



(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

HEALTHCARE MANAGEMENT SYSTEM

A Report for the Evaluation 3 of Project 2

Submitted by

AKASH K AJI

(1713111006)

in partial fulfilment for the award of the degree

of

BACHELOR OF COMPUTER APPLICATION

IN

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

Under the Supervision of

MR.P. RAJAKUMAR

Assistant Professor

APRIL / MAY- 2020



(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report “HEALTHCARE MANAGEMENT SYSTEM” is the bonafide work of “AKASH K AJI (1713111006)” who carried out this project work under my supervision.

SIGNATURE OF HEAD

Dr. MUNISH SABHARWAL
PhD (Management), PhD(CS)
Professor & Dean,
School of Computer Science &
Engineering

SIGNATURE OF SUPERVISOR

MR.P.RAJAKUMAR
Assistant Professor
School of Computer Science &
Engineering

ACKNOWLEDGEMENT

I am extremely grateful and remain indebted to my guide MR.P.RAJAKUMAR for being a source of inspiration and for his constant support in the Design, Implementation and Evaluation of the project. I am thankful for her constant constructive criticism and invaluable suggestions, which benefited me alot while developoing the project on “HEALTHCARE MANAGEMENT SYSTEM”. She has been constant source of inspiration and motivation for hard work. She has been very co-operative throughout this project work. Through this column, it would be my utmost pleasure to express my warm thanks for her encouragement, co-operation and consent without which I might not be able to accomplish this project.

I also express my gratitude to MR.P.RAJAKUMAR for providing me the infrastructure to carry out the project and to all staff members of my collage who were directly and indirectly medium in enabling me to stay committed for the project.

TABLE OF CONTENTS

CHAPTER NO	TITLE
1.	Abstract
2.	Introduction
3.	Existing System
4.	Proposed system
5.	Implementation or architecture diagrams
6.	Output / Result / Screenshot
7.	Conclusion/Future Enhancement
8.	References

ABSTRACT

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 healthcare 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
- The project has been planned to be having the view of distributed architecture, with centralized storage of the database. The application for the storage of the data has been planned. Using the constructs of MS-SQL Server and all the user interfaces have been designed using the ASP.Net technologies. The database connectivity is planned using the "SQL Connection" methodology.

The standards of security and data protective mechanism have been given a big choice for proper usage.

The application takes care of different modules and their associated reports, which are produced as per the applicable strategies and standards that are put forwarded by the administrative staff.

- The entire project has been developed keeping in view of the distributed client server computing technology, in mind.
- The total front end was dominated using the ASP.Net technologies. At all proper levels high care was taken to check that the system manages the data consistency with proper business rules or validations. The database connectivity was planned using the latest "SQL Connection" technology provided by Microsoft Corporation. The authentication and authorization was crosschecked at all the relevant stages.

INTRODUCTION

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 healthcare 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.

The important roles associated with the system would be:

Administrator: 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.

Doctor: Doctor can access a patient's record and update his observations about the patient in that particular visit.

Reports: The following reports can be generated. You can implement more reports which you think can be useful.

- Utilization of doctors
- List of Patients diagnosed by a disease
- List of patients who visited during a given time period

THIS APPROACH RESTS ON:

- A strategy where we architect, integrate and manage technology services and solutions - we call it AIM for success.
- A robust offshore development methodology and reduced demand on customer resources.
- A focus on the use of reusable frameworks to provide cost and times benefits.

They combine the best people, processes and technology to achieve excellent results - consistency. We offer customers the advantages of:

SPEED:

They understand the importance of timing, of getting there before the competition. A rich portfolio of reusable, modular frameworks helps jump-start projects. Tried and tested methodology ensures that we follow a predictable, low - risk path to achieve results. Our track record is testimony to complex projects delivered within and even before schedule.

EXPERTISE:

Our teams combine cutting edge technology skills with rich domain expertise. What's equally important - they share a strong customer orientation that means they actually start by listening to the customer. They're focused on coming up with solutions that serve customer requirements today and anticipate future needs.

A FULL SERVICE PORTFOLIO:

They offer customers the advantage of being able to Architect, integrate and manage technology services. This means that they can rely on one, fully accountable source instead of trying to integrate disparate multi-vendor solutions.

FEATURES:

- **Registration** - This feature allows to add/edit/delete patients in the system.
- **Patient Management** - Before we let a user to visit a doctor, we should classify whether he is an in-patient or an outpatient. Then basic symptoms should be captured and entered in the system. Existing patients can be searched by name, admission date, patient ID etc.
- **Patient Visit History** - Previous visit histories of the patient can be accessed
- **Assign Doctor** - Assign a patient to a doctor based on the basic information of the patient's problems. Based on the specialization and availability of the doctors, patients should be assigned.
- **Doctor Management** – Add / Edit / Delete the doctors working in the hospital.
- **Bed/Room Management** – The beds or rooms in a hospital can be managed. Add / Edit / Update the rooms with details. Check availability and assign a room to in-patients.
- Doctors can update the patient's record with their observation

PURPOSE

This project is aimed to developing patient healthcare management system. The entire project has been developed keeping in view of the distributed client server computing technology, in mind. This is useful for admin no matter where he is only by login through his username and password he can see information about his hospital like how many patients are admitted on a particular date, information about doctors, information about medicine, and test done on admitted patient by date. Admin Online can add new doctor, medicine, test type etc. into his hospital. He can add employee and assign them work.

SCOPE

Now a day at every place computers are used for making the task easier and it provide flexibility for more productive output. There are several day to day activities performed at hospital and it is very necessary to keep the data in such a way that whenever it is required, it can easily available in time. Hospital activities can be divided into different sections such as management section, administration section and doctor's activities etc. Each section has its own work to carry out. Sometimes it becomes difficult to manage all these activities, for example if doctor wants to check patients schedule the current date and he also wants to retrieve old patient's conversation dialog for reference, then it becomes difficult for him to locate such information. Further if administrator wants some reports of back date in a format then it's become difficult for him to find such a data. In multispecialty hospitals various doctors from other hospitals also provides their services. It is rather possible that the location of hospital might be in different city. In such situation if doctor sitting on Delhi wants to retrieve the information of patient consultation done in Kerala then it becomes difficult to find and retrieve the information. The developed system helps the doctor, admin, patient to find their information irrespective of location barrier. From the hospital activities such issues are identified and based on that web based Multi Agent System is developed using Ontology. The agent performs the assigned task and in several situations the agents communicates with other agents, thus inter agent communication is also implemented.

LITEATURE SURVEY

Multi Agent system is growing rapidly and it attracts more researchers for research. Healthcare is one of the domains which can take gain of agents. Multi agent systems are suitable for developing healthcare applications where the use of loosely coupled and heterogeneous components, the dynamic and distributed management of data and the remote collaboration between various users is considered as relevant requirements. [LSA]. Apart from agent technology ontology is also popular in computing community. Ontology along with agent technology plays a vital role for developing applications for healthcare. Healthcare domain is one of the domain where consistent data, maintenance of large volume of data, manipulation of data are required. Ontology is useful for managing consistency between various environments. It is also helpful for building the knowledge base by defining various classes and its properties and dependency. Later on this knowledge base can also be useful for data mining purpose. Multi agents systems are considered as future which efficiently deals with various kinds of problems in health care. An agent based system allows communicating between agents by using interface. The interface will provide different functions like registering patients, checking stock, decision making based on doctors remark etc. The system can be implemented either on web based environment or windows based environment or in both environment. Below list shows some of the domains where it is applicable:

- **Patient scheduling:** It schedules the activities performed on a patient in hospital. [LSAA]
- **Organ transplant management:** It helps to co-ordinate organ management and tissue transplants between different medical locations. [LSBB][LSCC]
- **Community care:** It performs all the activities which help to provide a well-organized health care benefits to the various people of communities. [LSDD]
- **Information access:** This domain helps the agents to collect, compile and unite healthcare information available on internet. [LSEE][LSFF]. It also provides mobile users with information about the medical locations or the doctors available in a particular town. [LSJJ]
- **Decision aid systems:** This system monitors the status of a hospitalized patient and helps to diagnose the patients. [LSGG]
- **Internal hospital tasks:** It is related to internal activities performed for the hospitals like monitoring the application of medical protocols [LSHH] or controlling the usage of restricted use antibiotics. [LSII]

PROBLEM STATEMENT

PROBLEM IN EXISTING SYSTEM

- 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.

SOLUTION OF THESE PROBLEMS

- 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.

SYSTEM REQUIREMENTS SPECIFICATION

Hardware requirements:

- **Hard disk:** PC with 2 GB hard-disk
- **Ram:** 256 MB RAM
- **Processor:** Intel Dual Core

Software requirements:

- **Operating System:** Windows 2000/ XP/ or Higher with MS-office
- **Database Server:** MS-SQL server2000/2005
- **Tools:** Ms-Visual Studio .NET 2005
- Ms-Internet Explorer
- **Code Behind:** VC# .NET

PROPESED SYSTEM

To debug the existing system, remove procedures those cause data redundancy, make navigational sequence proper. To provide information about audits on different level and also to reflect the current work status depending on organization/auditor or date. To build strong password mechanism.

NEED FOR COMPUTERIZATION

We all know the importance of computerization. The world is moving ahead at lightning speed and everyone is running short of time. One always wants to get the information and perform a task he/she/they desire(s) within a short period of time and too with amount of efficiency and accuracy. The application areas for the computerization have been selected on the basis of following factors:

- Minimizing the manual records kept at different locations.
- There will be more data integrity.
- Facilitating desired information display, very quickly, by retrieving information from users.
- Facilitating various statistical information which helps in decision-making?
- To reduce manual efforts in activities that involved repetitive work.
- Updating and deletion of such a huge amount of data will become easier.

FEASIBILITY STUDIES

Technical Feasibility

- The technical issue usually raised during the feasibility stage of the investigation includes the following:
- Does the necessary technology exist to do what is suggested?
- Do the proposed equipment's have the technical capacity to hold the data required to use the new system?
- Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
- Can the system be upgraded if developed?
- Are there technical guarantees of accuracy, reliability, ease of access and data security?

Operational Feasibility

- Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization's operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. Some of the important issues raised are to test the operational feasibility of a project includes the following:
- Will the system be used and work properly if it is being developed and implemented?
- This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits.
- The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

Economic Feasibility

- A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs.
- The system is economically feasible. It does not require any addition hardware or software. There is nominal expenditure and economical feasibility for certain.

SELECTED SOFTWARE

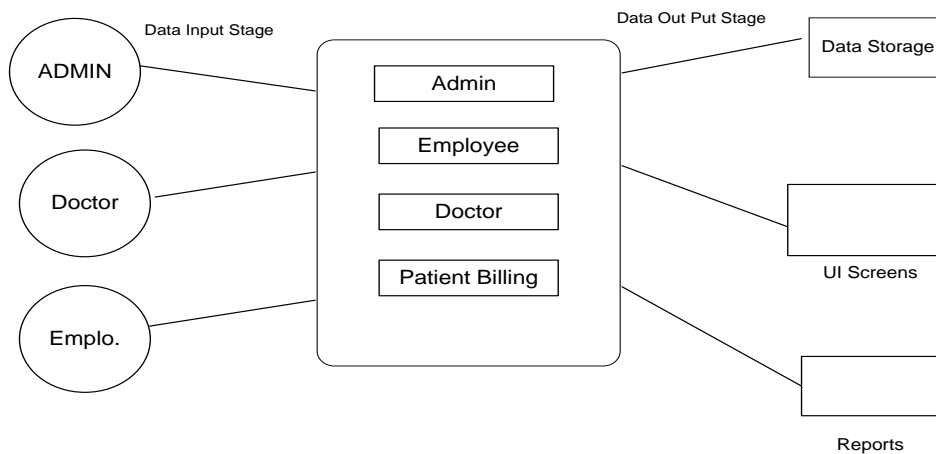
- The .NET Framework is a new computing platform that simplifies application development in the highly distributed environment of the Internet. The .NET Framework has two main components: the common language runtime and the .NET Framework class library. The .NET Framework class library is a collection of reusable types that tightly integrate with the common language runtime. The class library is object oriented, providing types from which your own managed code can derive functionality. ASP.NET is a programming framework built on the common language runtime that can be used on a server to build powerful Web applications. The ASP.NET Web Forms page framework is a scalable common language runtime programming model that can be used on the server to dynamically generate Web pages.
- ASP.NET Web Forms pages are text files with an .aspx file name extension. They can be deployed throughout an IIS virtual root directory tree. An ASP.NET page can be created simply by taking an existing HTML file and changing its file name extension to .aspx.
- A database management, or DBMS, gives the user access to their data and helps them transform the data into information. Such database management systems include dBase, paradox, IMS, SQL Server and SQL Server. These systems allow users to create, update and extract information from their database. A database is a structured collection of data. Data refers to the characteristics of people, things and events. SQL Server stores each data item in its own fields. In SQL Server, the fields relating to a particular person, thing or event are bundled together to form a single complete unit of data, called a record (it can also be referred to as raw or an occurrence). Each record is made up of a number of fields. No two fields in a record can have the same field name.

DATA FLOW DIAGRAMS

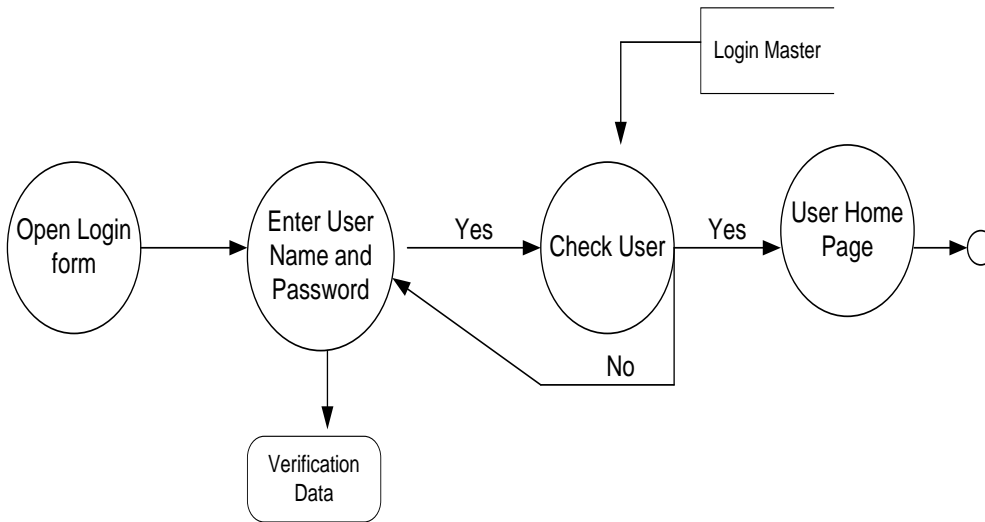
A data flow diagram is graphical tool used to describe and analyze movement of data through a system. These are the central tool and the basis from which the other components are developed. The transformation of data from input to output, through processed, may be described logically and independently of physical components associated with the system. These are known as the logical data flow diagrams.

A DFD is also known as a “bubble Chart” has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. So it is the starting point of the design to the lowest level of detail. A DFD consists of a series of bubbles joined by data flows in the system.

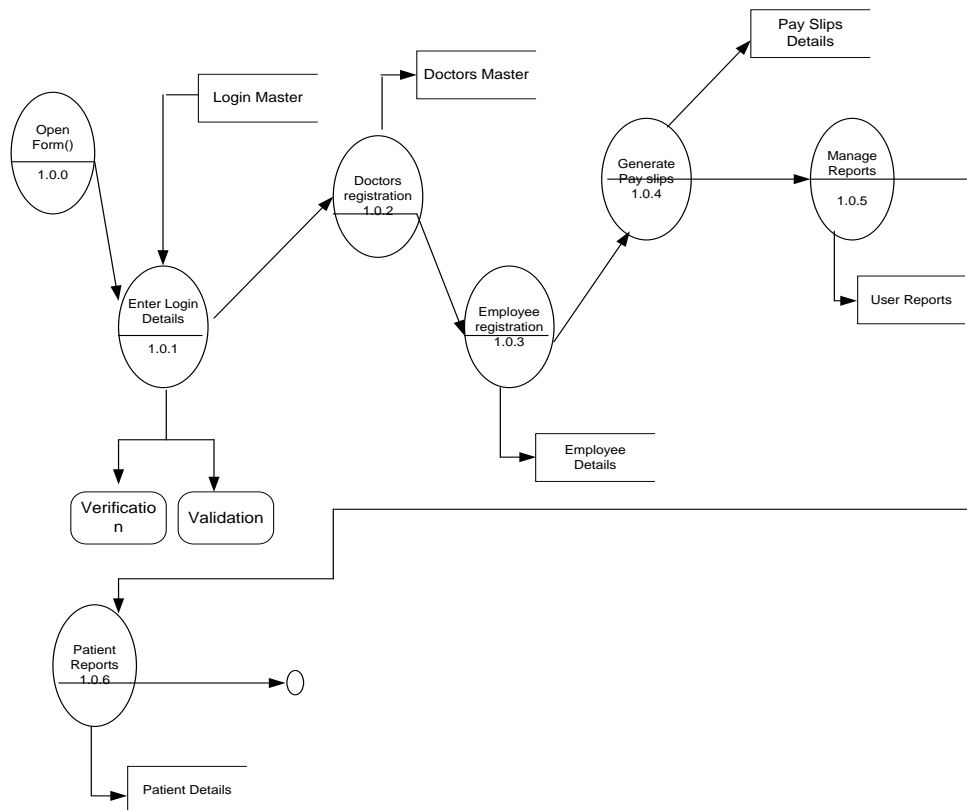
Context Level DFD



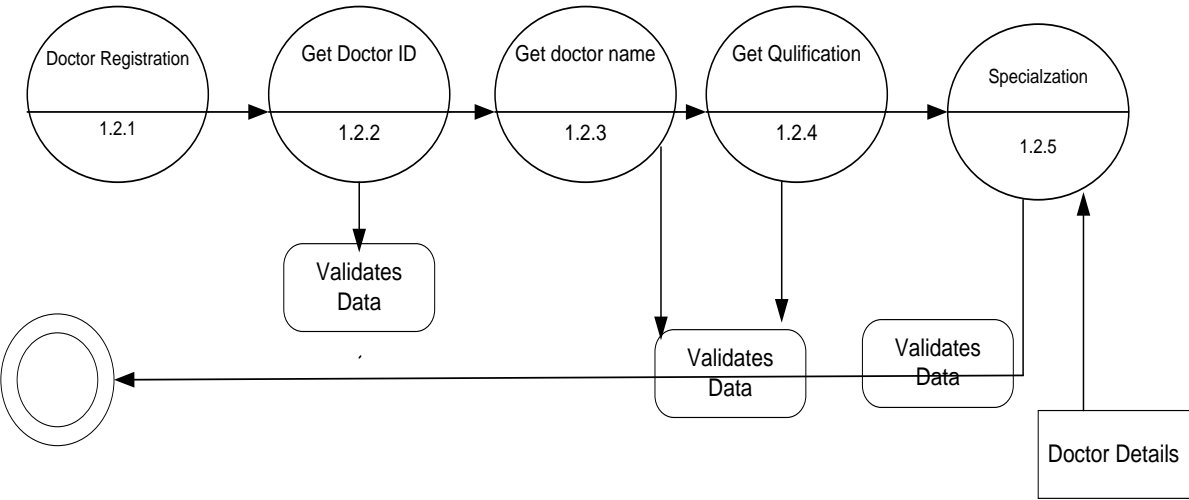
Login DFD



Admin Activities (1st Level)

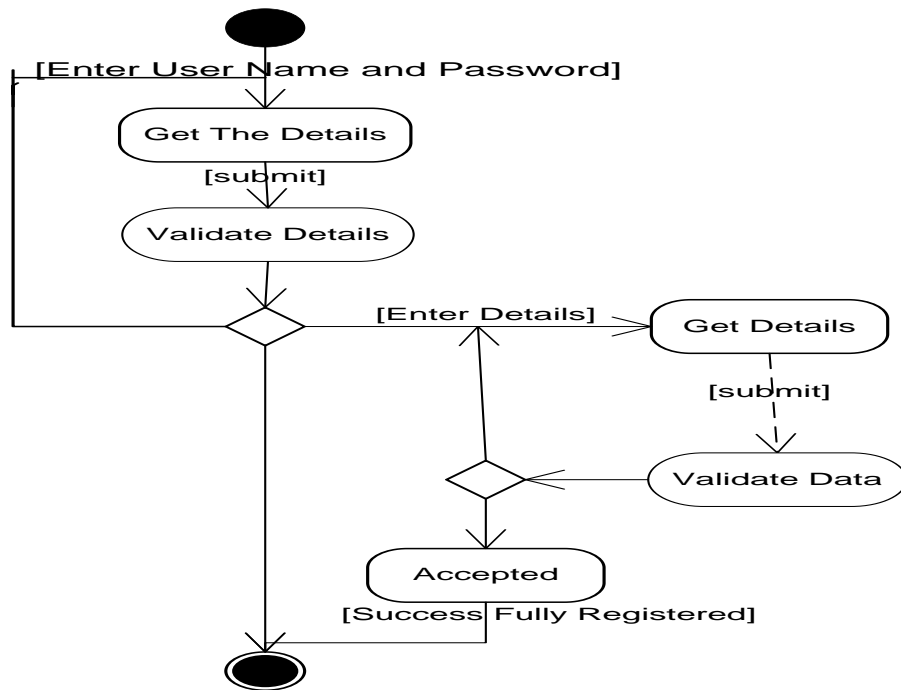


Admin Register Doctors

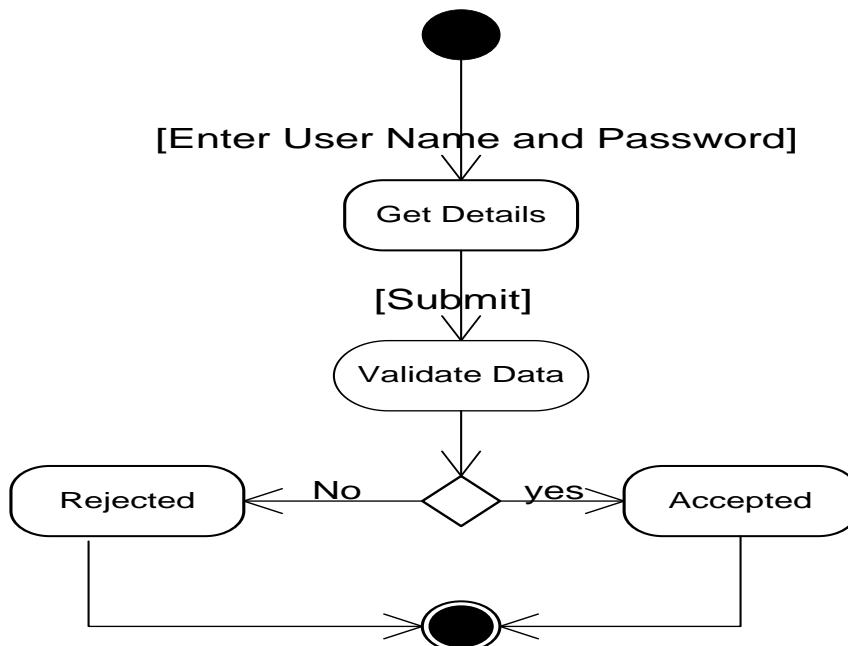


ACTIVITY DIAGRAM

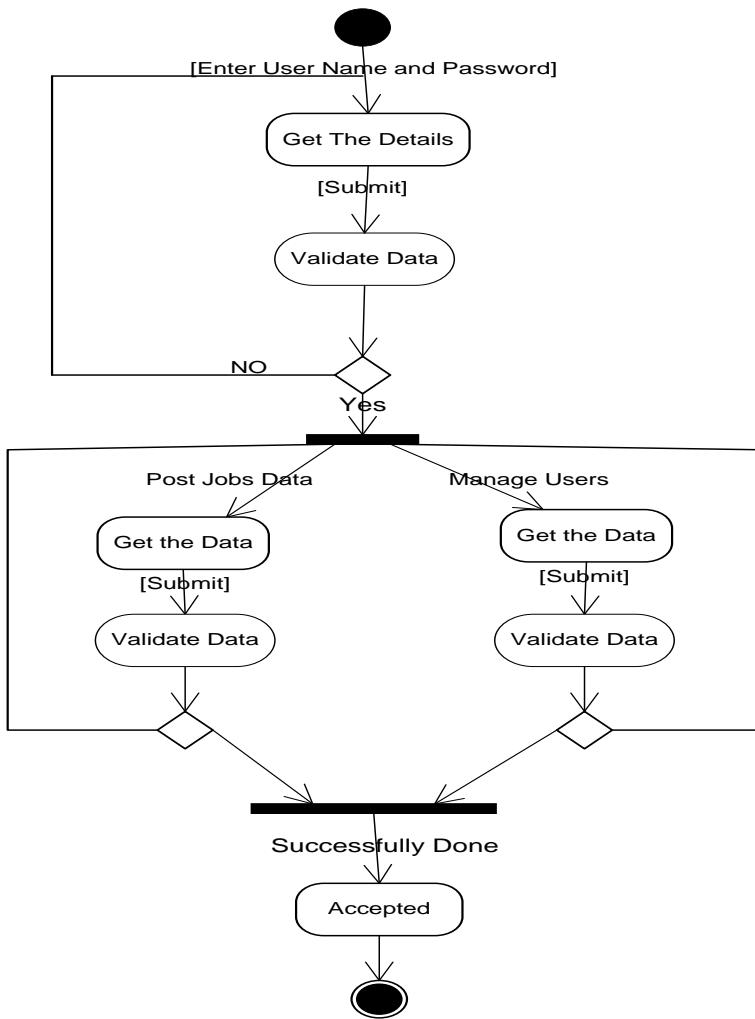
REGISTRATION ACTIVITY DIAGRAM



LOGIN ACTIVITY DIAGRAM



ADMIN ACTIVITY DIAGRAM



The following steps will be helpful to start off the project:

- Study and be comfortable with technologies such as: ASP.Net with C#, and SQL Server.
- Gather some knowledge of product hierarchies and maintenance before starting the design.
- Create a user database with different access levels.
- Start with creating the login screen.
- Create menus for navigation and group the functionalities as sub menus.
- Create the help-pages of the application in the form of FAQ. This will helps user.

SECURITY

The protection of computer based resources that includes hardware, software, data, procedures and people against unauthorized use or natural Disaster is known as System Security. System Security can be divided into four related issues:

- Security
- Integrity
- Privacy
- Confidentiality

SYSTEM SECURITY refers to the technical innovations and procedures applied to the hardware and operation systems to protect against deliberate or accidental damage from a defined threat.

DATA SECURITY is the protection of data from loss, disclosure, modification and destruction.

SYSTEM INTEGRITY refers to the power functioning of hardware and programs, appropriate physical security and safety against external threats such as eavesdropping and wiretapping.

PRIVACY defines the rights of the user or organizations to determine what information they are willing to share with or accept from others and how the organization can be protected against unwelcome, unfair or excessive dissemination of information about it.

CONFIDENTIALITY is a special status given to sensitive information in a database to minimize the possible invasion of privacy. It is an attribute of information that characterizes its need for protection.

CODING

Add Doctor on Patient:

```
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;
}
}

```

DISCHARGE 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);
}
}

```

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; }
    }

```

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;
    }

```

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);
        return ds;
    }
}

```



OUTPUT

Untitled Page - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites Refresh Print View Source

Address http://localhost:2914/PatientMgmtSystem/Admin/frmAdminLogin.aspx Go Links



Patient Management System

LOGIN

User Name:



Password:

Untitled Page - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites Refresh Print Mail Address Book Favorites

Address http://localhost:2914/PatientMgmtSystem/Admin/frmAddDoctor.aspx Go Links

**Patient Management System**

Home | Add | Manage | Delete | Search | Logout

Doctor Addition

Please Enter The * Value

Doctor Code:

Doctor Name:

Specialist:

Available
Timing: From AM To AM

Contact No.:

Description:

Charges:

User Name:

Password:



Done Local intranet

Untitled Page - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites Refresh Print Mail Address Book Favorites

Address http://localhost:2914/PatientMgmtSystem/Admin/frmAddTest.aspx Go Links

**Patient Management System**

Home | Add | Manage | Delete | Search | Logout

Test Addition

Please Enter The * Value

Test Code:

Test
Name:

Description:

Charges:

Add Clear



Done Local intranet

Untitled Page - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites Refresh Print Mail Address Book Recent Tasks

Address http://localhost:2914/PatientMgmtSystem/Admin/frmAddRoom.aspx Go Links

**Patient Management System**

|| Home | Add | Manage | Delete | Search | Logout ||

Room Addition

Please Enter The * Value

Room Code.:

Room Name:

Description:

Charges:

Add Clear



Done Local intranet

Untitled Page - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites Refresh Print Mail Address Book Recent Tasks

Address http://localhost:2914/PatientMgmtSystem/Admin/frmAddMedicine.aspx Go Links

**Patient Management System**

|| Home | Add | Manage | Delete | Search | Logout ||

Medicine Addition

Please Enter The * Value

Medicine Code:

Medicine Name:

Prices:

Manufacture Date: 4/5/2005

Expiry Date: 4/5/2005

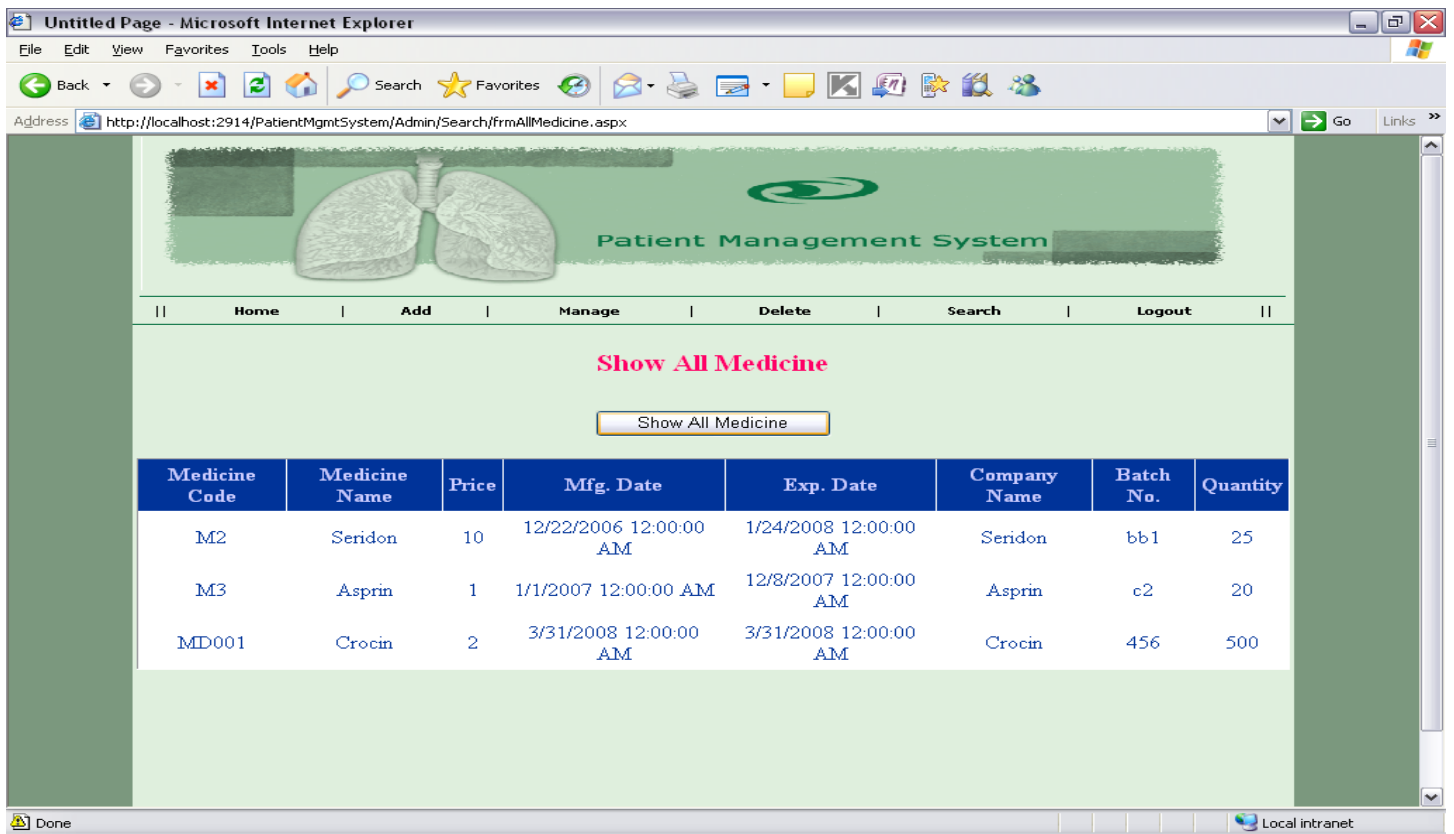
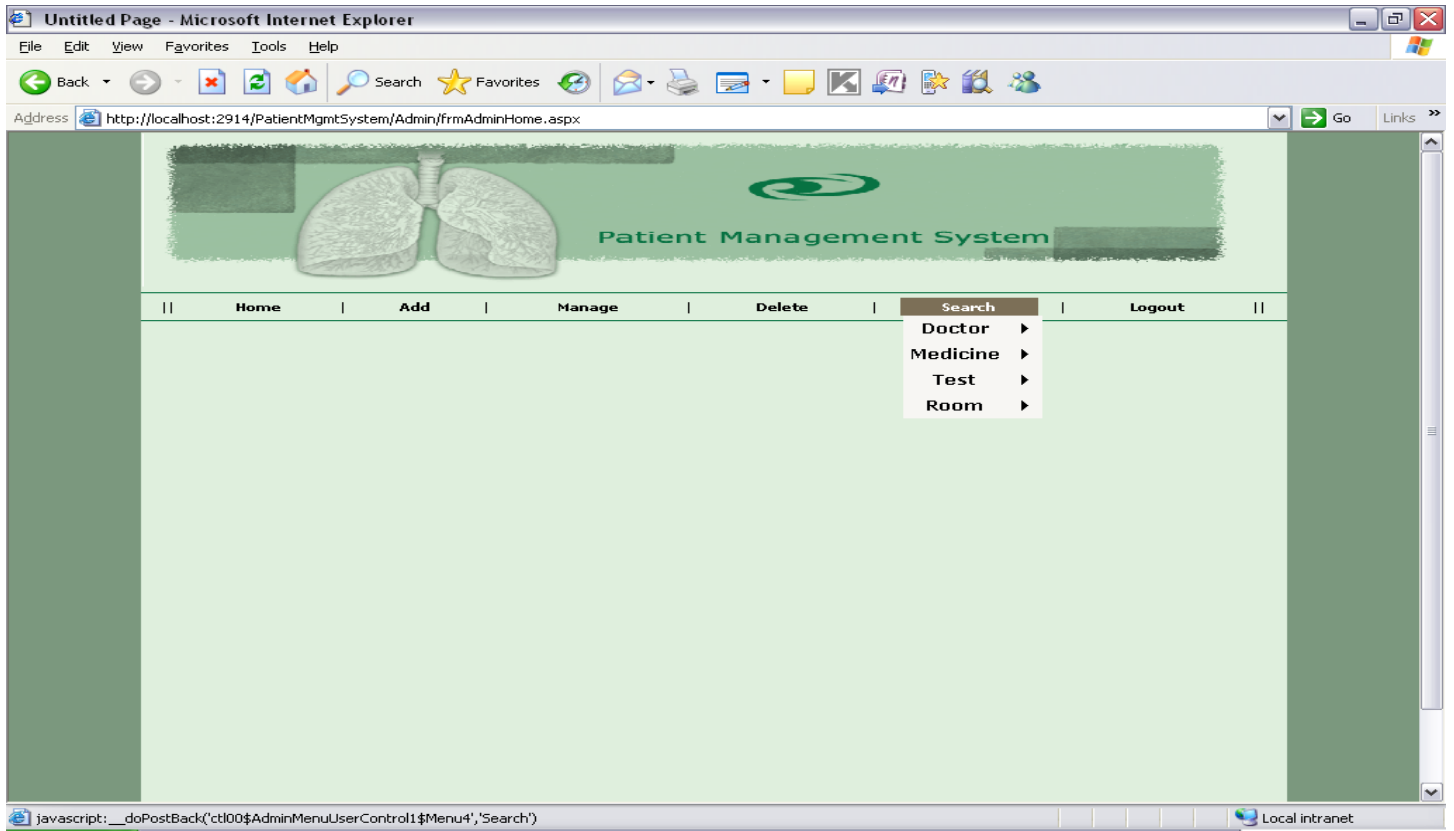
Company Name:

Batch No.:

Quantity:

Add Clear

Done, but with errors on page. Local intranet



Untitled Page - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites Refresh Print Mail Address Book Links

Address http://localhost:2914/PatientMgmtSystem/Employee/frnDischarge.aspx Go Links

Patient Discharge

Patient Code:	<input type="text" value="PTN13"/>	In/Out Patient	<input type="text" value="---Select---"/>
Patient Name:	<input type="text" value="Amir"/>	Date of Admit:	<input type="text" value="12/11/2007 12:00:00 AM"/>
Father/Husband Name:	<input type="text" value="Shakir"/>	Time of Admit:	<input type="text" value="09:41:45AM"/>
Chief Complain:	<input type="text" value="Osteoporesis"/>	Date of Discharge:	<input type="text" value="4/5/2005"/>
Sex:	<input type="text" value="Male"/>	Time of Discharge:	<input type="text" value="AM"/>
Address:	<input type="text" value="Ameerpet"/>	Day Stayed:	<input type="text" value="0"/>
Age:	<input type="text" value="15"/>	Advance:	<input type="text" value="1500"/>
Country:	<input type="text" value="India"/>	Doctor Charges:	<input type="text" value="500"/>
State:	<input type="text" value="AP"/>	Test Charges:	<input type="text" value="0"/>
City:	<input type="text" value="Hyderabad"/>	Room Charges:	<input type="text" value="0"/>
Doctor Name:	<input type="text" value="Arpit"/>	Medicine Charges:	<input type="text" value="0"/>
Room Name:	<input type="text"/>	Extra Charges:	<input type="text" value="0"/>
Patient Condition:	<input type="text"/>	Total Charges:	<input type="text" value="0"/>

CONCLUSION

It provides practical knowledge of not only programming in ASP.NET and VB.NET web based application and no some extent Windows Application and SQL Server, but also about all handling procedure related with Patient healthcare management system. It also provides knowledge about the latest technology used in developing web enabled application and client server technology that will be great demand in future. This will provide better opportunities and guidance in future in developing projects independently.

BENEFITS:

- This project offers user to enter the data through simple and interactive forms. This is very helpful for the client to enter the desired information through so much simplicity.
- The user is mainly more concerned about the validity of the data, whatever he is entering. There are checks on every stages of any new creation, data entry or updation so that the user cannot enter the invalid data, which can create problems at later date.
- Sometimes the user finds in the later stages of using project that he needs to update some of the information that he entered earlier. There are options for him by which he can update the records..
- User is provided the option of monitoring the records he entered earlier. He can see the desired records with the variety of options provided by him.
- From every part of the project the user is provided with the links through framing so that he can go from one option of the project to other as per the requirement. This is bound to be simple and very friendly as per the user is concerned.

LIMITATION:

- The size of the database increases day-by-day, increasing the load on the database back up and data maintenance activity.
- Training for simple computer operations is necessary for the users working on the system.

FUTURE ENHANCEMENT

- This System being web-based and an undertaking of Cyber Security Division, needs to be thoroughly tested to find out any security gaps.
- A console for the data centre may be made available to allow the personnel to monitor on the sites which were cleared for hosting during a particular period.
- Moreover, it is just a beginning; further the system may be utilized in various other types of auditing operation viz. Network auditing or similar process/workflow based applications...

REFERENCE

- www.support.microsoft.com
- www.developer.com
- www.15seconds.com
- www.msdn.microsoft.com
- www.asp.net
- www.aspfree.com
- www.msdn.microsoft.com/net/quickstart/aspplus/default.com