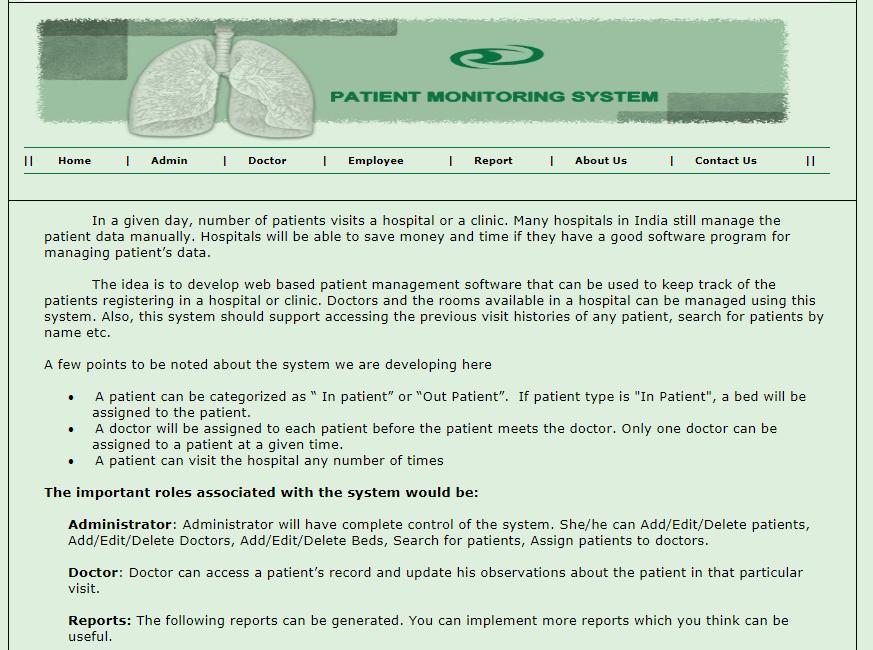
**Operating System** : Windows

* 1. **Input Requirements**

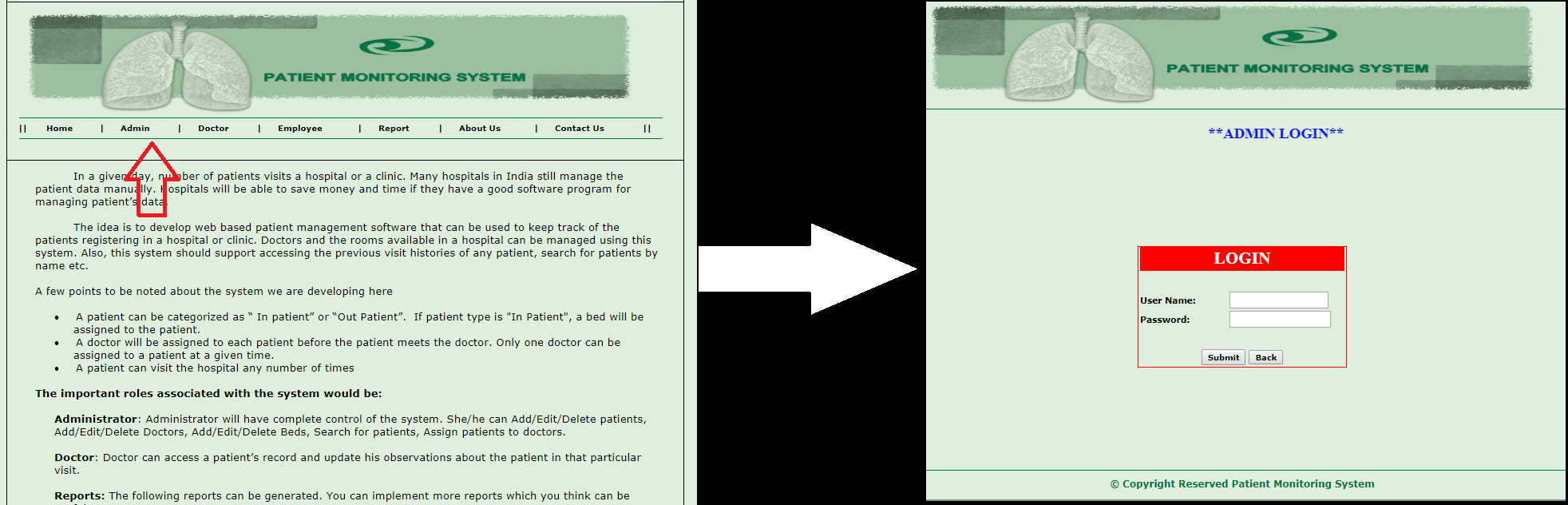
**Input requirements of the system are:**

* login passwords from administrator
* Employee’s personal information in case a new employee is being added
* Doctors' personal information in case a new doctor is being added
* Patient’s personal information for addition of new patient
* Updating data for all the above
  1. **Operation Manual**

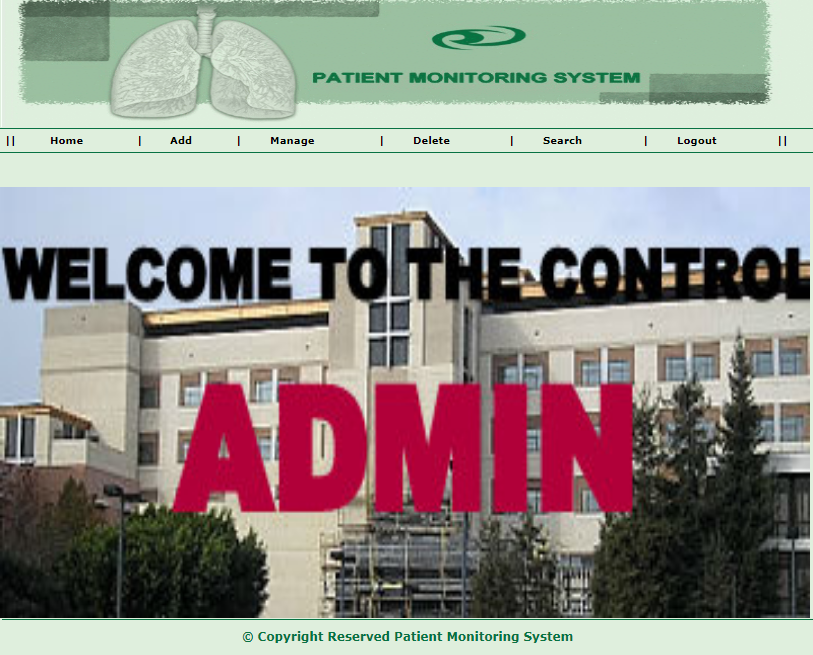
1. **User enters the website URL (provided after publishing the website), A Homepage is flashed as below**

****

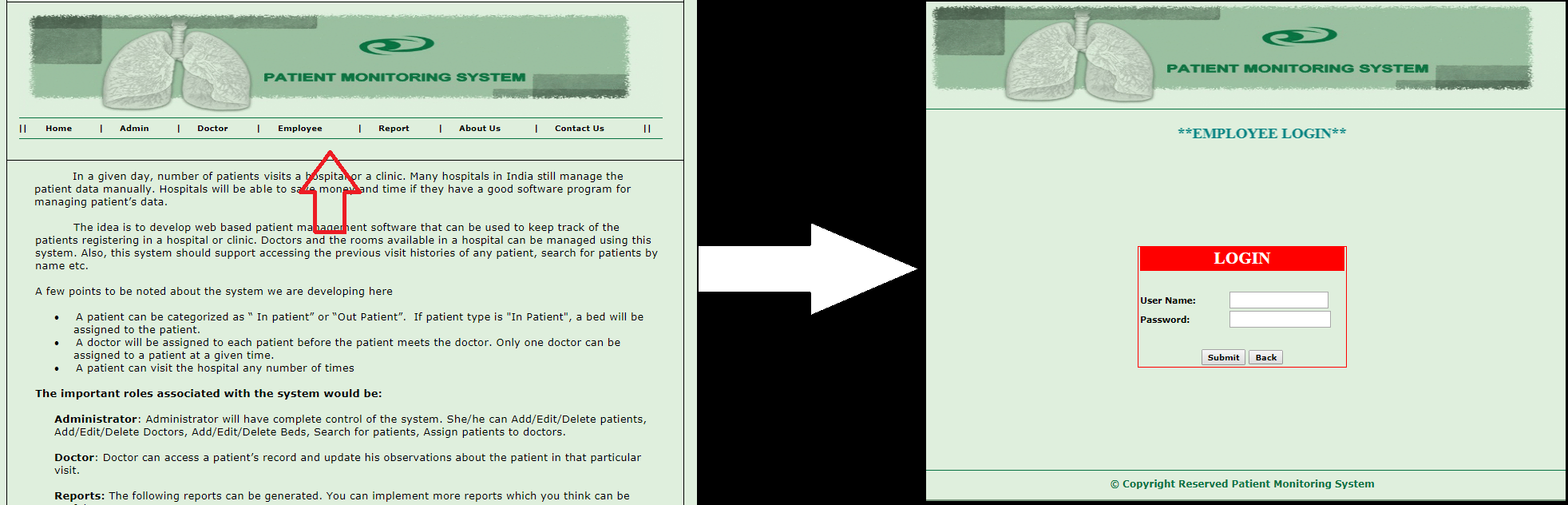
**2. If user is Local Admin, then it clicks on the admin section (Shown with red arrow) and then it leads to next page where Admin uses login credentials of local admin to login.**

****

**3. Once logged in local admin can do all task listed on the page.**

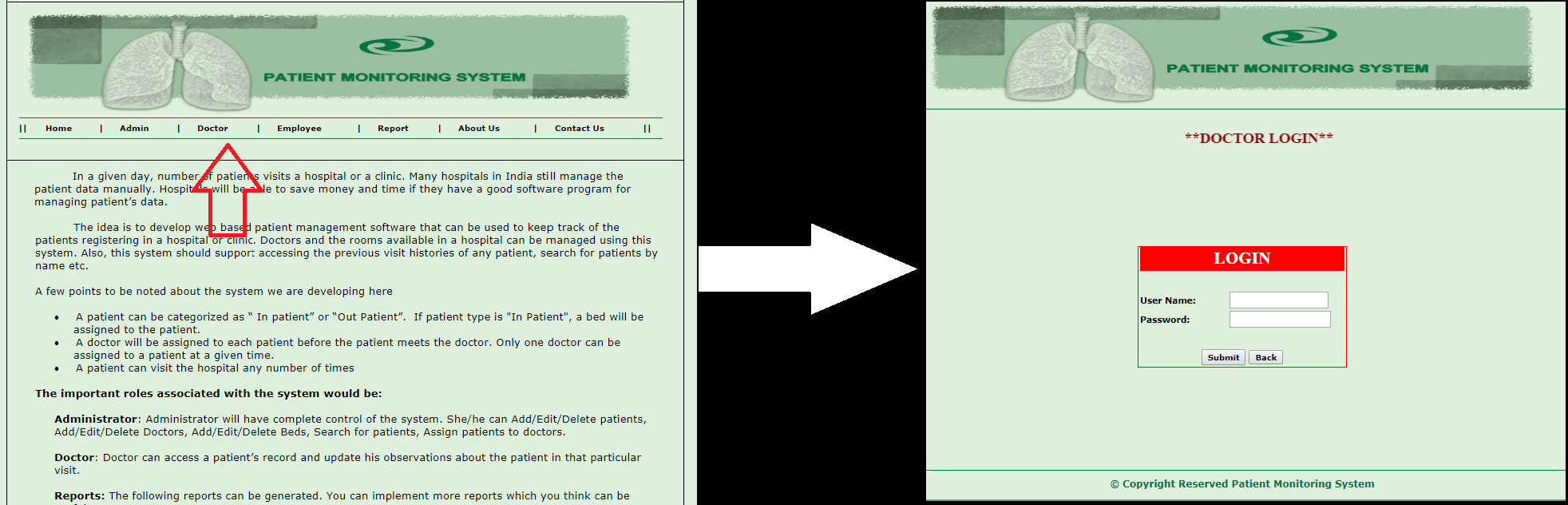
****

**4. Else if user is Employee it uses login credentials of employee to login as shown below:**

****

**5. Once logged in employee can do all task listed on the page.**

****

**6. Else if user is Doctor it uses login credentials of doctor to login as shown below:**

**6. Once logged in doctor can do all task listed on the page.**

****

**7. Else any other user can login and can view reports.**

* 1. **Output Reports to be obtained**

**Two types of reports are being fetched:**

1. **Report by In-Patient and Out-Patient**

Monthly, weekly, yearly reports can be fetched to see how many patient have been admitted in the hospital. This report can only be viewed to Admin and Employee.

1. **Personal Report**

Personal report of a patient can be fetched by providing login credentials at the home page in report section. The employee manages the report section.

**CHAPTER 5**

**RESULT, DISCUSSION AND CONCLUSION**

**5.1 Interpretations of result**

Results are in the form of reports of various types which can be fetched and printed to be viewed later.

**5.2 Application areas and scope of work**

The software can be used by any Hospital for managing their day to day tasks and generate various reports later.

**5.3 Advantages and special features of system**

* To improve the quality of life of the underprivileged through   
  care services, rehabilitation etc, which is provided by the Aligarh Hospital, hence a requirement of an information management system to speed up their work is indeed a must have thing.
* This project ensures the organization to keep track of patients , doctors , employees and much more.
* The system has been developed using .NET framework.So that all features are added in this system. At the time of handling this system, it allows multi-user environment facility.
* The burden of manual work is reduced because of the computerization of the system.
* The system is user friendly.
* Updating is very much user friendly and easy.
* It reduces data redundancy.

**5.4 Limitations**

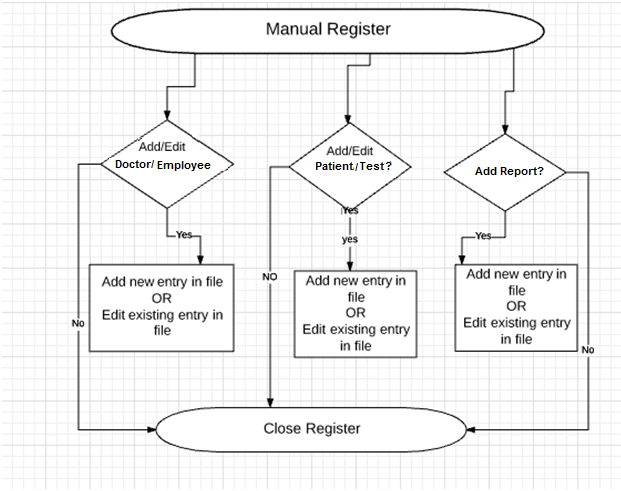
Online appointment is not available yet, due to which all the appointments have to be registered via telephonic conversation or by going to the hospital.

**5.5 Future extension**

As per the requirement the software can be enhanced with many other modules like online appointment, and some part of it can be made online in order to prioritize the appointment by paying some advance. It can allow people from different areas having serious illness to get checked up early.

**APPENDIX**

1. **Flow Chart**

****

1. **DFD/System Flow Diagram**

****

1. **Program Listing codes**

**Add Doctor on Patient BL**:

using System;

using System.Data;

using System.Configuration;

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;

using HospitalMgmt.DAL;

/// <summary>

/// Summary description for AddDoctorOnPatient

/// </summary>

public class AddDoctorOnPatient:Connection

{

public static DataSet ds;

public AddDoctorOnPatient()

{

//

// TODO: Add constructor logic here

//

}

string code, doccode, time,specialist;

public string Code

{

get { return code; }

set { code = value; }

}

public string Doccode

{

get { return doccode; }

set { doccode = value; }

}

public string Time

{

get { return time; }

set { time = value; }

}

public string Specialist

{

get { return specialist; }

set { specialist = value; }

}

int charge;

public int Charge

{

get { return charge; }

set { charge = value; }

}

DateTime date;

public DateTime Date

{

get { return date; }

set { date = value; }

}

public void InsertDoctorCharge()

{

SqlParameter[] p = new SqlParameter[6];

p[0] = new SqlParameter("@code", this.code);

p[0].DbType = DbType.String;

p[1] = new SqlParameter("@doccode", this.doccode);

p[1].DbType = DbType.String;

p[2] = new SqlParameter("@date", this.date);

p[2].DbType = DbType.DateTime;

p[3] = new SqlParameter("@time", this.time);

p[3].DbType = DbType.String;

p[4] = new SqlParameter("@charge", this.charge);

p[4].DbType = DbType.Int32;

p[5] = new SqlParameter("@specialist", this.specialist);

p[5].DbType = DbType.String;

SqlHelper.ExecuteNonQuery(con, CommandType.StoredProcedure, "SpInsertDoctorCharges", p);

}

public DataSet ShowDoctorOnPatient()

{

ds = new DataSet();

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowDoctorOnPatient");

return ds;

}

public DataSet ShowPatientInfoByDoctor()

{

ds = new DataSet();

SqlParameter[] p = new SqlParameter[1];

p[0] = new SqlParameter("@drcode", this.doccode);

p[0].DbType = DbType.String;

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowPatientInfoByDoctor",p);

return ds;

}

public DataSet ShowDoctorNameByCode()

{

ds = new DataSet();

SqlParameter[] p = new SqlParameter[1];

p[0] = new SqlParameter("@drcode", this.doccode);

p[0].DbType = DbType.String;

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowDoctorNameByCode", p);

return ds;

}

}

**Add Patient BL:**

using System;

using System.Data;

using System.Configuration;

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;

using HospitalMgmt.DAL;

/// <summary>

/// Summary description for AddPatientBL

/// </summary>

public class AddPatientBL:Connection

{

public static DataSet ds;

public AddPatientBL()

{

//

// TODO: Add constructor logic here

//

}

string code, name, hname, complaint, sex, address, country, state, city, iopatient, doctorcode, testcode, roomcode, condition, admittime;

public string Code

{

get { return code; }

set { code = value; }

}

public string Name

{

get { return name; }

set { name = value; }

}

public string Hname

{

get { return hname; }

set { hname = value; }

}

public string Complaint

{

get { return complaint; }

set { complaint = value; }

}

public string Sex

{

get { return sex; }

set { sex = value; }

}

public string Address

{

get { return address; }

set { address = value; }

}

public string Country

{

get { return country; }

set { country = value; }

}

public string State

{

get { return state; }

set { state = value; }

}

public string City

{

get { return city; }

set { city = value; }

}

public string Iopatient

{

get { return iopatient; }

set { iopatient = value; }

}

public string Doctorcode

{

get { return doctorcode; }

set { doctorcode = value; }

}

public string Testcode

{

get { return testcode; }

set { testcode = value; }

}

public string Roomcode

{

get { return roomcode; }

set { roomcode = value; }

}

public string Condition

{

get { return condition; }

set { condition = value; }

}

public string Admittime

{

get { return admittime; }

set { admittime = value; }

}

int age, advance;

public int Age

{

get { return age; }

set { age = value; }

}

public int Advance

{

get { return advance; }

set { advance = value; }

}

DateTime admitdate,date1,date2;

public DateTime Admitdate

{

get { return admitdate; }

set { admitdate = value; }

}

public DateTime Date1

{

get { return date1; }

set { date1 = value; }

}

public DateTime Date2

{

get { return date2; }

set { date2 = value; }

}

public void AddPatient()

{

SqlParameter[] p = new SqlParameter[18];

p[0] = new SqlParameter("@code", this.code);

p[0].DbType = DbType.String;

p[1] = new SqlParameter("@name", this.name);

p[1].DbType = DbType.String;

p[2] = new SqlParameter("@hname", this.hname);

p[2].DbType = DbType.String;

p[3] = new SqlParameter("@complaint", this.complaint);

p[3].DbType = DbType.String;

p[4] = new SqlParameter("@sex", this.sex);

p[4].DbType = DbType.String;

p[5] = new SqlParameter("@address", this.address);

p[5].DbType = DbType.String;

p[6] = new SqlParameter("@country", this.country);

p[6].DbType = DbType.String;

p[7] = new SqlParameter("@state", this.state);

p[7].DbType = DbType.String;

p[8] = new SqlParameter("@city", this.city);

p[8].DbType = DbType.String;

p[9] = new SqlParameter("@age", this.age);

p[9].DbType = DbType.Int16;

p[10] = new SqlParameter("@iopatient", this.iopatient);

p[10].DbType = DbType.String;

p[11] = new SqlParameter("@doctorcode", this.doctorcode);

p[11].DbType = DbType.String;

p[12] = new SqlParameter("@admitdate", this.admitdate);

p[12].DbType = DbType.DateTime;

p[13] = new SqlParameter("@admittime", this.admittime);

p[13].DbType = DbType.String;

p[14] = new SqlParameter("@testcode", this.testcode);

p[14].DbType = DbType.String;

p[15] = new SqlParameter("@roomcode", this.roomcode);

p[15].DbType = DbType.String;

p[16] = new SqlParameter("@advance", this.advance);

p[16].DbType = DbType.Int32;

p[17] = new SqlParameter("@condition", this.condition);

p[17].DbType = DbType.String;

SqlHelper.ExecuteNonQuery(con, CommandType.StoredProcedure, "SpAddPatient", p);

}

public DataSet ShowPatientInfo()

{

ds = new DataSet();

SqlParameter[] p = new SqlParameter[1];

p[0] = new SqlParameter("@code", this.code);

p[0].DbType = DbType.String;

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowPatientInfo", p);

return ds;

}

public DataSet ShowPatientCode()

{

ds = new DataSet();

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowPatientCode");

return ds;

}

public DataSet ShowDoctorChargeByCode()

{

ds = new DataSet();

SqlParameter[] p = new SqlParameter[1];

p[0] = new SqlParameter("@code", this.code);

p[0].DbType = DbType.String;

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowDoctorChargeByCode", p);

return ds;

}

public DataSet ShowTestChargeByCode()

{

ds = new DataSet();

SqlParameter[] p = new SqlParameter[1];

p[0] = new SqlParameter("@code", this.code);

p[0].DbType = DbType.String;

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowTestChargeByCode", p);

return ds;

}

public DataSet ShowRoomChargeByCode()

{

ds = new DataSet();

SqlParameter[] p = new SqlParameter[1];

p[0] = new SqlParameter("@code", this.code);

p[0].DbType = DbType.String;

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowRoomChargeByCode", p);

return ds;

}

public DataSet ShowMedicineChargeByCode()

{

ds = new DataSet();

SqlParameter[] p = new SqlParameter[1];

p[0] = new SqlParameter("@code", this.code);

p[0].DbType = DbType.String;

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowMedicineChargeByCode", p);

return ds;

}

public void UpdatePatient()

{

SqlParameter[] p = new SqlParameter[1];

p[0] = new SqlParameter("@code", this.code);

p[0].DbType = DbType.String;

SqlHelper.ExecuteNonQuery(con, CommandType.StoredProcedure, "SpUpdatePatient", p);

}

public DataSet ShowPatientInfoByDate()

{

ds = new DataSet();

SqlParameter[] p = new SqlParameter[1];

p[0] = new SqlParameter("@date", this.admitdate);

p[0].DbType = DbType.DateTime;

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowPatientInfoByDate", p);

return ds;

}

public DataSet ShowPatientInfoByDoctor()

{

ds = new DataSet();

SqlParameter[] p = new SqlParameter[1];

p[0] = new SqlParameter("@drcode", this.doctorcode);

p[0].DbType = DbType.String;

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowPatientInfoByDoctor", p);

return ds;

}

public DataSet ShowPatientInfoBetweenDate()

{

ds = new DataSet();

SqlParameter[] p = new SqlParameter[2];

p[0] = new SqlParameter("@date1", this.date1);

p[0].DbType = DbType.DateTime;

p[1] = new SqlParameter("@date2", this.date2);

p[1].DbType = DbType.DateTime;

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowPatientBetweenDate", p);

return ds;

}

public DataSet ShowPatientCodeByStatus()

{

ds = new DataSet();

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowPatientCodeByStatus");

return ds;

}

public void DeletePatientPermanently()

{

SqlParameter[] p = new SqlParameter[1];

p[0] = new SqlParameter("@code", this.code);

p[0].DbType = DbType.String;

SqlHelper.ExecuteNonQuery(con, CommandType.StoredProcedure, "SpDeletePatientPermanently", p);

}

}

**CITY BL:**

using System;

using System.Data;

using System.Configuration;

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 HospitalMgmt.DAL;

using System.Data.SqlClient;

/// <summary>

/// Summary description for CityBL

/// </summary>

public class CityBL:Connection

{

public CityBL()

{

//

// TODO: Add constructor logic here

//

}

string cityName,cityDesc;

int stateId;

public static DataSet ds;

public string CityName

{

get { return cityName; }

set { cityName = value; }

}

public string CityDesc

{

get { return cityDesc; }

set { cityDesc = value; }

}

int countryId;

public int StateId

{

get { return stateId; }

set { stateId = value; }

}

public void InsertCity()

{

SqlParameter[] p = new SqlParameter[3];

p[0] = new SqlParameter("@CityName", this.cityName);

p[0].DbType =DbType.String;

p[1] = new SqlParameter("@CityDesc", this.cityDesc);

p[1].DbType =DbType.String;

p[2] = new SqlParameter("@StateId", this.stateId);

p[2].DbType = DbType.Int16;

SqlHelper.ExecuteNonQuery(con, CommandType.StoredProcedure, "AddCity", p);

}

public DataSet ShowAllCity()

{

DataSet ds = new DataSet();

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "ShowCity");

return ds;

}

public DataSet GetCityByState()

{

ds = new DataSet();

SqlParameter[] p = new SqlParameter[1];

p[0] = new SqlParameter("stateid", this.stateId);

p[0].DbType = DbType.Int16;

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "spGetCityByStateId", p);

return ds;

}

}

**DISCARGE PATIENT BL:**

using System;

using System.Data;

using System.Configuration;

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;

using HospitalMgmt.DAL;

/// <summary>

/// Summary description for DischargePatientBL

/// </summary>

public class DischargePatientBL:Connection

{

public static DataSet ds;

public DischargePatientBL()

{

//

// TODO: Add constructor logic here

//

}

string code,name,hname,complain,sex,address,country,state,city,doctorname,roomname,inout,admittime;

public string Code

{

get { return code; }

set { code = value; }

}

public string Name

{

get { return name; }

set { name = value; }

}

public string Hname

{

get { return hname; }

set { hname = value; }

}

public string Complain

{

get { return complain; }

set { complain = value; }

}

public string Sex

{

get { return sex; }

set { sex = value; }

}

public string Address

{

get { return address; }

set { address = value; }

}

public string Country

{

get { return country; }

set { country = value; }

}

public string State

{

get { return state; }

set { state = value; }

}

public string City

{

get { return city; }

set { city = value; }

}

public string Doctorname

{

get { return doctorname; }

set { doctorname = value; }

}

public string Roomname

{

get { return roomname; }

set

{

if (value.ToString() == "")

{

roomname = "";

}

else

{

roomname = value;

}

}

}

public string Inout

{

get { return inout; }

set { inout = value; }

}

public string Admittime

{

get { return admittime; }

set { admittime = value; }

}

string dischargetime, daystayed, condition;

public string Dischargetime

{

get { return dischargetime; }

set { dischargetime = value; }

}

public string Daystayed

{

get { return daystayed; }

set { daystayed = value; }

}

public string Condition

{

get { return condition; }

set { condition = value; }

}

DateTime admitdate,dischargedate;

public DateTime Admitdate

{

get { return admitdate; }

set { admitdate = value; }

}

public DateTime Dischargedate

{

get { return dischargedate; }

set { dischargedate = value; }

}

int age,advance,doctorcharges,testcharges,roomcharges,medicinecharge,extracharge,totcharge;

public int Age

{

get { return age; }

set {age = value; }

}

public int Advance

{

get { return advance; }

set

{

if (value.ToString() == "")

{

advance = 0;

}

else

{

advance = value;

}

}

}

public int Doctorcharges

{

get

{

return doctorcharges;

}

set

{

if (value.ToString() == "")

{

doctorcharges = 0;

}

else

{

doctorcharges = value;

}

}

}

public int Testcharges

{

get

{

return testcharges;

}

set

{

if (value.ToString() == "")

{

testcharges = 0;

}

else

{

testcharges = value;

}

}

}

public int Roomcharges

{

get

{

return roomcharges;

}

set

{

if (value.ToString() == "")

{

roomcharges = 0;

}

else

{

roomcharges = value;

}

}

}

public int Medicinecharge

{

get { return medicinecharge; }

set {

if (value.ToString() == "")

{

medicinecharge =0;

}

else

{

medicinecharge = value;

}

}

}

public int Extracharge

{

get { return extracharge; }

set

{

if (value.ToString() == "")

{

extracharge = 0;

}

else

{

extracharge = value;

}

}

}

public int Totcharge

{

get { return totcharge; }

set

{

if (value.ToString() == "")

{

totcharge = 0;

}

else

{

totcharge = value;

}

}

}

public void InsertDischargePatient()

{

SqlParameter[] p = new SqlParameter[26];

p[0] = new SqlParameter("@code", this.code);

p[0].DbType = DbType.String;

p[1] = new SqlParameter("@name", this.name);

p[1].DbType = DbType.String;

p[2] = new SqlParameter("@hname", this.hname);

p[2].DbType = DbType.String;

p[3] = new SqlParameter("@complain", this.complain);

p[3].DbType = DbType.String;

p[4] = new SqlParameter("@sex", this.sex);

p[4].DbType = DbType.String;

p[5] = new SqlParameter("@address", this.address);

p[5].DbType = DbType.String;

p[6] = new SqlParameter("@country", this.country);

p[6].DbType = DbType.String;

p[7] = new SqlParameter("@state", this.state);

p[7].DbType = DbType.String;

p[8] = new SqlParameter("@city", this.city);

p[8].DbType = DbType.String;

p[9] = new SqlParameter("@doctorname", this.doctorname);

p[9].DbType = DbType.String;

p[10] = new SqlParameter("@roomname", this.roomname);

p[10].DbType = DbType.String;

p[11] = new SqlParameter("@inout", this.inout);

p[11].DbType = DbType.String;

p[12] = new SqlParameter("@admitdate", this.admitdate);

p[12].DbType = DbType.DateTime;

p[13] = new SqlParameter("@admittime", this.admittime);

p[13].DbType = DbType.String;

p[14] = new SqlParameter("@dischargedate", this.dischargedate);

p[14].DbType = DbType.DateTime;

p[15] = new SqlParameter("@dischargetime", this.dischargetime);

p[15].DbType = DbType.String;

p[16] = new SqlParameter("@age", this.age);

p[16].DbType = DbType.Int16;

p[17] = new SqlParameter("@daystayed", this.daystayed);

p[17].DbType = DbType.String;

p[18] = new SqlParameter("@advance", this.advance);

p[18].DbType = DbType.Int32;

p[19] = new SqlParameter("@doctorcharges", this.doctorcharges);

p[19].DbType = DbType.Int32;

p[20] = new SqlParameter("@testcharges", this.testcharges);

p[20].DbType = DbType.Int32;

p[21] = new SqlParameter("@roomcharges", this.roomcharges);

p[21].DbType = DbType.Int32;

p[22] = new SqlParameter("@medicinecharge", this.medicinecharge);

p[22].DbType = DbType.Int32;

p[23] = new SqlParameter("@extracharge", this.extracharge);

p[23].DbType = DbType.Int32;

p[24] = new SqlParameter("@totcharge", this.totcharge);

p[24].DbType = DbType.Int32;

p[25] = new SqlParameter("@condition", this.condition);

p[25].DbType = DbType.String;

SqlHelper.ExecuteNonQuery(con, CommandType.StoredProcedure, "SpInsertDischargePatient", p);

}

}

**DOCTOR MASTER BL:**

using System;

using System.Data;

using System.Configuration;

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;

using HospitalMgmt.DAL;

/// <summary>

/// Summary description for DoctorMasterBL

/// </summary>

public class DoctorMasterBL:Connection

{

public static DataSet ds;

public DoctorMasterBL()

{

//

// TODO: Add constructor logic here

//

}

string code,name,desc,time1,time2,contactno,username,password;

public string Username

{

get { return username; }

set { username = value; }

}

public string Password

{

get { return password; }

set { password = value; }

}

public string Time1

{

get { return time1; }

set { time1 = value; }

}

public string Time2

{

get { return time2; }

set { time2 = value; }

}

public string Contactno

{

get { return contactno; }

set { contactno = value; }

}

public string Code

{ get { return code; }

set { code = value; }

}

public string Name

{

get { return name; }

set { name = value; }

}

public string Desc

{

get { return desc; }

set { desc = value; }

}

int id, charge,roleid;

public int Roleid

{

get { return roleid; }

set { roleid = value; }

}

public int Id

{

get { return id; }

set { id = value; }

}

public int Chrge

{

get { return charge; }

set {charge = value; }

}

public void InsertDoctor()

{

SqlParameter[] p = new SqlParameter[11];

p[0] = new SqlParameter("@code", this.code);

p[0].DbType = DbType.String;

p[1] = new SqlParameter("@name", this.name);

p[1].DbType = DbType.String;

p[2] = new SqlParameter("@id", this.id);

p[2].DbType = DbType.Int16;

p[3] = new SqlParameter("@time1", this.time1);

p[3].DbType = DbType.String;

p[4] = new SqlParameter("@time2", this.time2);

p[4].DbType = DbType.String;

p[5] = new SqlParameter("@contactno", this.contactno);

p[5].DbType = DbType.String;

p[6] = new SqlParameter("@charge", this.charge);

p[6].DbType = DbType.Int32;

p[7] = new SqlParameter("@desc", this.desc);

p[7].DbType = DbType.String;

p[8] = new SqlParameter("@uname", this.username);

p[8].DbType = DbType.String;

p[9] = new SqlParameter("@password", this.password);

p[9].DbType = DbType.String;

p[10] = new SqlParameter("@roleid", this.roleid);

p[10].DbType = DbType.Int16;

SqlHelper.ExecuteNonQuery(con, CommandType.StoredProcedure, "SpInsertDoctor", p);

}

public DataSet ShowDoctor()

{

ds = new DataSet();

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowDoctor");

return ds;

}

public DataSet ShowDoctorByID()

{

ds = new DataSet();

SqlParameter[] p = new SqlParameter[1];

p[0] = new SqlParameter("@id", this.id);

p[0].DbType = DbType.Int16;

ds=SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowDoctorByID", p);

return ds;

}

public void DeleteDoctor()

{

SqlParameter[] p = new SqlParameter[1];

p[0] = new SqlParameter("@id", this.id);

p[0].DbType = DbType.Int16;

SqlHelper.ExecuteNonQuery(con, CommandType.StoredProcedure, "SpDeleteDoctor", p);

}

public DataSet ShowAllDoctor()

{

//SqlParameter[] p = new SqlParameter[1];

//p[0] = new SqlParameter("@id", this.id);

//p[0].DbType = DbType.Int16;

ds = new DataSet();

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowAllDoctor");

return ds;

}

public DataSet ShowDoctorByCode()

{

ds = new DataSet();

SqlParameter[] p = new SqlParameter[1];

p[0] = new SqlParameter("@code", this.code);

p[0].DbType = DbType.String;

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowDoctorInfoByCode", p);

return ds;

}

public void ModifyDoctorDetail()

{

SqlParameter[] p = new SqlParameter[8];

p[0] = new SqlParameter("@doccode", this.code);

p[0].DbType = DbType.String;

p[1] = new SqlParameter("@name", this.name);

p[1].DbType = DbType.String;

p[2] = new SqlParameter("@id", this.id);

p[2].DbType = DbType.Int16;

p[3] = new SqlParameter("@time1", this.time1);

p[3].DbType = DbType.String;

p[4] = new SqlParameter("@time2", this.time2);

p[4].DbType = DbType.String;

p[5] = new SqlParameter("@contact", this.contactno);

p[5].DbType = DbType.String;

p[6] = new SqlParameter("@charge", this.charge);

p[6].DbType = DbType.Int32;

p[7] = new SqlParameter("@desc", this.desc);

p[7].DbType = DbType.String;

SqlHelper.ExecuteNonQuery(con, CommandType.StoredProcedure, "SpModifyDoctorDetail", p);

}

}

**LOGIN INFO BL:**

using System;

using System.Data;

using System.Configuration;

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;

using HospitalMgmt.DAL;

/// <summary>

/// Summary description for LoginInfoBL

/// </summary>

public class LoginInfoBL:Connection

{

public static DataSet ds;

public LoginInfoBL()

{

//

// TODO: Add constructor logic here

//

}

private string name, password;

public string Name

{

get { return name; }

set { name = value; }

}

public string Password

{

get { return password; }

set { password = value; }

}

DateTime date;

public DateTime Date

{

get { return date; }

set { date = value; }

}

public bool CheckAdmininfo()

{

SqlParameter[] p = new SqlParameter[2];

p[0] = new SqlParameter("@name", this.name);

p[0].DbType = DbType.String;

p[1] = new SqlParameter("@password", this.password);

p[1].DbType = DbType.String;

int count;

count=int.Parse(SqlHelper.ExecuteScalar(con,CommandType.StoredProcedure,"SpCheckAdminInfo",p).ToString());

if (count > 0)

{

return true;

}

else

{

return false;

}

}

public bool CheckEmployeeinfo()

{

SqlParameter[] p = new SqlParameter[2];

p[0] = new SqlParameter("@name", this.name);

p[0].DbType = DbType.String;

p[1] = new SqlParameter("@password", this.password);

p[1].DbType = DbType.String;

int count;

count = int.Parse(SqlHelper.ExecuteScalar(con, CommandType.StoredProcedure, "SpCheckEmployeeInfo", p).ToString());

if (count > 0)

{

return true;

}

else

{

return false;

}

}

public bool CheckDoctorInfo()

{

SqlParameter[] p = new SqlParameter[2];

p[0] = new SqlParameter("@name", this.name);

p[0].DbType = DbType.String;

p[1] = new SqlParameter("@password", this.password);

p[1].DbType = DbType.String;

int count;

count = int.Parse(SqlHelper.ExecuteScalar(con, CommandType.StoredProcedure, "SpCheckDoctorInfo", p).ToString());

if (count > 0)

{

return true;

}

else

{

return false;

}

}

public DataSet ShowPatientByDoctor()

{

SqlParameter[] p = new SqlParameter[3];

p[0] = new SqlParameter("@uname", this.name);

p[0].DbType = DbType.String;

p[1] = new SqlParameter("@password", this.password);

p[1].DbType = DbType.String;

p[2] = new SqlParameter("@date", this.date);

p[2].DbType = DbType.Date;

ds = new DataSet();

ds=SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowDoctorByLogin", p);

return ds;

}

public DataSet ShowPatientByDoctor1()

{

SqlParameter[] p = new SqlParameter[3];

p[0] = new SqlParameter("@uname", this.name);

p[0].DbType = DbType.String;

p[1] = new SqlParameter("@password", this.password);

p[1].DbType = DbType.String;

p[2] = new SqlParameter("@date", this.date);

p[2].DbType = DbType.Date;

ds = new DataSet();

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowDoctorByLogin1", p);

return ds;

}

public void ChangeDoctorPassword()

{

SqlParameter[] p = new SqlParameter[2];

p[0] = new SqlParameter("@uname", this.name);

p[0].DbType = DbType.String;

p[1] = new SqlParameter("@password", this.password);

p[1].DbType = DbType.String;

SqlHelper.ExecuteNonQuery(con, CommandType.StoredProcedure, "SpChangeDoctorPassword", p);

}

public void ChangeEmployeePassword()

{

SqlParameter[] p = new SqlParameter[2];

p[0] = new SqlParameter("@uname", this.name);

p[0].DbType = DbType.String;

p[1] = new SqlParameter("@password", this.password);

p[1].DbType = DbType.String;

SqlHelper.ExecuteNonQuery(con,CommandType.StoredProcedure, "SpChangeEmployeePassword", p);

}

public DataSet ShowDoctorInfo()

{

SqlParameter[] p = new SqlParameter[2];

p[0] = new SqlParameter("@uname", this.name);

p[0].DbType = DbType.String;

p[1] = new SqlParameter("@password", this.password);

p[1].DbType = DbType.String;

ds = new DataSet();

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowDoctorInfo", p);

return ds;

}

}

MEDICINE CHARGE BL:

using System;

using System.Data;

using System.Configuration;

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;

using HospitalMgmt.DAL;

/// <summary>

/// Summary description for MedicineChargeBL

/// </summary>

public class MedicineChargeBL:Connection

{

public static DataSet ds;

public MedicineChargeBL()

{

//

// TODO: Add constructor logic here

//

}

string code, medicinecode, name;

public string Code

{

get { return code; }

set { code = value; }

}

public string Medicinecode

{

get { return medicinecode; }

set { medicinecode = value; }

}

public string Name

{

get { return name; }

set { name = value; }

}

int charge;

public int Charge

{

get { return charge; }

set { charge = value; }

}

DateTime date;

public DateTime Date

{

get { return date; }

set { date = value; }

}

public void InsertMedicineCharge()

{

SqlParameter[] p = new SqlParameter[5];

p[0] = new SqlParameter("@code", this.code);

p[0].DbType = DbType.String;

p[1] = new SqlParameter("@medicinecode", this.medicinecode);

p[1].DbType = DbType.String;

p[2] = new SqlParameter("@name", this.name);

p[2].DbType = DbType.String;

p[3] = new SqlParameter("@date", this.date);

p[3].DbType = DbType.DateTime;

p[4] = new SqlParameter("@charge", this.charge);

p[4].DbType = DbType.Int32;

SqlHelper.ExecuteNonQuery(con, CommandType.StoredProcedure, "SpInsertMedicineCharges", p);

}

public DataSet ShowMedicineByDate()

{

ds = new DataSet();

SqlParameter[] p = new SqlParameter[2];

p[0] = new SqlParameter("@code", this.code);

p[0].DbType = DbType.String;

p[1] = new SqlParameter("@date", this.date);

p[1].DbType = DbType.DateTime;

ds=SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowMedicineByDate", p);

return ds;

}

public DataSet ShowMedicineCode()

{

ds = new DataSet();

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowMedicineCode");

return ds;

}

public DataSet ShowpatientnameByCode()

{

ds = new DataSet();

SqlParameter[] p = new SqlParameter[1];

p[0] = new SqlParameter("@code", this.code);

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowpatientnameByCode", p);

return ds;

}

public DataSet ShowMedicineByPatientCode()

{

ds = new DataSet();

SqlParameter[] p = new SqlParameter[1];

p[0] = new SqlParameter("@code", this.code);

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowMedicineByPatientCode", p);

return ds;

}

public DataSet ShowAllMedicineInfoByCode()

{

ds = new DataSet();

SqlParameter[] p = new SqlParameter[1];

p[0] = new SqlParameter("@medcode", this.medicinecode);

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowAllMedicineInfoByCode", p);

return ds;

}

}

**ROOM CHARGE:**

using System;

using System.Data;

using System.Configuration;

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;

using HospitalMgmt.DAL;

/// <summary>

/// Summary description for RoomChargeBL

/// </summary>

public class RoomChargeBL:Connection

{

public static DataSet ds;

public RoomChargeBL()

{

//

// TODO: Add constructor logic here

//

}

string code,roomcode,time;

public string Code

{

get { return code; }

set { code = value; }

}

public string Roomcode

{

get { return roomcode; }

set { roomcode = value; }

}

public string Time

{

get { return time; }

set { time = value; }

}

int charge;

public int Charge

{

get { return charge; }

set { charge = value; }

}

DateTime date;

public DateTime Date

{

get { return date; }

set { date = value; }

}

public void InsertRoomCharge()

{

SqlParameter[] p = new SqlParameter[5];

p[0] = new SqlParameter("@code", this.code);

p[0].DbType = DbType.String;

p[1] = new SqlParameter("@roomcode", this.roomcode);

p[1].DbType = DbType.String;

p[2] = new SqlParameter("@date", this.date);

p[2].DbType = DbType.DateTime;

p[3] = new SqlParameter("@time", this.time);

p[3].DbType = DbType.String;

p[4] = new SqlParameter("@charge", this.charge);

p[4].DbType = DbType.Int32;

SqlHelper.ExecuteNonQuery(con, CommandType.StoredProcedure, "SpInsertRoomCharges", p);

}

public DataSet ShowPatientRoomCode()

{

ds = new DataSet();

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowPatientRoomCode");

return ds;

}

public DataSet ShowRoomNameByCode()

{

SqlParameter[] p = new SqlParameter[1];

p[0] = new SqlParameter("@code", this.roomcode);

p[0].DbType = DbType.String;

ds = new DataSet();

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowRoomNameByCode",p);

return ds;

}

public DataSet ShowPatientInfoByRoomCode()

{

SqlParameter[] p = new SqlParameter[1];

p[0] = new SqlParameter("@code", this.roomcode);

p[0].DbType = DbType.String;

ds = new DataSet();

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowPatientInfoByRoomCode", p);

return ds;

}

public void UpdateRoomMaster()

{

SqlParameter[] p = new SqlParameter[1];

p[0] = new SqlParameter("@code", this.code);

p[0].DbType = DbType.String;

SqlHelper.ExecuteNonQuery(con, CommandType.StoredProcedure, "SpUpdateRoomMaster", p);

}

}

**TEST CHARGE BL:**

using System;

using System.Data;

using System.Configuration;

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;

using HospitalMgmt.DAL;

/// <summary>

/// Summary description for TestChargeBL

/// </summary>

public class TestChargeBL:Connection

{

public static DataSet ds;

public TestChargeBL()

{

//

// TODO: Add constructor logic here

//

}

string code,testcode,time;

public string Code

{

get { return code; }

set { code = value; }

}

public string Testcode

{

get { return testcode; }

set { testcode = value; }

}

public string Time

{

get { return time; }

set { time = value; }

}

DateTime date, date1, date2;

public DateTime Date

{

get { return date; }

set { date = value; }

}

public DateTime Date1

{

get { return date1; }

set { date1 = value; }

}

public DateTime Date2

{

get { return date2; }

set { date2 = value; }

}

int charge;

public int Charge

{

get { return charge; }

set { charge = value; }

}

public void InsertTestCharge()

{

SqlParameter[] p = new SqlParameter[5];

p[0] = new SqlParameter("@code", this.code);

p[0].DbType = DbType.String;

p[1] = new SqlParameter("@testcode", this.testcode);

p[1].DbType = DbType.String;

p[2] = new SqlParameter("@date", this.date);

p[2].DbType = DbType.DateTime;

p[3] = new SqlParameter("@time", this.time);

p[3].DbType = DbType.String;

p[4] = new SqlParameter("@charge", this.charge);

p[4].DbType = DbType.Int32;

SqlHelper.ExecuteNonQuery(con, CommandType.StoredProcedure, "SpInsertTestCharges", p);

}

public DataSet ShowTestInfoByDate()

{

ds = new DataSet();

SqlParameter[] p = new SqlParameter[1];

p[0] = new SqlParameter("@date", this.date);

p[0].DbType = DbType.DateTime;

ds=SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowTestInfoByDate", p);

return ds;

}

public DataSet ShowPatientCodeBYTest()

{

ds = new DataSet();

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowPatientCodeBYTest");

return ds;

}

public DataSet ShowTestInfoByDatePatient()

{

ds = new DataSet();

SqlParameter[] p = new SqlParameter[2];

p[0] = new SqlParameter("@code", this.code);

p[0].DbType = DbType.String;

p[1] = new SqlParameter("@date", this.date);

p[1].DbType = DbType.DateTime;

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowTestInfoByDatePatient", p);

return ds;

}

public DataSet ShowTestInfoByOnlyDate()

{

ds = new DataSet();

SqlParameter[] p = new SqlParameter[1];

p[0] = new SqlParameter("@date", this.date);

p[0].DbType = DbType.DateTime;

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowTestInfoByOnlyDate", p);

return ds;

}

public DataSet ShowTestCodeOnPatient()

{

ds = new DataSet();

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowTestCodeOnPatient");

return ds;

}

public DataSet ShowTestCodeByPateintCode()

{

ds = new DataSet();

SqlParameter[] p = new SqlParameter[1];

p[0] = new SqlParameter("@code", this.code);

p[0].DbType = DbType.String;

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowTestCodeByPateintCode", p);

return ds;

}

public DataSet ShowTestNameByTestCode()

{

ds = new DataSet();

SqlParameter[] p = new SqlParameter[1];

p[0] = new SqlParameter("@code", this.testcode);

p[0].DbType = DbType.String;

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowTestNameByTestCode", p);

return ds;

}

public DataSet ShowTestInfoByTestPatientCode()

{

ds = new DataSet();

SqlParameter[] p = new SqlParameter[2];

p[0] = new SqlParameter("@code", this.code);

p[0].DbType = DbType.String;

p[1] = new SqlParameter("@testcode", this.testcode);

p[1].DbType = DbType.String;

ds = SqlHelper.ExecuteDataset(con, CommandType.StoredProcedure, "SpShowTestInfoByTestPatientCode", p);

return ds;

}

public DataSet SpShowTestBetweenDate()

{

ds = new DataSet();

SqlParameter[] p = new SqlParameter[2];

p[0] = new SqlParameter("@date1", this.date1);

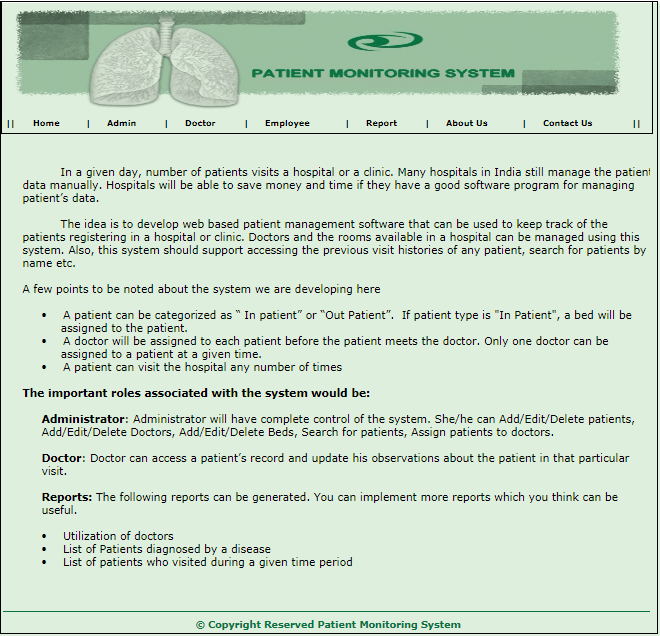
p[0].DbType = DbType.DateTime;

p[1] = new SqlParameter("@date2", this.date2);

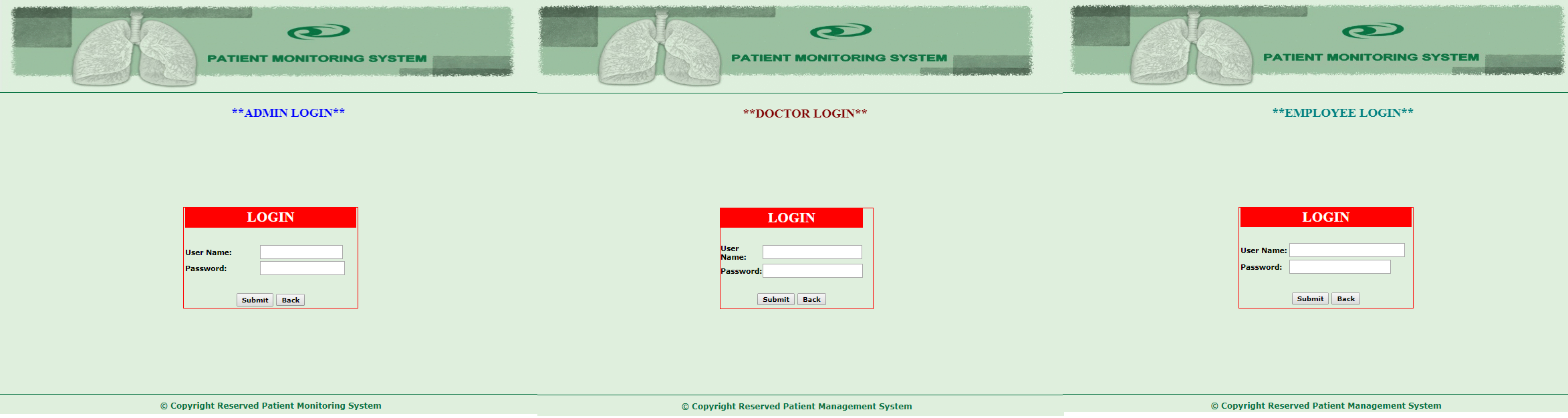
p[1].DbType = DbType.DateTime;

ds=SqlHelper.ExecuteDataset(con CommandType.StoredProcedure, "SpShowTestBetweenDate", p);

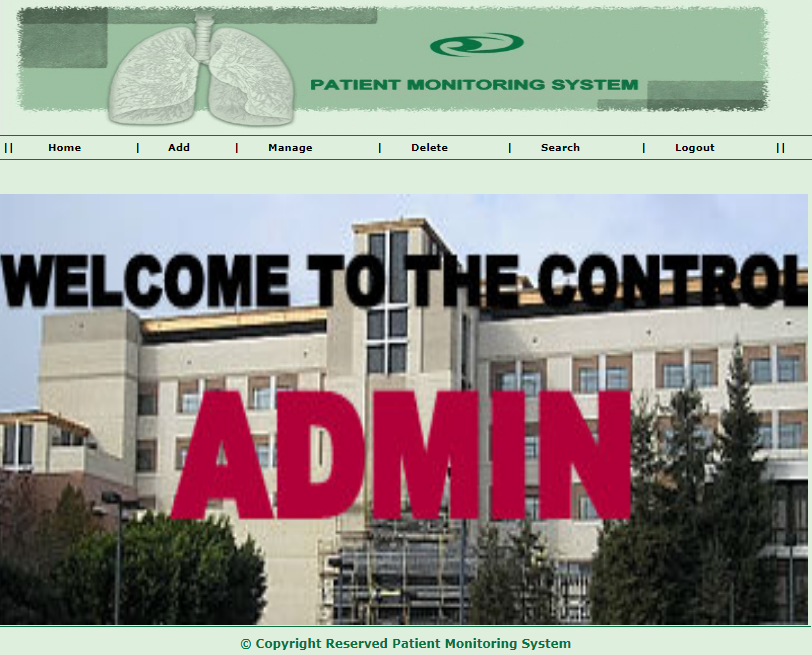
1. **Input/Test data screens/ output screen/Report screens**
2. **Welcome Page**

****

1. **Various Login pages**

****

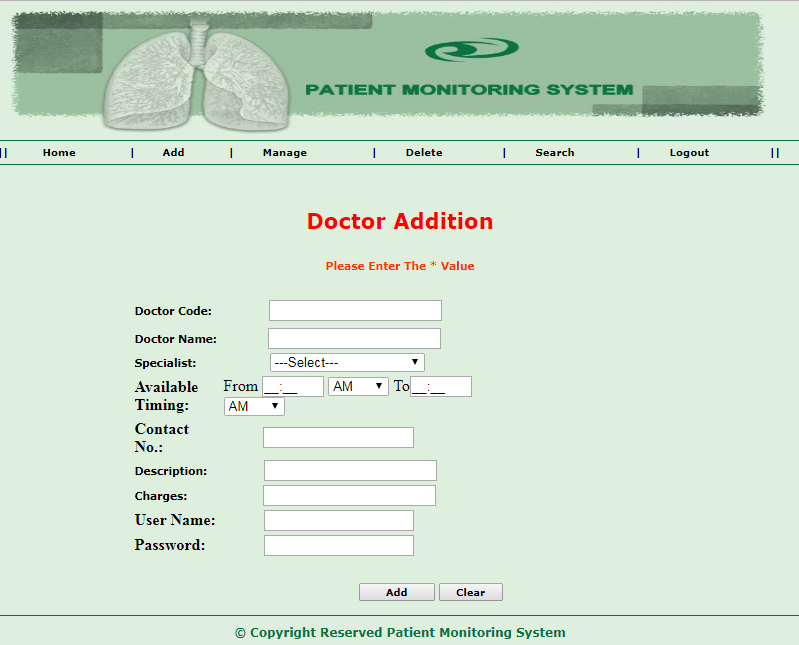
1. **Local Admin Page**

****

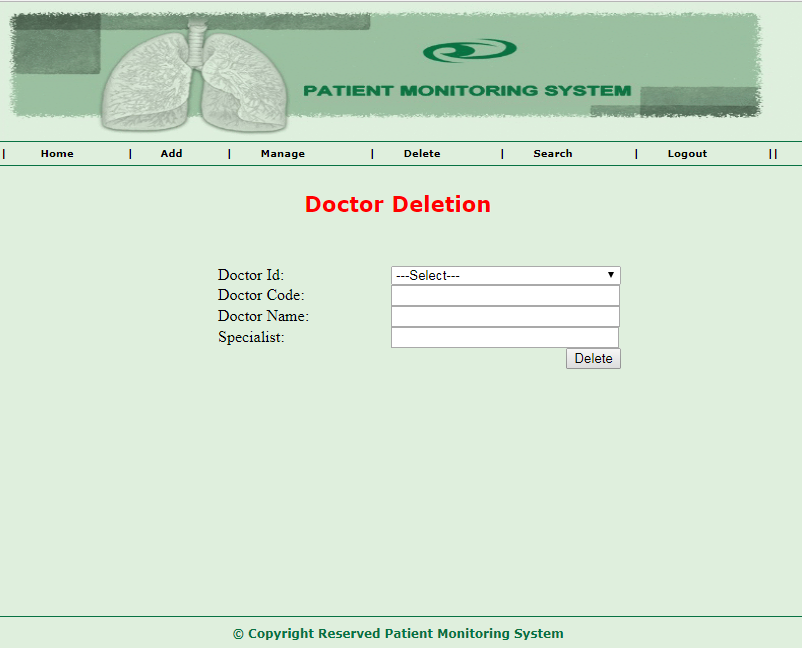
1. **Add Employee**

****

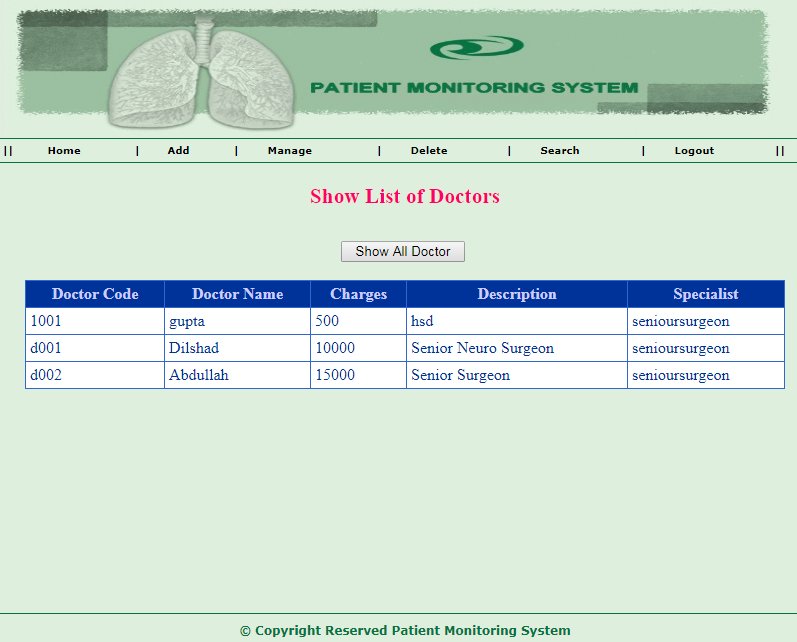
1. **Add Doctor**

****

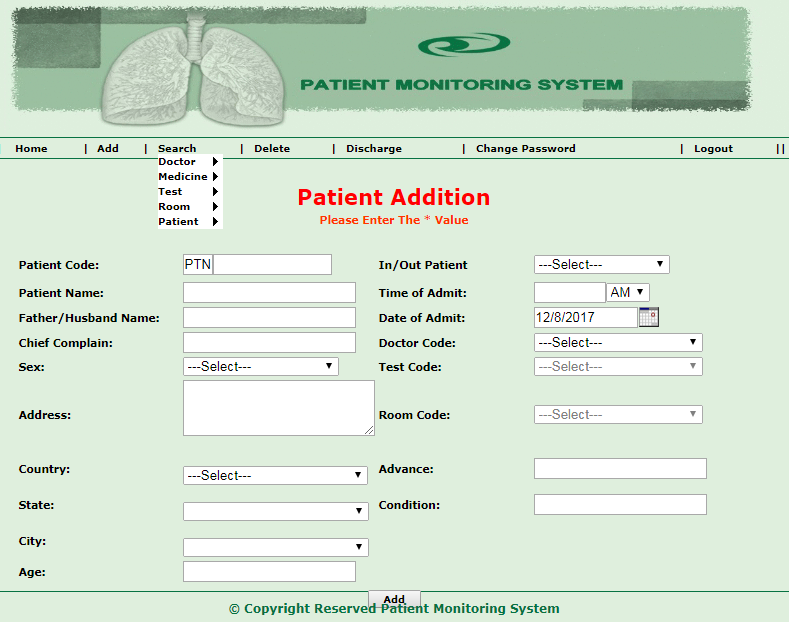
1. **Delete Doctor**

****

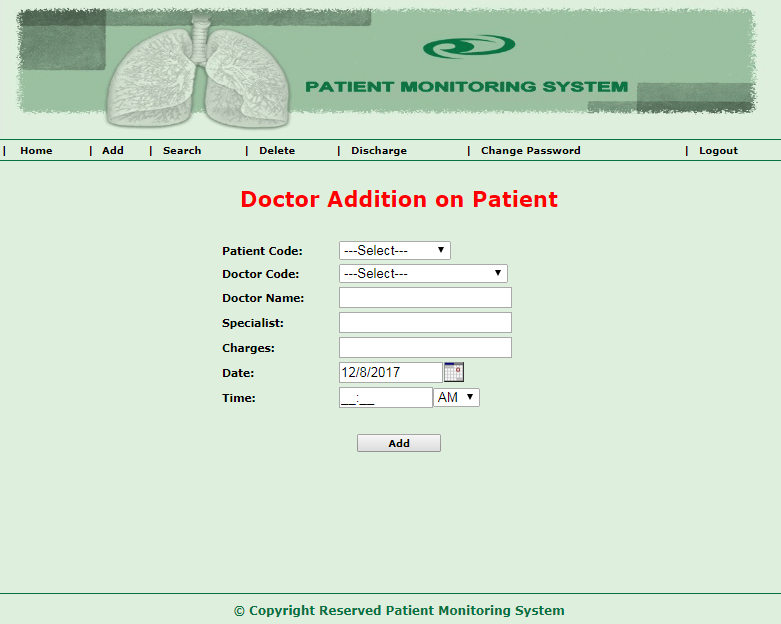
1. **Search Doctors**

****

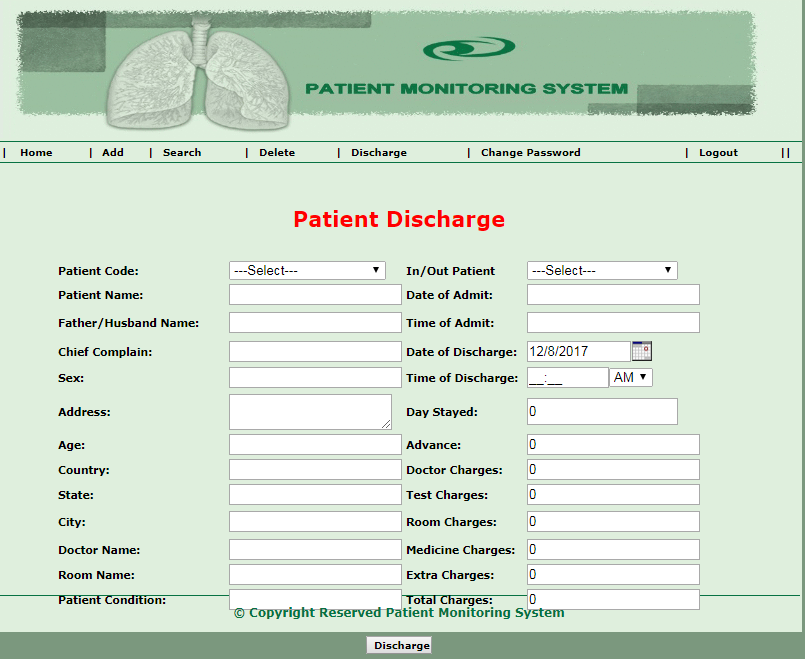
1. **Add Patient**

****

1. **Doctor on Patient**



1. **Patient Discharge**

****

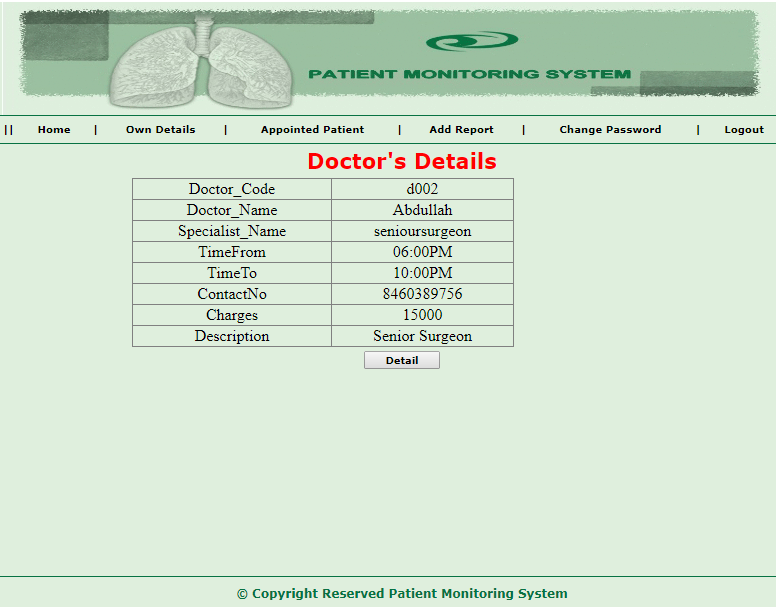
1. **Add Report**

****

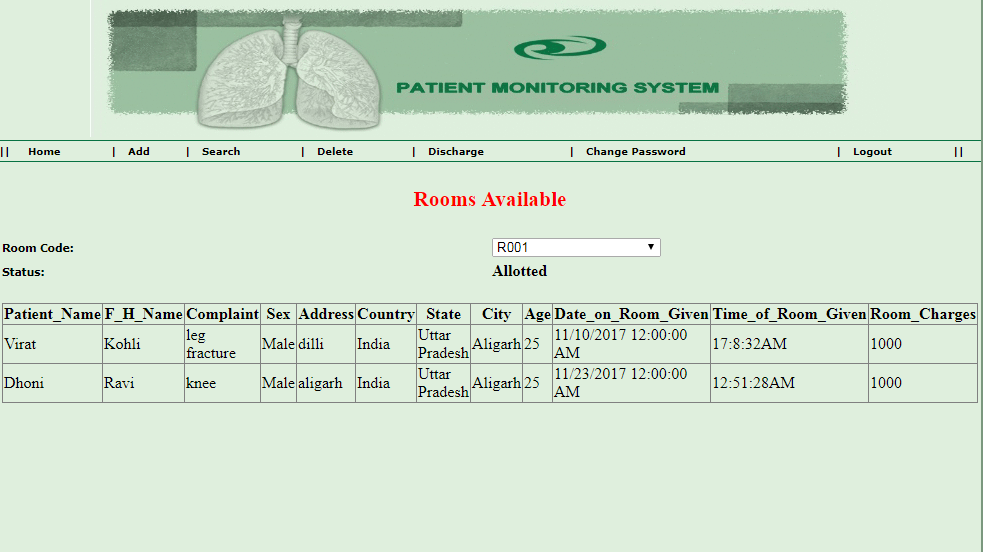
1. **View Appointed Patients**

****

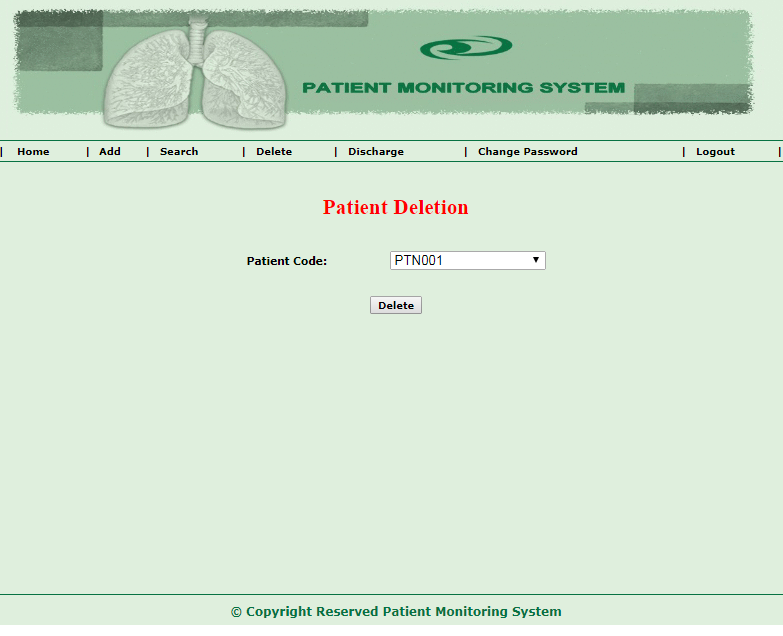
1. **View/Edit Doctor Detail**

****

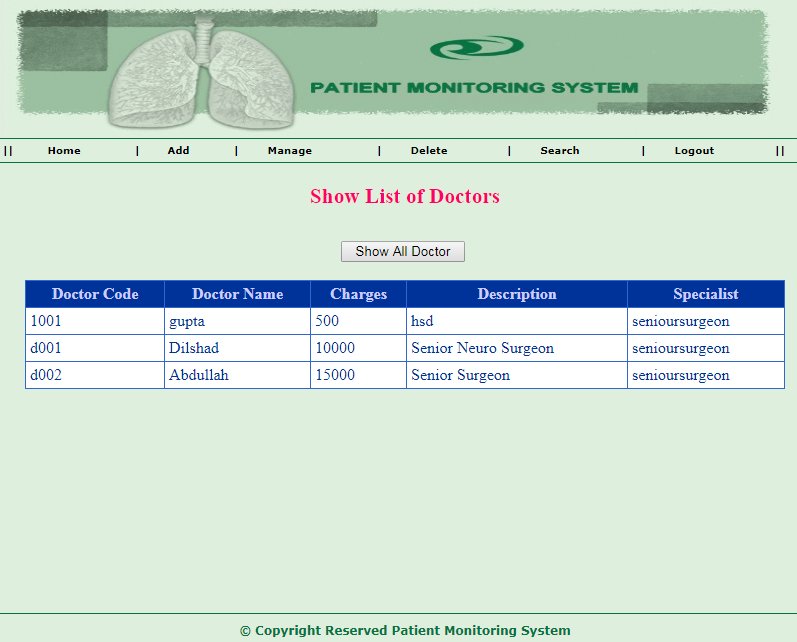
1. **Room Availability**

****

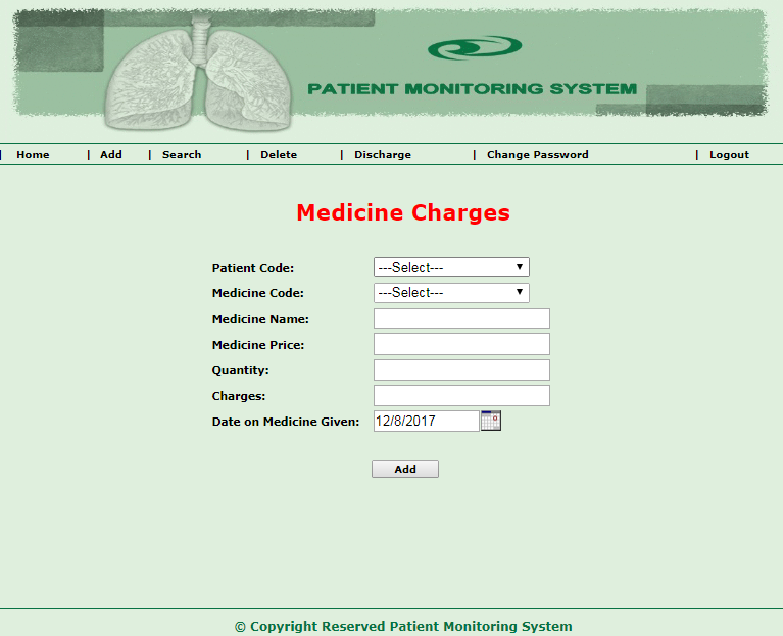
1. **Patient Information Deletion**

****

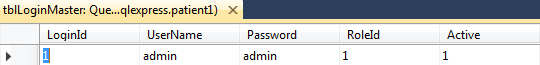
1. **Doctor Detail Report**

****

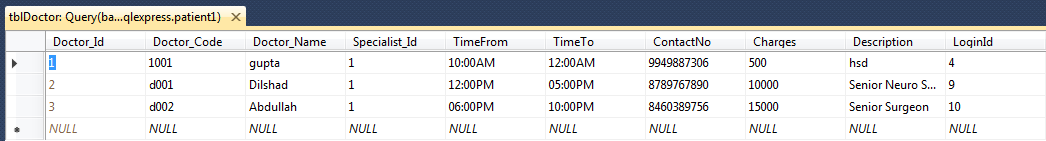
1. **Medicine Charge**

****

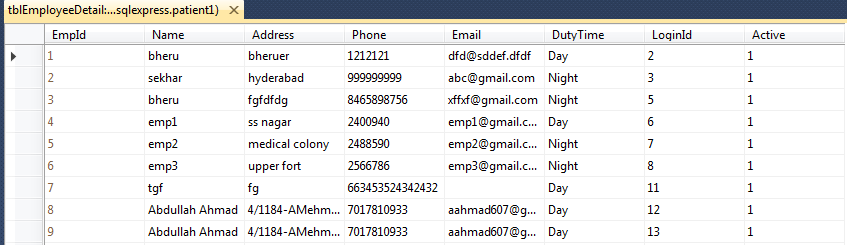
1. **Database tables**
2. **Login Table**



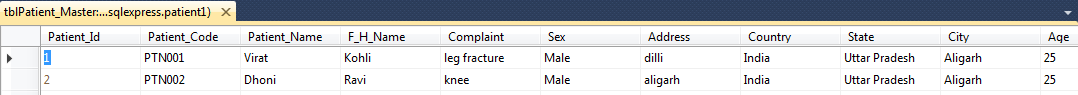
1. **Doctor Table**

****

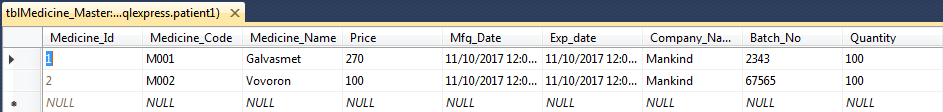
1. **Employee Table**

****

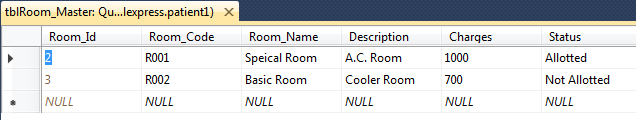
1. **Patient Table**

****

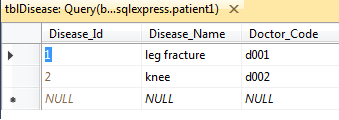
1. **Medicine Table**

****

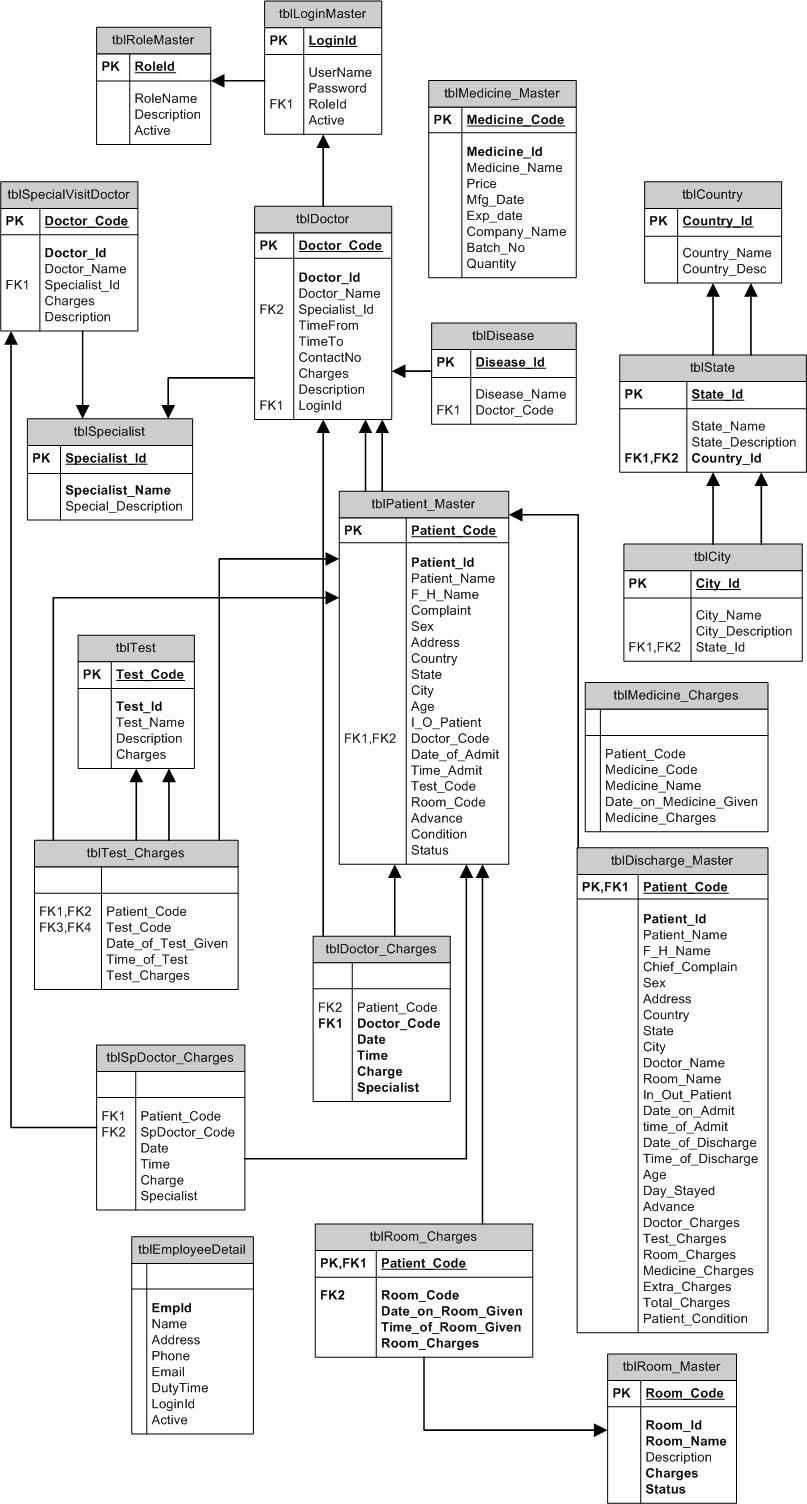
1. **Room Table**

****

1. **Disease Table**

****

1. **CLASS DIAGRAM**

****