**CHAPTER NO. DESCRIPTIONS PAGE NO .**

## 1.) INTRODUCTION :

A Online Appointment System (**OLAS**) is an automated software system that manages the function of a health care. The presented project here is made in view overcoming the problems faced by many Hospitals regarding maintaining patient online Appointment, patient records, recording and keeping of Physician's information, Bed management, Report generation etc.

This project can be widely used in any Hospital which contains different departments with various employees (doctor) having different designations etc. **OLAS** not only provides an opportunity to the hospital to enhance their patient care but also can increase the profitability of the organization.

## 1.1) Background :

A good health is a place where Patients come up for general diseases. Online Appointment System provides facilities like:-

- Take online Appointment and Consultation by Doctors on Diseases.
- Diagnosis for diseases.
- Providing treatment facility.
- Facility for admitting Patients (providing beds, nursing, medicines etc.)
- Immunization for Patients/Children.

- **Various operational works that are done in a Online Appointment System are:-**
  - Recording information about the Patients that come.
  - Generating bills.
  - Recording information related to diagnosis given to Patients.
  - Keeping record of the Immunization provided to children/patients.
  - Keeping information about various diseases and medicines available to cure them.

These are the various jobs that need to be done in a Hospital by the operational staff and Doctors. All these works are done on papers.
The work is done as follows:-

- Information about Patients is done by just writing the Patients name, age and sex. Whenever the Patient comes up his information is stored freshly.
- Bills are generated by recording price for each facility provided to Patient on a separate sheet and at last they all are summed up.
- Diagnosis information to patients is generally recorded on the document, which contains Patient information. It is destroyed after some time period to decrease the paper load in the office.
-Immunization records of children are maintained in pre-formatted sheets, which are kept in a file.
- Information about various diseases is not kept as any document. Doctors

themselves do this job by remembering various medicines.

All this work is done manually by t he receptionist and other operational staff and lot of papers are needed to be handled and taken care of Doctors have to remember various medicines available for diagnosis and sometimes miss better alternatives as they can't remember them at that time.

## 1.2) OBJECTIVES :

- Generating a unique user/patient enquiry number to each user/patient enquiry.
- This application also maintains all records related to patient, and its Test report.
- To give all the permissions to users like add, modify, delete, restore data and data backup by the administrator.
- **Planned approach towards working: -** The working in the organization will be well planned and organized. The data will be stored properly in data stores, which will help in retrieval of information as well as its storage.
- **Accuracy: -** The level of accuracy in the proposed system will be higher. All operation would be done correctly and it ensures that whatever information is coming from the center is accurate.
- **Reliability:** - The reliability of the proposed system will be high due to the above stated reasons. The reason for the increased reliability of the system is that now there would be proper storage of information.
- **No Redundancy: -** In the proposed system utmost care would be that no information is repeated anywhere, in storage or otherwise. This would assure economic use of storage space and consistency in the data stored.
- **Immediate retrieval of Information: -** The main objective of proposed system is to provide for a quick and efficient retrieval of information. Any type of information would be available whenever the user/patient requires.

- **Immediate storage of information: -** In manual system there are many problems to store the largest amount of information.
- **Easy to Operate: -** The system should be easy to operate and should be such that it can be developed within a short period of time and fit in the limited budget of the user/patient.

## 1.3) Purpose, Scope, and Applicability :

### 1.3.1 Purpose :

• The Software is for the automation of Online Appointment System.

• It maintains two levels of users :-
- -Administrator Level.
- - User/Patient Level.

• The Software includes:-
- - Maintaining Patient details.
- - Providing Online Appointment Prescription, Precautions and Diet advice.
- - Providing and maintaining all kinds of tests for a patient.
- - Billing and Report generation.

### 1.3.2 Scope :

It can be used in any Hospital, Clinic, Dispensary or Pathology labs for maintaining patient details, Online Appointment and their test results.

### 1.3.3 Applicability :

This is an online project so the user can be from any part of the world.

## 2) SURVEY OF TECHNOLOGIES :

**Web Application**

A web application or "web app" is a software program that runs on a web server. Unlike traditional desktop applications, which are launched by your operating system, web apps must be accessed through a web browser. Web apps have several advantages over desktop application. Since they run inside web browsers, developers do not need to develop web apps for multiple platforms.

**ASP .NET**

ASP.NET is a unified Web development model that includes the services necessary for you to build enterprise-class Web applications with a minimum of coding. ASP.NET is part of the .NET Framework, and when coding ASP.NET applications you have access to classes in the .NET Framework. You can code your applications in any language compatible with the common language runtime (CLR), including Microsoft Visual Basic and C#.

**C#**

C# (pronounced "C-sharp") is an object-oriented programming language

from Microsoft that aims to combine the computing power of C++ with the programming ease of Visual Basic. C# is based on C++ and contains features similar to those of Java. C# is designed to work with Microsoft's .Net platform.

**SQL Server**

SQL Server is a Microsoft product used to manage and store information. Technically, SQL Server is a "relational database management system" (RDMS). Broken apart, this term means two things. First, that data stored inside SQL Server will be housed in a "relational database", and second, that SQL Server is an entire "management system", not just a database. SQL Server runs on T-SQL (Transact -SQL), a set of programming extension from Sybase and Microsoft that add several features to standard SQL, including transaction control, exception and error handling, row processing, and declared variable.

# 3) REQUIREMENTS AND ANALYSIS

## 3.1) Problem Definition :

Problems with conventional system

- **Lack of immediate retrievals: -**The information is very difficult to retrieve and to find particular information like- E.g. - To find out about the patient's history, the user has to go through various registers. This results in inconvenience and wastage of time.
- **Lack of immediate information storage: -** The information generated by various transactions takes time and efforts to be stored at right place.
- **Lack of prompt updating: -** Various changes to information like patient details or immunization details of child are difficult to make as paper work is involved.

- **Error prone manual calculation: -** Manual calculations are error prone and take a lot of time this may result in incorrect information. For example calculation of patient's bill based on various treatments.
- **Preparation of accurate and prompt reports: -** This becomes a difficult task as information is difficult to collect from various registers.

## 3.2) Requirements Specification :

**About the Organization**

A Hospital normally contains information about patient. But a ONLINE APPOINTMENT SYSTEM normally contains the following parts:-

**Patient Registration**

Patient registration means capturing information of all the patients visiting the hospital and provides record of all the registered patients through a unique patient identification number or room number.

**View details of patients**

It views the details of existing patients by registration number.

**Searching process**

Searches the details of patients by city or by blood group etc.

**Delete record:**

Deletes the record of patients by registration number.

- **PROJECT RECORD MAINTENANCE**

To maintain the proper details of hospital management system provides the following : Registration, list of patients, adding doctors, record of patients, modify details, discharge of patients so on under various provision provided by the management.

**Our Vision :**

We shall define ourselves in the cutting edge technology in the coming era. We shall create honest working environment with see-through-glass planning.

**Our Mission :**

To create opportunity for growth & self actualization and provide an environment of highly conducive works culture

## 3.3) Planning and Scheduling :

**Gantt Chart or Time Line Chart** : A Gantt Chart Can be develop for the entire project or a separate chart can be developed for each function. A tabular form is maintained where rows indicate the task with milestone and columns indicate the task duration(Months). The horizontal bars that spans across columns indicate duration of the task. The circle indicate the milestones.

| Tasks | Month1 | Month2 | Month3 | Month4 |
|---|---|---|---|---|
| **A**<br>• **a1**<br>• **a2**<br>**Milestone** | | | | |
| **B**<br>• **b1**<br>• **b2**<br>**Milestone** | | | | |
| **C**<br>• **c1**<br>**Milestone** | | | | |
| **D**<br>• **d1**<br>• **d2**<br>**Milestone** | | | | |

**PERT CHART**

| Project Phases | Date | Activity |
|---|---|---|
| Feasibility Study | 14th july(7 Days) | A |
| System Analysis | 21th July(9 Days) | B |
| System Design | 30th July(5 Days) | C |
| Database Design | 5th August (4 Days) | D |
| Coding & Design | 10th August (30 Days) | E |
| System Integration | 10th Sept.(10 Days) | F |
| System Testing | 20th Sept.(20 Days) | G |
| System Implementation | 9th Oct(12 Days) | H |
| User Training | 21th OCt.(10 Days) | I |
| Post-Implementation Review | 1th Nov(10 Days) | J |
| **Total Days** | **117 Days** | |

# 3.4) Software and Hardware Requirements :

**Software Requirement:**

Front End : Asp.Net MVC, Microsoft Visual Studio 2019

Back End : SQL Server 2012

OS : windows 10

**Hardware Requirement**

CPU : Intel Core I3-4005u CPU @ 1.70GHZ

RAM : 128 MB

Hard disk : 500 GB

Equipment : Mouse, Keyboard, Monitor

## 3.5) Preliminary Product Description

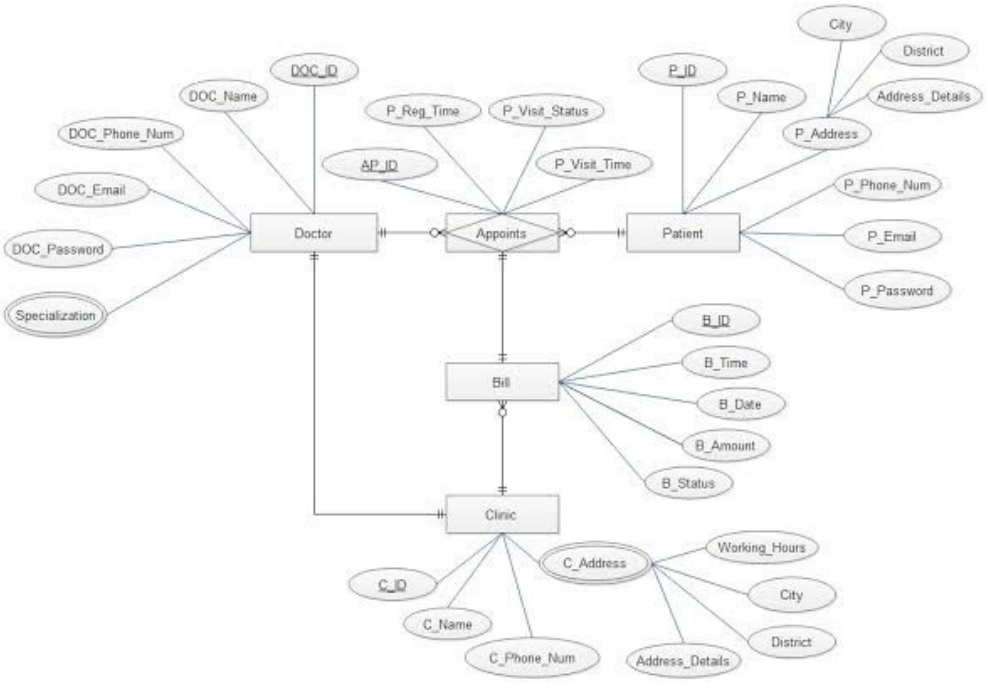
- **FEASIBILITY STUDY :-**

After request clarification the feasibility study is performed. In actually when we are performing feasible study ,we must take care of three types of feasible study. These three feasibilities are as follows

- **TECHNICAL FEASIBILITY :-** In this feasibility test, we check out that , can the work for the project be done with current equipment ? , Existing software technology & by available personnel.?
- **ECONOMIC FEASIBILITY: -** Are there sufficient benefits in creating the system to make the cost acceptable .?
- **OPREATION FEASIBILITY** :- Will the system be used if it is developed and implemented .?

## 3.6) Conceptual Model:

### 3.6.1 ER Diagram :-

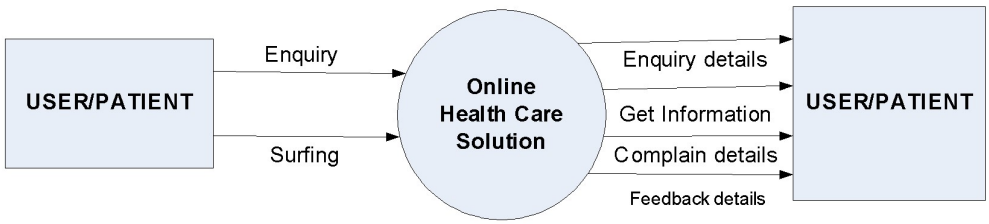



### 3.6.2 DFD Diagram:-

# Data Flow Diagrams

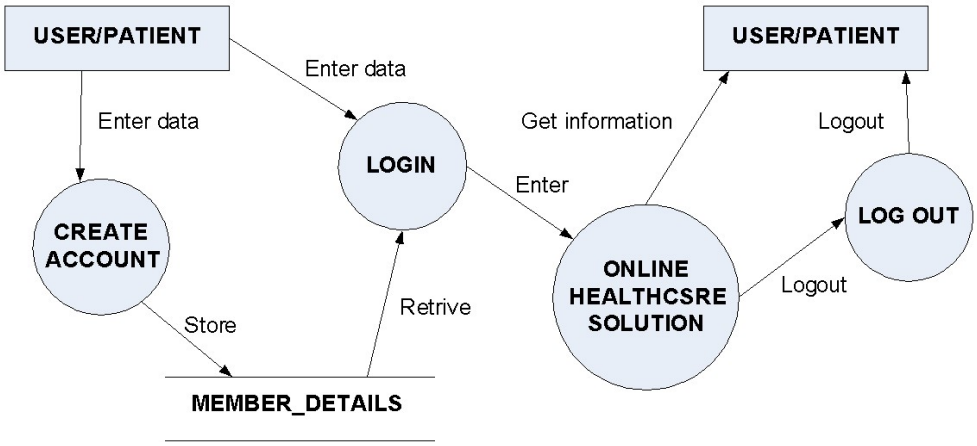
**"A data flow diagram is a graph which shows the flow of data values from their sources in objects through process that transform them to their destination in other object."**

**0-Level DFD**

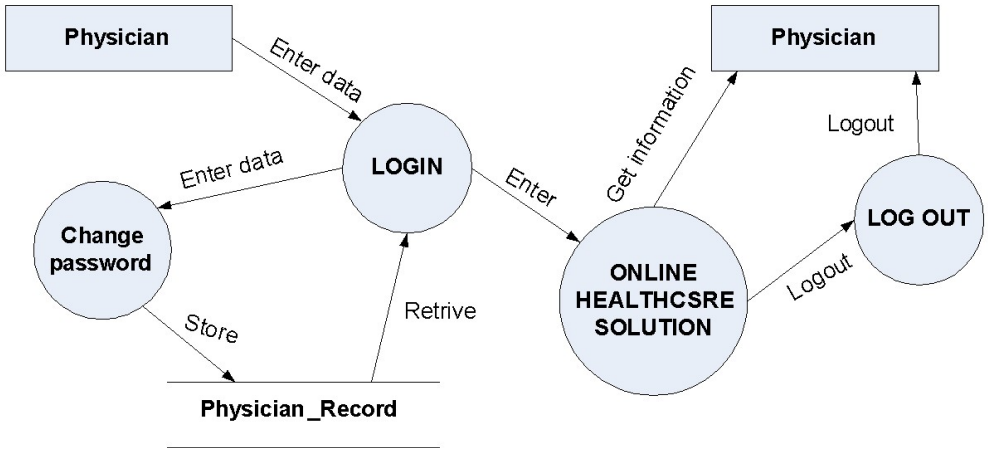


## 1st - Level DFD

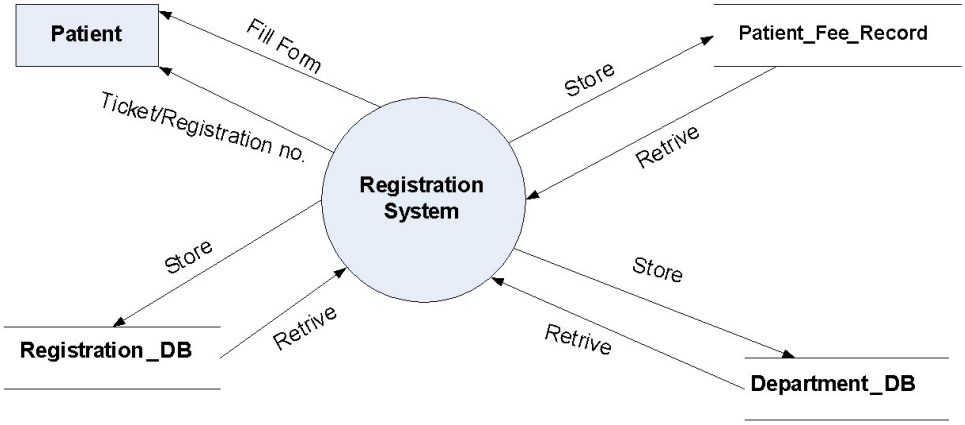
### Usear Module :




### PHYSICIAN MODULE

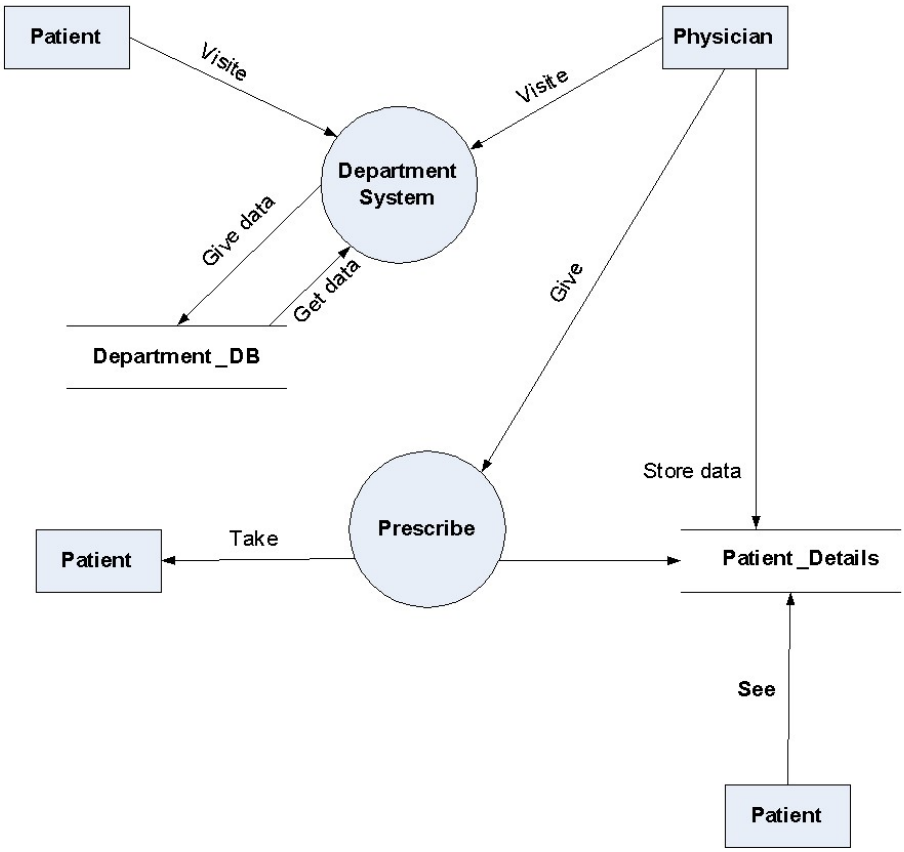


## REGISTRATION SYSTEM MODULE




## DEPARTMENT SYSTEM MODULE

### PATIENT DETAILS MODULE



# 4) SYSTEM DESIGN :

## 4.1) BASIC MODULES

**In this project we have taken some modules:**

### USER

- Login Module
- New User Entry Module
- Change Password Module
- Patient Registration Module
- Appointment Module

### ADMINISTRATOR

- Login Module
- Change Password Module
- Physician Module
- physician Attendance
- Physician Salary Module
- TEST Module
- Hospital Module
- User Permission Module
- **Login Module:** This is a very first module in this project; a person can want to enter into this software project he/she must have pass through this module,

in other word we can say that this is Authentication module because the users authenticate with his/her **password** and **username**.

- **New User Module:** In this module we can create new user his/her username, password and his rights as our choice.

- **Change Password Module -**This module is used to change the user password. To change password of a user select login of user and type password in password field. After typing new password, typed re password for checking new password and re password respectively. Click Apply to conform and after this click Ok button for change password.

- **Patient Registration Module:** This is customer registration module, this module contains the whole information about the customer registration like registration no, name, address, phone no., age, sex, diagnosis, etc . This record will be store in the patient registration table.

- **Appointment Module:** This is customer appointment module, this module contains the information about the customer registration no. like registration Id, name, address, phone no., age, sex, diagnosis, etc . This record will be store in the patient registration table.

- **Physician Module:** This is Physician module, it contains the whole information about the physician like his name, designation, salary, date of joining, date of birth, specialization etc.

- **Physician Attendance Module:** This is Physician attendance module, this module contains the whole information about the physician attendance like his/her attending days, absent days, month etc.
- **Physician Salary:** This is physician salary module, contains information about the physician salary, when we generate his/her salary data will retrieve from physician attendance module and calculate his/her salary.
- **Hospital Module:** -This module is use to store the information of about Hospital.
- **Test Module :-** This module is to store use test name and its fee.
- **User Permission Module : -**This module is use to giving permission to the users. To give permission Select the login id and check the permission and click apply

# 4.2) Procedural Design

## 4.2.1) Data Structures (TABLES)

### 1. User Login :

| Field Name | DataType | Length | Constraint | Description |
|---|---|---|---|---|
| User_Name | Varchar | 255 | NOT NULL | Users Name |
| User-Pass | Varchar | 55 | NOT NULL | Users Password |

### 2. Administrator Login:

| Field Name | DataType | Length | Constraint | Description |
|---|---|---|---|---|
| Admin_Name | Varchar | 255 | NOT NULL | Administrator Name |
| Password | Varchar | 55 | NOT NULL | Administrator Password |

## 3. New_Account_Form :

| Field Name | DataType | Length | Constraint | Description |
|---|---|---|---|---|
| Fname | Varchar | 55 | Not Null | Fast name |
| lname | Char | 55 | Null | Last Name |
| loginid | Varchar | 55 | Not Null | Login id |
| password | Varchar | 8 | Not null | Password |
| Db | Varchar | 10 | Not null | Date of birth |
| Gender | Char | 5 | Not Null | Gender |
| Rtype | Char | 10 | Not null | Registration Type |
| Address | Varchar | 55 | Not Null | Address |
| city | Varchar | 15 | Not Null | City |
| State | Varchar | 15 | Not Null | State |
| Country | Varchar | 15 | Not Null | Country |
| Phone | numeric | 12 | Not Null | Phone numer |
| Emailid | Varchar | 55 | null | E_mail ID |

## 4. Patient Registration Form

| Field Name | DataType | Length | Constraint | Description |
|---|---|---|---|---|
| P_Name | Char | 55 | NOT NULL | Patient Name |

| | | | | |
|---|---|---|---|---|
| P_Age | Numeric | 3 | NOT NULL | Patient Age |
| P_Sex | Char | 1 | NOT NULL | Patient Gender |
| P_Add | Varchar | 55 | NOT NULL | Patient Address |
| P_Phone | Numeric | 10 | NULL | Patient Phone Number |
| P_Email_ID | Varchar | 55 | NULL | Patient Email ID |
| P_Pro_Diag | Varchar | 25 | NOT NULL | Patient Provisional Diagnosis |

## 5. Physician_ Details :

| Field Name | DataType | Length | Constraint | Description |
|---|---|---|---|---|
| Phy_ID | Varchar | 10 | Primary Key | Physician ID |
| Phy_Name | Char | 30 | NOT NULL | Physician Name |
| Phy_Sex | Char | 1 | NOT NULL | Physician Gender |
| Phy_Phone | Number | 15 | NULL | Physician Phone Number |
| Phy_Email_ID | Varchar | 55 | NULL | Physician Email ID |
| Phy_Add | Varchar | 55 | NULL | Physician Address |
| Phy_W_Time | Times | 10 | NOT NULL | Physician Working Time |
| Phy_W_Day | Date | 55 | NOT NULL | Physician Working Time |
| Phy_ Dept | Char | 20 | NOT NULL | Physician Department |

## 6. Hospital Details :

| Field Name | DataType | Length | Constraint | Description |
|---|---|---|---|---|
| H_ID | Varchar | 10 | Primary Key | Hospital ID |
| H_Name | Char | 25 | NOT NULL | Hospital Name |
| H_Loc | Varchar | 55 | NOT NULL | Hospital Location |
| H_city | Char | 15 | NOT NULL | Hospital City |
| H_Phone | Numeric | 15 | NOT NULL | Hospital Phone |
| H_Email | Varchar | 55 | NULL | Hospital Email ID |
| H_Web | Varchar | 55 | NULL | Hospital Website |
| H_Spe | Char | 15 | NOT NULL | Hopital Specialization |

## 7. Test_Report_Details :

| Field Name | DataType | Length | Constraint | Description |
|---|---|---|---|---|
| P_ID | Varchar | 25 | Foreign Key | Patien ID |
| T_ID | Varchar | 25 | Primary Key | Test ID |
| T_Name | Varchar | 25 | NOT NULL | Test Name |
| Test_Date | Date | 10 | NOT NULL | Test Date |
| T_Time | time | 10 | NOT NULL | Test Time |
| T_Result | Varchar | 15 | NOT NULL | Test Result |

## 8. Patient_ Fee_Details :

| Field Name | DataType | Length | Constraint | Description |
|---|---|---|---|---|
| P_ID | Varchar | 25 | Foreign Key | Patient ID |
| P_Fee | Numeric | | NULL | Patient Fee |

### 4.2.2) System Security Management :

- Since this application is being prepared for Internet purpose security plays a crucial role for successful running of the website. For the same **windows form authentication mode** is used to provide the security. Every unauthorized user is being prohibited.
- Special care has been taken for admin panel. Even after login form the subsequent sections are also being **checked using session.**
- **Double Check Security** mechanism has also been taken for security purpose. The admin will again enter password after entering username and password for logging in.
- Another security measure I took in this project is **Username** and **Password** provide to selected User/Patient so that no unidentified person can not access the system.

# 5. Coding

***Home Controller :-***

```
using System;

using System.Collections.Generic;

using System.Linq;
```

```
using System.Web;
using System.Web.Mvc;
using Entity_OAPS.Model;


namespace Entity_OAPS.Controllers
{
   public class HomeController : Controller
   {
      Entity_OAPSEntities1 db = new Entity_OAPSEntities1();
      public ActionResult HomeIndex( )
      {
         return View();
      }
   }
```

## *Home cshtml:-*

```
@{
    ViewBag.Title = "HomeIndex";
}
```

## *Patient Controller :-*

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Entity_OAPS.Model;

namespace Entity_OAPS.Controllers
{
   public class PatientDetailController : Controller
   {
      Entity_OAPSEntities1 db = new Entity_OAPSEntities1();
      public ActionResult PatientDetailIndex()
      {
         return View();
      }
      public void Insert(PatientDetail obj)
      {
         db.PatientDetails.Add(obj);
         db.SaveChanges();
      }
   }
}
```

## *Patient cshtml:-*

```
@{
    ViewBag.Title = "PatientDetailIndex";
}
```

```
<script src="~/jquery.min.js"></script>

<script src="~/PatientDetail.js"></script>


<table style="background-color:cyan; color:black; width:30%">

    <tr>

        <td>

            <h2>Patient Record</h2>

        </td>

    </tr>

</table>


<table style="background-color:pink; color:blue">

    <tr>

        <td>Patient Fname:</td>

        <td><input type="text" id="txtfname" /></td>

    </tr>

    <tr>

        <td>Patient Lname:</td>

        <td><input type="text" id="txtlname" /></td>

    </tr>

    <tr>

        <td>Patient Age:</td>

        <td><input type="text" id="txtage" /></td>

    </tr>

    <tr>

        <td>Patient Gender :</td>
```

```
    <td>
        <input type="radio" name="Gender" value="1" />Male
        <input type="radio" name="Gender" value="2" />Female
        <input type="radio" name="Gender" value="3" />Other
    </td>
</tr>
<tr>
    <td>Patient Address:</td>
    <td><input type="text" id="txtaddress" /></td>
</tr>
<tr>
    <td>Patient Phone:</td>
    <td><input type="text" id="txtphone" /></td>
</tr>
<tr>
    <td>Patient Email :</td>
    <td>
        <input type="text" id="txtemail" />
    </td>
<tr>
<tr>
    <td>Patient Current Diagnosis :</td>
    <td>
        <input type="text" id="txtdiagnosis" />
    </td>
<tr>
<tr>
```

```
        <td>Patient Blood Group  :</td>
        <td>
            <input type="text" id="txtgroup" />
        </td>
    <tr>
    <tr>
        <td>Patient Bed Number:</td>
        <td><input type="text" id="txtbed" /></td>
    </tr>
    <tr>
        <td>Patient Department:</td>
        <td><input type="text" id="txtdpt" /></td>
    </tr>
    <tr>
        <td>Patient Hospital Name:</td>
        <td><input type="text" id="txthospital" /></td>
    </tr>
    <tr>
        <td></td>
        <td><input type="button" id="btnpatient" value="Patient"
onclick="PatientDataData()" /></td>
    </tr>
</table>
```

## *Patient JavaScript:-*

```
function Clear() {
```

```
    $("#txtfname").val("");

    $("#txtlname").val("");

    $("#txtage").val("");

    $("input:radio").attr("checked", false);

    $("#txtaddress").val("");

    $("#txtphone").val("");

    $("#txtemail").val("");

    $("#txtdiagnosis").val("");

    $("#txtdtxtgroup").val("");

    $("#txtbed").val("");

    $("#txtdpt").val("");

    $("#txthospital").val("");

    $("#btnpatient").val("Patient");

}


function PatientDataData() {

    $.ajax({

        url: '../PatientDetail/Insert',

        data:      {      p_fname:      $("#txtfname").val(),      p_lname:
$("#txtlname").val(),       p_age:       $("#txtage").val(),       p_gender:
$('input:radio[name=Gender]:checked').val(),                      p_address:
$("#txtaddress").val(),     p_phone:     $("#txtphone").val(),     p_email:
$("#txtemail").val(),     p_current_diagnosis:     $("#txtdiagnosis").val(),
p_blood_group:  $("#txtgroup").val(),  p_bed_no:  $("#txtbed").val(),  p_dept:
$("#txtdpt").val(), p_hopital_name: $("#txthospital").val() },

        success: function () {

            alert("Patient Data successfull insert!");

            Clear();

        },

        error: function () {
```

```
            alert("Patient Data not insert");
        }
    });
}
```

## *Hospital Controller:-*

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Entity_OAPS.Model;

namespace Entity_OAPS.Controllers
{
   public class HospitalDetailController : Controller
   {
      Entity_OAPSEntities1 db = new Entity_OAPSEntities1();
      public ActionResult HospitalDetailIndex()
      {
         return View();
      }
      public void Insert(HospitalDetail obj)
      {
         db.HospitalDetails.Add(obj);
```

```
        db.SaveChanges();
    }
    public JsonResult DisplayCity()
    {
        var data = (from a in db.tblCities select a).ToList();
        return Json(data, JsonRequestBehavior.AllowGet);
    }
  }
}
```

## *Hospital cshtml:-*

```
@{
    ViewBag.Title = "HospitalDetailIndex";
}

    <script src="~/jquery.min.js"></script>
    <script src="~/HospitalDetail.js"></script>

<table style="background-color:cyan; color:black; width:23%">
    <tr>
        <td colspan="2">
            <h2>Hospital Record</h2>
        </td>
    </tr>
</table>
```

```
<table style="background-color:blue; color:white">
    <tr>
        <td>Hospital Name:</td>
        <td><input type="text" id="txtname" placeholder="Hospital Name" /></td>
    </tr>
    <tr>
        <td>Hospital City:</td>
        <td><select id="ddlcity">
            <option value="0">SELECT</option>
            </select></td>
    </tr>
    <tr>
        <td>Hospital Phone:</td>
        <td><input type="text" id="txtphone" placeholder="123-456-789" /></td>
    </tr>
    <tr>
        <td></td>
        <td><input type="button" id="btnhospital" value="Hospital" onclick="HospitalDetailData()" /></td>
    </tr>
</table>
```

## *Hospital JavaScript:-*

```
$(document).ready(function () {
```

```
    BindCity();
});
function Clear() {
    $("#txtname").val("");
    $("#ddlcity").val("0");
    $("#txtphone").val("");
    $("#btnhospital").val("Hospital");
}
function HospitalDetailData() {
    $.ajax({
        url: '../HospitalDetail/Insert',
        data: { H_name: $("#txtname").val(), H_city: $("#ddlcity").val(),
H_phone: $("#txtphone").val() },
        success: function () {
            alert("Hospita Data successfull insert!");
            Clear();
        },
        error: function () {
            alert("Hospital Data not insert");
        }
    });
}
function BindCity() {
    $.ajax({
        url: '../HospitalDetail/DisplayCity',
        type: 'post',
        data: {},
        async: false,
```

```
        success: function (data) {
            for (var i = 0; i < data.length; i++) {
                $("#ddlcity").append($('<option/>').attr("value",
data[i].cid).text(data[i].cname));
            }
        },
        error: function () {
            alert("City not found!");
        }
    });
}
```

## *Aboutus Controller:-*

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Entity_OAPS.Model;

namespace Entity_OAPS.Controllers
{
   public class AboutusController : Controller
   {
      Entity_OAPSEntities1 db = new Entity_OAPSEntities1();
```

```
        public ActionResult AboutusIndex()
        {
            return View();
        }
    }
}
```

## *Aboutus cshtml:-*

```
@{
    ViewBag.Title = "AboutusIndex";
}

<table>
    <tr>
        <td colspan="2" style="color:red">
            <h2><u>Aboutus>></u></h2>
        </td>
    </tr>
</table>
<table style="background-color:deeppink; color:yellow">
    <tr>
        <td colspan="2">
            <b>
                <i>
                        150 bed multi super specialty hospital with
comprehensive services under one roof
```

Conveniently located on turner road near to ISBT

Comprehensive and committed team of qualified doctors, staff nurses and paramedical

Centrally air conditioned building with all safety features as per NBC norms

100% power backup, independent water supply, security and access control

Clean Drinking Water RO water plant

In-house kitchen with monitored dietary norms.

Advance Hospital Information System for co-ordinated care.

Round the clock Trauma & Emergency Team.

Emergency OT, plaster unit room, trauma bay and cardiac emergency unit.

Equipped with cardiac monitors, Ventilators, Mobile X-Ray unit, ABG and other instruments.

Inhouse Radiology, pathology and pharmacy

Mobile ICU ambulances with life support services.

Advance security and safety measures with surveillance cameras & fire protection system.

Adequate power backup with contingencies with supplement water supply.

Respecting environment by deployment of STP & ETP with Biomedical Waste Management norms.

Centralized and piped medical gases with anesthesia gas scavenging system.

24×7 Ambulance, Pharmacy and Cafeteria services.

</i>

</b>

</td>

</tr>

```
</table>
```

## *Registration Controller:-*

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Entity_OAPS.Model;

namespace Entity_OAPS.Controllers
{
   public class RegController : Controller
   {
      Entity_OAPSEntities1 db = new Entity_OAPSEntities1();
      public ActionResult RegIndex( )
      {
         return View();
      }
      public void Insert(tblRegistration obj)
      {
         db.tblRegistrations.Add(obj);
         db.SaveChanges();
      }
   }
}
```

## *Registration cshtml:-*

```
@{
    ViewBag.Title = "RegIndex";
}

<script src="~/jquery.min.js"></script>
<script src="~/Registration.js"></script>
<script src="~/RegValidation.js"></script>

<table style="background-color:cyan; color:black; width:30%">
    <tr>
        <td>
            <h2>Registration Form</h2>
        </td>
    </tr>
</table>
<table style="background-color:gray; color:yellow">
    <tr>
        <td>Patient Fname:</td>
        <td><input type="text" id="txtfname" /></td>
    </tr>
    <tr>
        <td>Patient Lname:</td>
        <td><input type="text" id="txtlname" /></td>
    </tr>
```

```
<tr>
    <td>Patient Age:</td>
    <td><input type="text" id="txtage" /></td>
</tr>
<tr>
    <td>Patient Gender :</td>
    <td>
        <input type="radio" name="Gender" value="1" />Male
        <input type="radio" name="Gender" value="2" />Female
        <input type="radio" name="Gender" value="3" />Other
    </td>
</tr>
<tr>
    <td>Patient Address:</td>
    <td><input type="text" id="txtaddress" /></td>
</tr>
<tr>
    <td>Patient Phone:</td>
    <td><input type="text" id="txtphone" /></td>
</tr>
<tr>
    <td>Patient Email :</td>
    <td>
        <input type="text" id="txtemail" />
    </td>
</tr>
<tr>
```

```
        <td>Patient Current Diagnosis :</td>
        <td>
            <input type="text" id="txtdiagnosis" />
        </td>
    </tr>
    <tr>
        <td>Password :</td>
        <td>
            <input type="text" id="txtpwd" />
        </td>
    </tr>
    <tr>
        <td>Confirm Password :</td>
        <td>
            <input type="password" id="txtconpwd" />
        </td>
    </tr>
    <tr>
        <td></td>
        <td><input   type="button"   id="btnregistration"   value="Register"
onclick="return Validation()" /></td>
    </tr>
</table>
```

## *Registration JavaScript:-*

```
function Clear() {
```

```
    $("#txtfname").val("");

    $("#txtlname").val("");

    $("#txtage").val("");

    $("input:radio").attr("checked", false);

    $("#txtaddress").val("");

    $("#txtphone").val("");

    $("#txtemail").val("");

    $("#txtdiagnosis").val("");

    $("#txtpwd").val("");

    $("#txtconpwd").val("");

    $("#btnregistration").val("Registered");

}


function RagisterdData() {

    $.ajax({

        url: '../Reg/Insert',

     data: { r_fname: $("#txtfname").val(), r_lname: $("#txtlname").val(),
r_age:                    $("#txtage").val(),                    r_gender:
$('input:radio[name=Gender]:checked').val(),                    r_address:
$("#txtaddress").val(),     r_phone:     $("#txtphone").val(),     r_email:
$("#txtemail").val(),     r_current_diagnosis:     $("#txtdiagnosis").val(),
r_password: $("#txtpwd").val(), r_cpassword: $("#txtconpwd").val() },

        success: function () {

            alert("Registration is successfull !");

            Clear();

        },

        error: function () {

            alert("Registration fail!!");

        }
```

```
    });
}
```

## *Registration Validation:-*

```
function Validation() {
    var Dabba = "";

    Dabba += checkfname();
    Dabba += checklname();
    Dabba += checkage();
    Dabba += checkaddress();
    Dabba += checkphone();
    Dabba += checkemail();
    Dabba += checkdiagnosis();
    Dabba += checkpaw();

    if (Dabba != "") {
        alert(Dabba);
        return false;
    }
    else {
        RagisterdData();
    }
}
function checkfname() {
    var TB = $("#txtfname");
```

```
    var Exp = /^[a-z A-Z]+$/
    if (TB.val() == "") {
        return "Please Enter Your Fname!!\n";
    }
    else if (!Exp.test(TB.val())) {
        return "Please Enter Only In Alphabate!!\n";
    }
    else {
        return "";
    }
}
function checklname() {
    var TB = $("#txtlname");
    var Exp = /^[a-z A-Z]+$/
    if (TB.val() == "") {
        return "Please Enter Your Lname!!\n";
    }
    else if (!Exp.test(TB.val())) {
        return "Please Enter Only In Alphabate!!\n";
    }
    else {
        return "";
    }
}
function checkage() {
    var TB = $("#txtage");
    var Exp = /^[0-9]+$/
```

```
    if (TB.val() == "") {
        return "Please Enter Your age!!\n";
    }
    else if (!Exp.test(TB.val())) {
        return "Please Enter Only In numerical!!\n";
    }
    else {
        return "";
    }
}
function checkaddress() {
    var TB = $("#txtaddress");
    if (TB.val() == "") {
        return "Please enter the address!!\n";
    }
    else {
        return "";
    }
}
function checkphone() {
    var TB = $("#txtphone");
    if (TB.val() == "") {
        return "Please enter the phone number!!\n";
    }
    else {
        return "";
    }
```

```
}
function checkemail() {
    var TB = $("#txtemail");
    var Exp = /^([a-zA-Z0-9_.-])+@(([a-zA-Z0-9-])+.)+([a-zA-Z0-9]{2,4})+$/;
    if (TB.val() == "") {
        return 'Please enter your email\n'
    }
    else if (!Exp.test(TB.val())) {
        return "Please Enter valid email!!\n";
    }
    else {
        return "";
    }
}
function checkdiagnosis() {
    var TB = $("#txtdiagnosis");
    if (TB.val() == "") {
        return "Please enter Your diagnosis!!\n";
    }
    else {
        return "";
    }
}
function checkpaw() {
    var TB1 = $("#txtpwd");
    var TB2 = $("#txtconpwd");
    if (TB1.val() == "") {
```

```
        return "Please enter your password\n";
    }
    else if (TB2.val() == "") {
        return "Please enter your confirm password\n";
    }
    else if (TB1.val() != TB2.val()) {
        return "Password did not matched\n";
    }
    else {
        return "";
    }
}
```

## ***Login Controller:-***

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Entity_OAPS.Model;

namespace Entity_OAPS.Controllers
{
   public class LoginController : Controller
   {
      Entity_OAPSEntities1 db = new Entity_OAPSEntities1();
```

```
        public ActionResult LoginIndex()
        {
            return View();
        }
        public JsonResult LoginUser(tblRegistration obj)
        {
            var data = (from x in db.tblRegistrations where x.r_email == obj.r_email && x.r_password == obj.r_password select x).ToList();
            return Json(data, JsonRequestBehavior.AllowGet);
        }
    }
}
```

## *Login cshtml:-*

```
@{
    ViewBag.Title = "LoginIndex";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
    <script src="~/jquery.min.js"></script>
    <script src="~/Login.js"></script>
    <script src="~/LoginValidation.js"></script>

    <table style="background-color:deepskyblue; color:yellow">
        <tr>
            <td>Email :</td>
            <td>
```

```
                <input type="text" id="txtemail" placeholder="Email" />
            </td>
        <tr>
        <tr>
            <td>Password :</td>
            <td>
                <input type="password" id="txtpassword"
placeholder="Password" />
            </td>
        </tr>
        <tr>
            <td></td>
            <td>
                <input type="button" id="btnlogin" value="Login"
onclick="return LValidation()" />
                <a href="../Reg/RegIndex"><input type="button"
id="btnsignup" value="SignUp" onclick="Signup()" /></a>
            </td>
        </tr>
    </table>
```

## *Login JavaScript:-*

```
function Login() {
    $.ajax({
        url: '../Login/LoginUser',
        data: { r_email: $("#txtemail").val(), r_password:
$("#txtpassword").val() },
```

```
        success: function (data) {
            if (data.length > 0) {
                window.location.href = "../ShowReg/RegShowIndex?QS=" + data[0].r_id;
            }
        },
        error: function () {
            alert("Login  fail!");
        }
    });
}
```

## *Login Validation:-*

```
function LValidation() {
    var Dabba = "";

    Dabba += checkemail();
    Dabba += checkpaw();

    if (Dabba != "") {
        alert(Dabba);
        return false;
    }
    else {
        Login();
    }
```

```
}
function checkemail() {
    var TB = $("#txtemail");
    var Exp = /^([a-zA-Z0-9_.-])+@(([a-zA-Z0-9-])+.)+([a-zA-Z0-9]{2,4})+$/;
    if (TB.val() == "") {
        return 'Please enter your email\n'
    }
    else if (!Exp.test(TB.val())) {
        return "Please Enter valid email!!\n";
    }
    else {
        return "";
    }
}
function checkpaw() {
    var TB = $("#txtpassword");
    if (TB.val() == "") {
        return "Please enter your password\n";
    }
    else {
        return "";
    }
}
```

## ***Appointment Controller:-***

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Entity_OAPS.Model;

namespace Entity_OAPS.Controllers
{
   public class AppointmentController : Controller
   {
      Entity_OAPSEntities1 db = new Entity_OAPSEntities1();
      public ActionResult AppointmentIndex()
      {
         return View();
      }
      public void Insert(tblAppointment obj)
      {
         db.tblAppointments.Add(obj);
         db.SaveChanges();
      }
   }
}
```

***Appointment cshtml:-***

```
@{
    ViewBag.Title = "AppointmentIndex";
}

<script src="~/jquery.min.js"></script>
<script src="~/Appointment.js"></script>

<table style="background-color:cyan; color:black; width:26%">
    <tr>
        <td>
            <h2>Appintment Form</h2>
        </td>
    </tr>
</table>

<table style="background-color:goldenrod; color:black">
    <tr>
        <td>Patient Name:</td>
        <td><input type="text" id="txtname" placeholder="First Name"
/></td>
    </tr>
    <tr>
        <td>Patient Department: </td>
        <td><input type="text" id="txtldept" placeholder="Department"
/></td>
    </tr>
    <tr>
        <td>Appointment Date:</td>
```

```
        <td><input type="date" id="txtappointment" /></td>
    </tr>
    <tr>
        <td>Patient Problem:</td>
        <td><input    type="text"    id="txtproblem"    placeholder="Patient
Problem" /></td>
    </tr>
    <tr>
        <td>Patient Phone:</td>
        <td><input   type="text"   id="txtphone"   placeholder="123-456-789"
/></td>
    </tr>
    <tr>
        <td>Patient Email :</td>
        <td>
            <input type="text" id="txtemail" placeholder="Email" />
        </td>
    </tr>
    <tr>
        <td></td>
        <td><input   type="button"   id="btnappointment"   value="Appointment"
onclick="Appointment()" /></td>
    </tr>
</table>
```

## *Appointment JavaScript:-*

```
function Clear() {
```

```
    $("#txtname").val("");

    $("#txtldept").val("");

    $("#txtappointment").val("");

    $("#txtproblem").val("");

    $("#txtphone").val("");

    $("#txtemail").val("");

    $("#btnappointment").val("Appointment");

}

function Appointment() {

    $.ajax({

        url: '../Appointment/Insert',

        data: { a_name: $("#txtname").val(), a_dept: $("#txtldept").val(),
a_appointment_date:          $("#txtappointment").val(),          a_problem:
$("#txtproblem").val(),     a_phone:     $("#txtphone").val(),     a_email:
$("#txtemail").val() },

        success: function () {

            alert("Appointment is successfull !");

            Clear();

        },

        error: function () {

            alert("Appointment faill!!");

        }

    });

}
```

## ***Feedback Controller:-***

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Entity_OAPS.Model;

namespace Entity_OAPS.Controllers
{
   public class FeedbackController : Controller
   {
      Entity_OAPSEntities1 db = new Entity_OAPSEntities1();
      public ActionResult FeedbackIndex()
      {
         return View();
      }
      public void Insert(tblfeedback obj)
      {
         db.tblfeedbacks.Add(obj);
         db.SaveChanges();
      }
   }
}
```

***Feedback cshtml:-***

```
@{
```

```
    ViewBag.Title = "FeedbackIndex";
}

    <script src="~/jquery.min.js"></script>
    <script src="~/Feedback.js"></script>

<table style="background-color:blue; color:white; width:21%">
    <tr>
        <td colspan="2">
            <h2>Feedback</h2>
        </td>
    </tr>
</table>
<table style="background-color:aqua; color:red">
    <tr>
        <th>Name:</th>
        <td><input type="text" id="txtname" placeholder="Name" /></td>
    </tr>
    <tr>
        <th>Comments:</th>
        <td><input   type="text"   id="txtcomments"   placeholder="Comments"
/></td>
    </tr>
    <tr>
        <td></td>
        <td><input    type="button"    id="btnfeedback"    value="Feedback"
onclick="Feedback()" /></td>
    </tr>
```

```
</table>
```

## *Feedback JavaScript:-*

```
function Clear() {
    $("#txtname").val("");
    $("#txtcomments").val("");
    $("#btnfeedback").val("Feedback");
}

function Feedback() {
    $.ajax({
        url: '../Feedback/Insert',
        data: { f_name: $("#txtname").val(), f_comment: $("#txtcomments").val() },
        success: function () {
            alert("Feedback is successfull send !");
            Clear();
        },
        error: function () {
            alert("Feddback fail!!");
        }
    });
}
```

## *Contact Controller:-*

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Entity_OAPS.Model;

namespace Entity_OAPS.Controllers
{
   public class ContactController : Controller
   {
      Entity_OAPSEntities1 db = new Entity_OAPSEntities1();
      public ActionResult ContactIndex()
      {
         return View();
      }
   }
}
```

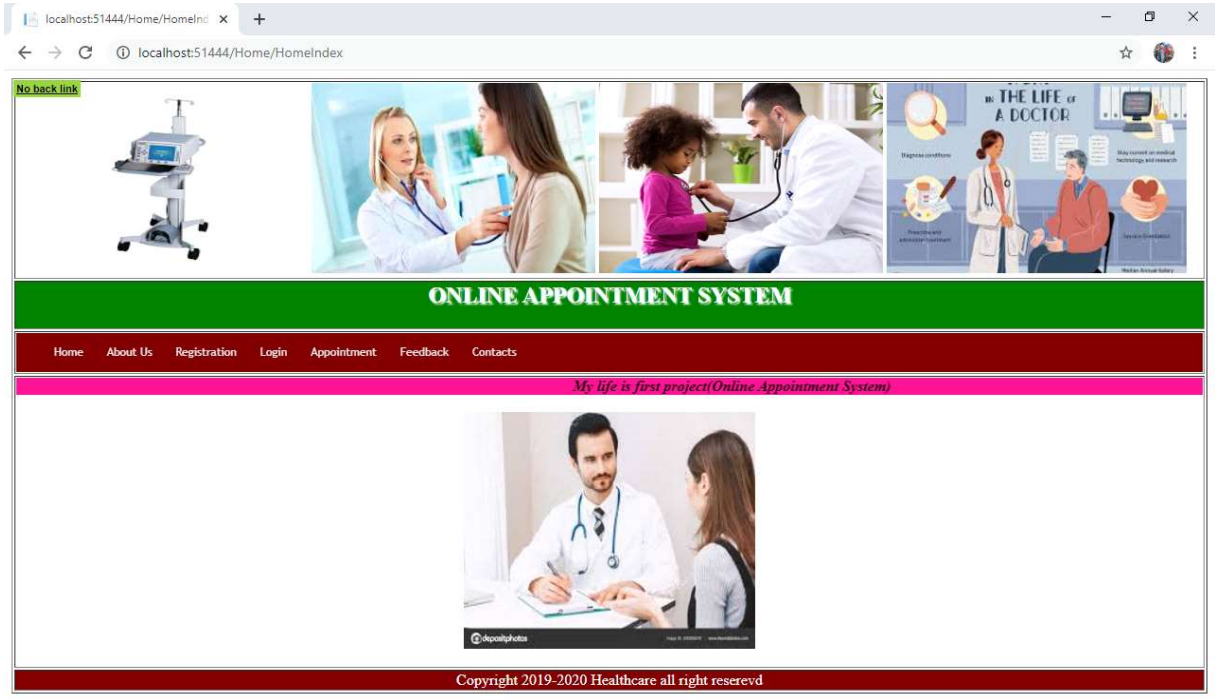
## *Contact cshtml:-*

```
@{
    ViewBag.Title = "ContactIndex";
}
```

```
<table style="color:red" border="1">
    <tr>
        <td style="        border-bottom-color: black;
        border-right-color: black;
        border-top-color: aqua;
        border-left-color: aqua;
        border-right-width: 10px;
        border-bottom-width: 10px;
        border-left-width: 10px;
        border-top-width: 10px">
            <h2>
                <i>
                    <b>
                            Contact Us :<br />
                            Administration <mark>Department</mark><br />
                            +91-9810124920<br />
                            mohammadpyar143@gmail.com
                    </b>
                </i>
            </h2>
        </td>
    </tr>
</table>
```

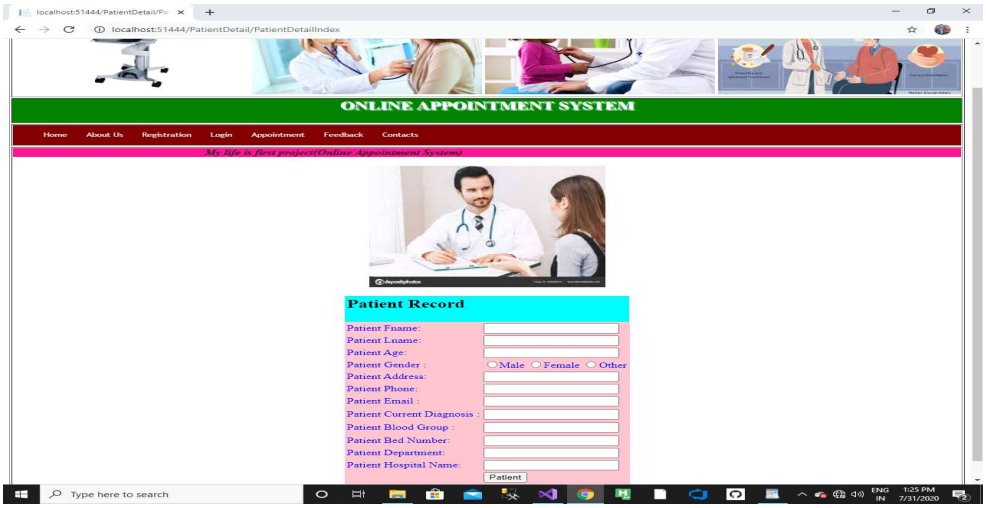
# 6. User Interface Screen

## Home Page :-



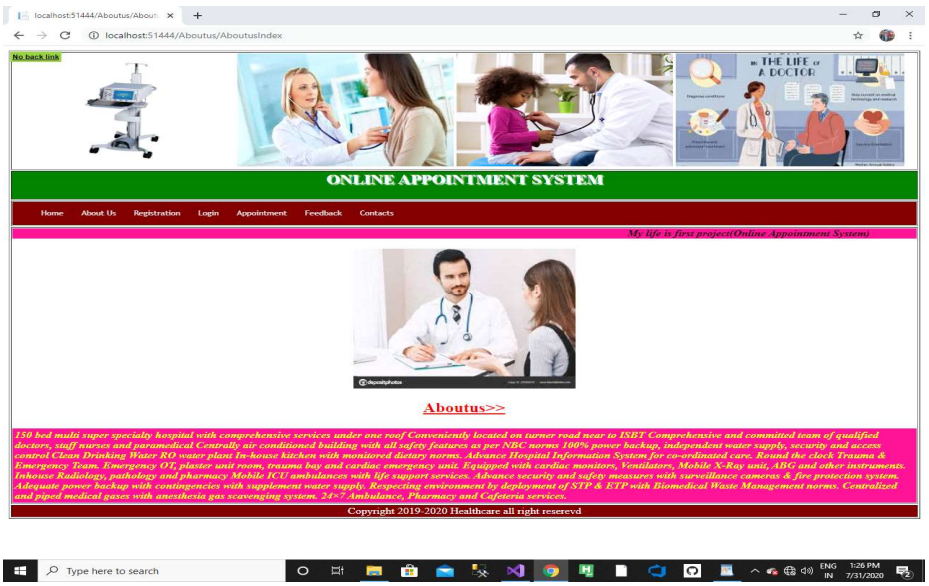


## Patient Page:-

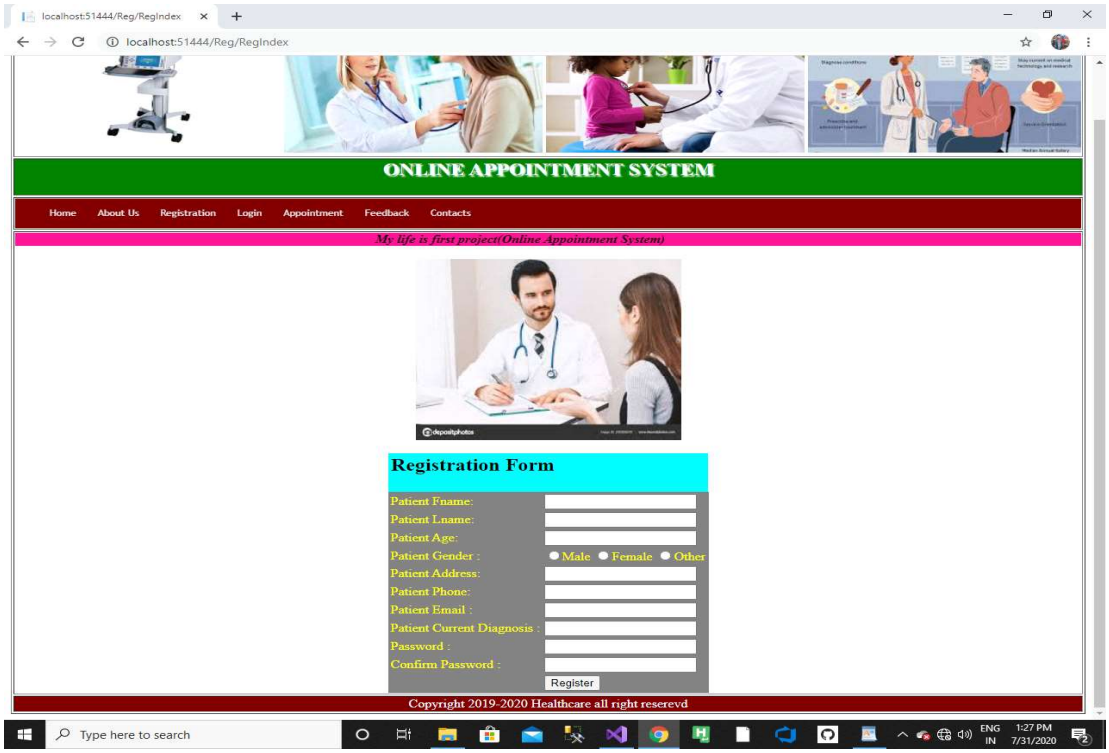


# Hospital Page:-


# About us Page:-

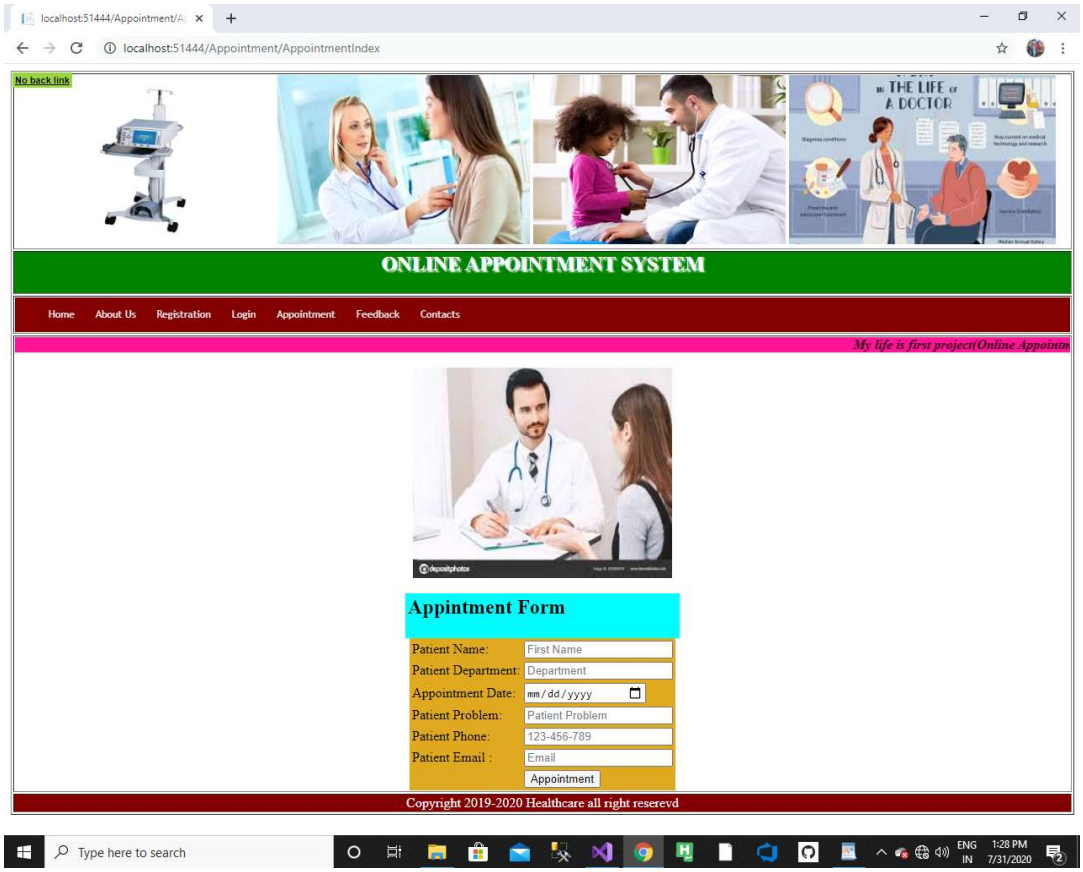


# Registration Page:-

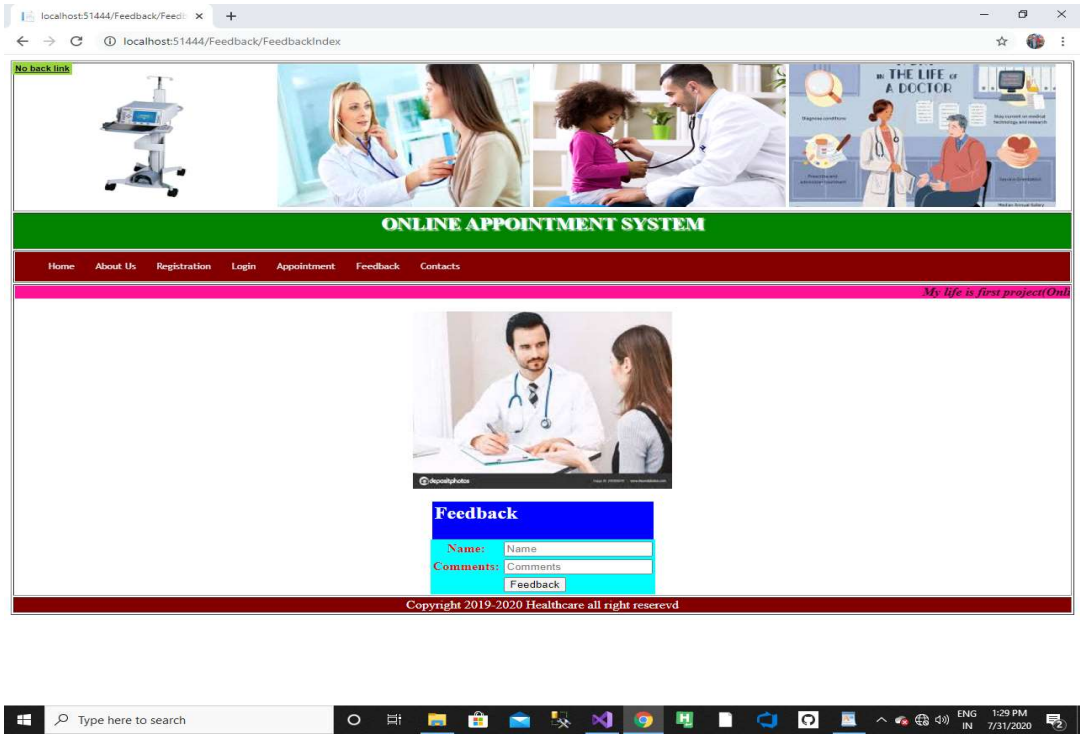



# *Login Page:*

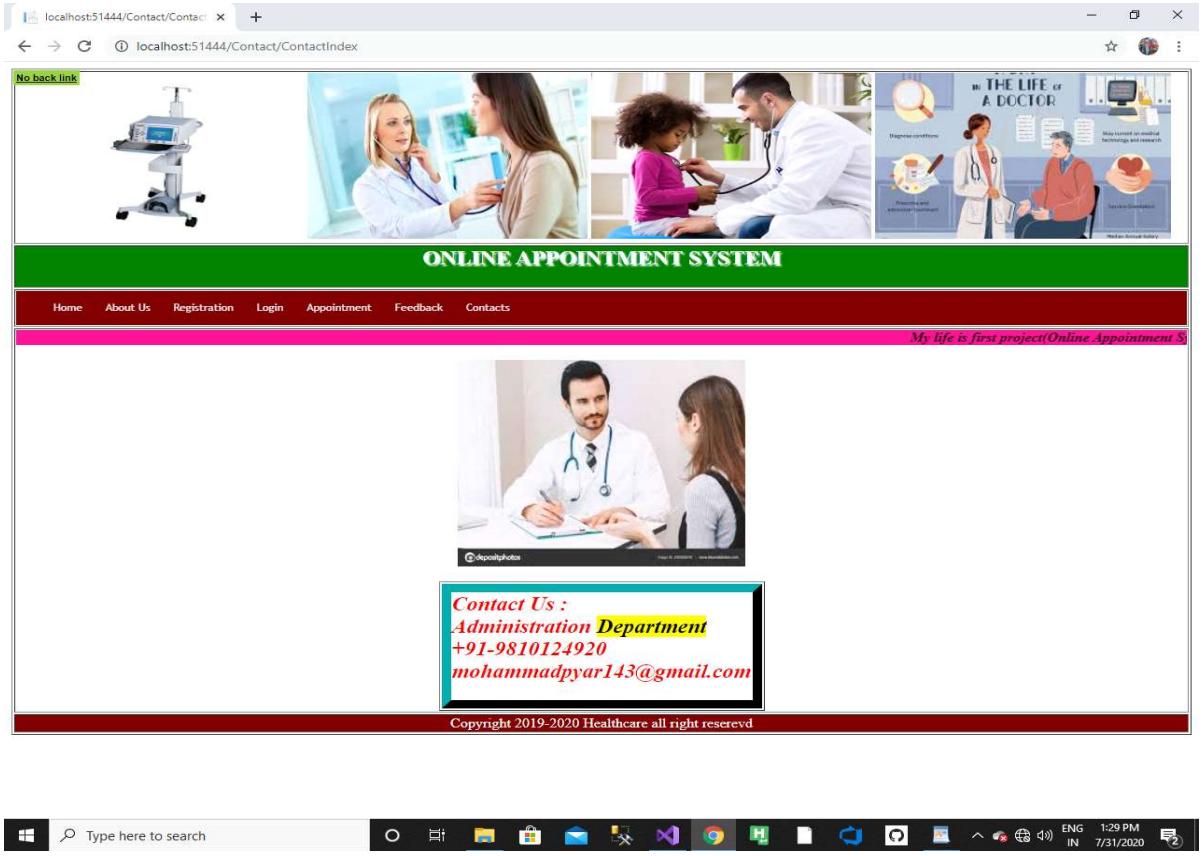
# Appointment Page:




# Feedback Page:-



## ***Contactus Page:-***







# 7. Implementation and Testing

## 7.1) Implementation Approaches

Our project works on the **Waterfall Model with Feedback.**

**Introduction to Waterfall Model with Feedback**

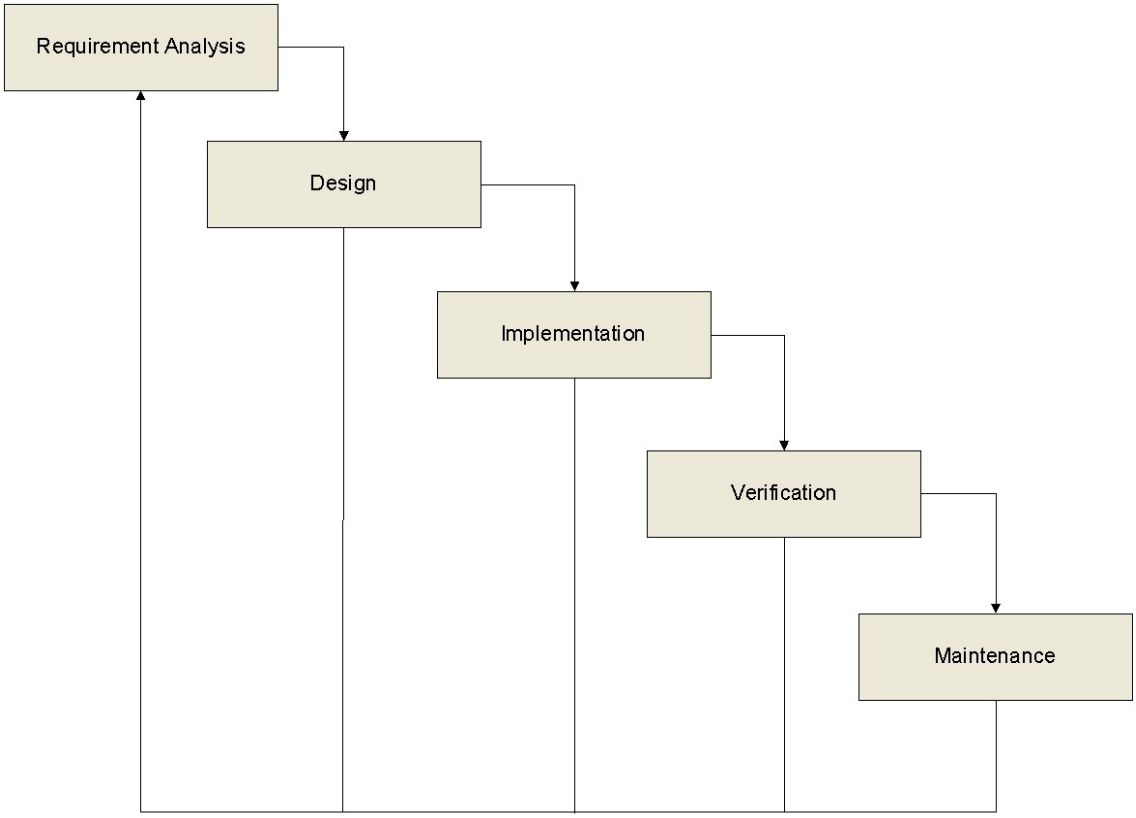



**Fig: Waterfall Model with Feedback**

We have the process of waterfall model with feedback as below:

### 1. Requirements Analysis

In this phase, we perform the analysis of requirements for the project. We analysis the problem and define its possible solutions. Then we move further with the best possible solution to start our project. To work with the solution, we follow the method of Divide and Conquer approach, and break down our project in different modules. Also, we perform the feasibility study to check whether this project is feasible or not on various parameters. We also analysis the best tool available in the market to use as Front End and Back-End, and in the last, we analysis the number of days required to finish the project. We analysis the number of days with the help of Gantt chart.

### 2. Design

In the second phase, on the behalf of possible modules (defined in the first phase), we create the Data Flow Diagram (DFD) up to 2nd Level. We also design the user interface that could attract both IT people and Non-IT people. We also define the entities with required attributes and their relationship with the help of ER Diagram.

**3. Implementation**

In the third phase, we implement the project through programming code. We develop the number of pages required for the project and break it into two sections – one for Administrator and second for other Users.

**4. Verification**

In the fourth phase, we make the possible test cases and test data for testing. We follow both white box testing and black box testing approach. In this phase, we identify various errors and security issues like, error on wrong login id or password, like or dislike the articles or blogs only after login etc.

In this phase, we create a list of possible solutions for the achieved errors and security issues, and then pass the feedback to the analysis phase for the modification in the modules, design phase for any modification in the user interface or DFD or in the ER Diagram. We also use feedback in the Implementation phase, for changing in the code.

**5. Maintenance**

In this phase, we define the hardware and software requirements through which we can view our website without any error for long time. However if any error received, the feedback will pass to the every phase to rectify the error.

## *7.2) TESTING Approach*

A systematic and wholly procedural plan for testing was adopted the procedure while implementing testing went in hand with preparing a unit test plan.

- Test the modules thoroughly- cover all the access paths. GeneCharges enough data to cover all the access paths arising from conditions.
- Test the modules by passing wrong data.
- To test different access paths, look at the conditional statement. Enter some data in the test file, which would satisfy the condition and again test the script. Repeat this process many times.
- After each test , analyze the log file to ensure proper, understandable and useful messages are present in the log file.

•Test for locking by invoking multiple concurrent processes.

*TESTING OBJECTIVES*

- Testing is a process of executing a program with the intent of finding an error.
- A goal test case is one that has a probability of finding an as yet undiscovered error.
- A successful test is one that uncovers an as yet undiscovered error.

*TESTING PRINCIPLE*

- All tests should be traceable to customer requirements.
- Test should be planned long before testing begins.
- The testing should begin “in the small” and progress towards testing “in the large”.
- Exhaustive testing is not possible.

- To be most efficient, an independent third party should conduct testing.

*CHARACTERS OF A GOOD TEST*

- A good test has a high probability of finding an error.
- A good test is not redundant.
- A good test should be best of breed.
- A good test should be neither too simple nor too complicated.

*TYPES OF TESTING*

- White box testing.
- Black box testing.

***WHITE BOX TESTING***

White Box testing which is performed early in the testing process, is also called glass-box testing. Using white box testing , the software engineer can derive test cases that

- Guarantee that all the independent paths within the module have been exercised at least once.
- Exercise all logical decisions on their true and false sides.
- Execute all loops at their boundaries and within their operational bounds.
- Exercise internal data structure to assure their validity.

***BLACK BOX TESTING***

Black box testing to be applied during later stage of testing. Black box testing, focuses on the functional requirement of the software. Black box testing enables the software engineer to derive sets of input conditions that will fully exercise the functional requirement of a program.

Black box testing enables to find error in the following categories:-

- Incorrect or missing functions.
- Interface errors.
- Errors in the data structure or external database access.
- Performance errors.
- Initialization and termination errors

By applying Black box testing a set of test cases that satisfy the following criteria ---

- Test cases that reduce, by a count that is greater than one, the number of additional test case must be designed to achieve reasonable testing.
- Test cases that tell us something about the presence or absence of classes of errors, rather than associated only with the specific test at hand.

## Project Testing

### Testing

Testing is the filter to catch defects before they are “discovered” by the user. Every time a user runs the program, he/she generate a “test case”. We tried the test cases to find the defects first since software development is a human activity and there may be potential errors. So testing before the product is delivered helped us to assure the quality and save resources.

### Testing Steps

We started testing each individual new component and worked out as units test, integration test, high order test and different testing techniques are used at different times in the process.

**Testing Techniques**

We have applied the following testing techniques in this project.

**White Box Testing**

White-box testing (also known as clear box testing, glass box testing, and transparent box testing and structural testing) is a method of testing software that tests internal structures or workings of an application. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs.

While white-box testing can be applied at the unit**, integration and system** levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test.

**Black Box Testing**

In black box testing we exercises the input and output requirements of a program, tested by conducting specifications of requirements for which the software should do. This test is derived from the I/O specification and used in most functional tests.

## Testing Report

**Module Name: Login**

**Phase Name: Login Form**

| Test No | Test Data (if Any) | Output | Suggestion |
|---|---|---|---|
| 1 | If login id not entered | Login Id cannot be blank | Enter Login id |
| 2 | If password not entered | Password cannot be blank | Enter password |
| 3 | If login Id Not found | User Not found. | Enter Login Id and Password again. |

**Phase Name: Registration Form**

| Test No | Test Data (if Any) | Output | Suggestion |
|---|---|---|---|
| 1 | If login id not entered | Login Id cannot be blank | Enter Login id |
| 2 | If login id has already been allotted | Login Id is already registered. Choose another Login Id. | Enter any unique login Id |
| 3 | if password is less than 6 characters | Password must be greater than 6 letters. | Choose a long password but not a dictionary word. |
| 4 | If password not Matched | Password not confirmed | Both password and its confirmation password must be same |
| 5 | If security question not entered | Security Question will be used for password recovery | Enter Security question |
| 6 | If security answer not entered | Security answer will be used for password recovery when security question will asked. | Enter security answer. |
| 7 | If user not agreed with the term and conditions of company | User account cannot be created until he agreed with the company terms and conditions. | Click on Agree button. |

**Phase Name: Write Feedback Comments**

| Test No | Test Data (if Any) | Output | Suggestion |
|---|---|---|---|
| 1 | If user is not name | Not accecept data Page | First Enter name |
| 2 | If comment is more than 200 words | More than allowed space. | Enter article within 200 words. |

# 8) Conclusion :

## 8.1) Conclusion

The project **Online Appointment System(OLAS)** is for computerizing the working in a web-world. The software takes care of all the requirements of an average hospital and is capable to provide easy and effective storage of information related to patients that come up to the hospital.

It generates test reports; provide prescription details including various tests, diet advice, and medicines prescribed to patient and doctor. It also provides injection details and billing facility on the basis of patient's status whether it is an indoor or outdoor patient.

The system also provides the facility of backup as per the requirement.

## 8.2) Limitation of the Project :

- Don't store more data in database.
- Not applied in emergency case.
- Not treatments all types of Disease.
- Need of server for working this project.

## 8.3) Future scope of Application :

The scope of Online Appointment System is :

- Large patient database can be accommodated in our application in the future.
- Necessary steps can be taken to develop/revise all the documents preferably the patient bill make more understandable to the patient as well as to the departmental staff.
- Necessary steps can be taken in the proposed system to accommodate the forth coming changes regarding the course scheduling.
- The formats of different types of reports can be made to suite the requirements.

How to paid charge (fee) :

- Functions can be embodied for payments through cheques, credit cards (and barcode).

- The program's compatibility can be increased for 'online' transactions. We can upgrade the program to complete utility Online Appointment System software
- The program's compatibility can be increased for 'online' transactions.
- Surely, we can upgrade the program to a complete utility

  Compalint & Feedback management software.

Enquiry and Services for future.

# REFERENCES


**BOOK :**

**SOFTWARE ENGRINEERING :** Roger S. P. Pressmen; McGraw Hill International Edition And IGNOU Block MCS-034

**SYSTEM ANALYSIS AND DESIGN :** TMGH And IGNOU Block MCS-014

**C#.NET**: The Complete Reference, TATA McGRAW HILL

**WEB-SITE :**

http://standards.ieee.org

http://www.rspa.com

**VISITE at** :

PATEL HOSPITAL, DELHI