# CHAPTER 1

# INTRODUCTION

This is an online event management system software project that serves the functionality of an event manager. The system allows only registered users to login and new users are allowed to register on the application. The project provides most of the basic functionality required for an event. It allows the user to select from a list of event types. Once they enters an event type the system then allows the user to select the data and time of event, place and the event equipments.

Event management is the application of project management to the creation and development of large scale events such as marriages, festivals, birthday party, conferences, formal parties, concerts, or conventions. It involves studying the brand, identifying the target audience, devising the event concept, planning the logistics and coordinating the technical aspect. The process of planning and coordinating the event is usually referred to as event planning and can include budgeting, scheduling, site selection, acquiring necessary permits, coordinating transportation and parking, arranging for speakers or entertainers, arranging decor, event security, catering and emergency plans.

The events industry now includes events of all size from the Olympics down to business breakfast meetings. Many industries, charitable organizations, and interest groups hold events in order to market themselves build business relationships, raise money or celebrate achievement. Event management is considered one of the strategic marketing and communication tools by companies of all sizes. A number of elements such as music, live entertainment or even the particular venue may be used to influence the tone and atmosphere of an event.

Event manager may also use news media to target their audience, hoping to generate media coverage which will reach thousands or millions of people. They can invite their audience to their events and reach them at the actual event. All these data is logged in the database and the user is given a receipt number for these booking. This data is then sent to the administrator (website owner) and they interact with the client as per requirements and his contact data is stored in the database.

Everybody wants their special day to be perfect in every way whether it’s a surprise celebration, wedding reception, a corporate meeting or launch of a new product or company, special Event planning and Décor can make it seamless, memorable and exceptional.

* Wedding planning
* Bridal flowers
* Reception and party décor
* Silk floral arrangements
* Engagement
* Birthday Party

AIM AND OBJECTS

1.1: AIM

Our aim is to make user friendly application that can let a user to plan a decoration for a event .He can select the event and according to the event the appropriate decoration can made from the list of panel . client will also find out the cost for particular decoration and virtual representation of that particular decoration

The project will help clients to decorate for an event and the cost required for particular decoration

1.2: OBJECTIVE

No need to visit decorators: Client can make a virtual representation of the decoration from its place.

**CHAPTER 2**

**LITERATURE SURVEY**

It is the process of bringing developed system of revised system into operational use .If the implementation phase is not carefully planned and controlled , it can lead to many problems. Thus proper implementation is essential to provide a reliable system to meet managerial requirements.

The application is completely written .net .This enables the debit card issues to use this application across a variety of devices independent of the vendor of the devices. We use sql as a back end for storing database.

**2.1: TECHNOLOGY USED:**

**2.1.1: Dot Net:**

The .NET Framework (pronounced dot net) is a proprietary, partially open source freeware software framework developed by Microsoft that runs primarily on Microsoft Windows.

**2.1.2: What is Dot Net?**

The **.NET** is the technology from Microsoft, on which all other Microsoft technologies will be depending on in future. It is a major technology change, introduced by Microsoft, to catch the market from the SUN's Java. Few years back, Microsoft had only VC++ and VB to compete with Java, but Java was catching the market very fast. With the world depending more and more on the Internet/ Web and java related tools becoming the best choice for the web applications, Microsoft seemed to be losing the battle. Thousands of programmers moved to java from VC++ and VB. To recover the .market, .Microsoft announced .NET.

Microsoft has a wonderful history of starting late but catching up quickly. This is true in case of .NET too. Microsoft put their best men at work for a secret project called Next Generation Windows Services (NGWS)., under the direct supervision of Mr. Bill Gates.

The outcome of the project is what we now know as .NET. Even though .NET has borrowed most of its ideas from Sun's J2EE, it has really outperformed their competitors.

Microsoft's VC++ was a powerful tool. But it was too complex. It has too many data types, and developers had to learn many libraries including Windows SDK, MFC, ATL, COM etc. There were many data type compatibility issues while exchanging data between different layers. Visual Basic was too easy, and many serious programmers hated it just for that reason. Even though Visual basic was very easy to use, it was not very flexible to develop serious applications. SUN's Java became a very good choice for these reasons. It had the flexibility and power of C++ and at the same time easy enough to catch the attention of VB programmers.

**2.1.3: The structure of Dot Net:**

* It is a platform for application developers.
* It is a Framework that supports Multiple Language and Cross language integration.
* IT has IDE (Integrated Development Environment).
* Framework is a set of utilities or can say building blocks of your application system.
* .NET Framework provides GUI in a GUI manner.
* .NET is a platform independent but with help of Mono Compilation System (MCS). MCS is a middle level interface.
* .NET Framework provides interoperability between languages i.e. Common Type System (CTS).
* .NET Framework also includes the .NET Common Language Runtime (CLR), which is responsible for maintaining the execution of all applications developed using the .NET library.
* The .NET Framework consists primarily of a gigantic library of code.



**Class Library:**

.NET Framework includes a set of standard class libraries. The class library is organized in a hierarchy of namespaces. Most of the built-in APIs are part of either system or Microsoft namespaces. These class libraries implement a large number of common functions, such as file reading and writing, graphic rendering, database interaction, and XML document manipulation, among others. .NET class libraries are available to all CLI compliant languages. .NET Framework class library is divided into two parts: Framework Class Library (FCL) and Base Class Library (BCL).

FCL includes a small subset of the entire class library and is the core set of classes that serve as the basic API of CLR. Classes in and some classes.inSystem.dll and System.core.dll are part of FCL. FCL classes are available in .NET Framework as well as its alternative implementations including .NET Compact Framework, Microsoft Silver light and Mono.

BCL is a superset of FCL and refers to the entire class library that ships with .NET Framework. It includes an expanded set of libraries, including Windows Forms, ADO.NET, ASP.NET, Language Integrated Query (LINQ), Windows Presentation Foundation (WPF), Windows Communication Foundation (WCF) and Workflow Foundation (WF). BCL is much larger in scope than standard libraries for languages like C++, and comparable in scope to standard libraries of Java.

**.NET Core:**

.NET Core is a free and open-source partial implementation of the .NET Framework. It consists of CoreCLR and CoreFX, which are partial forks of CLR and BCL respectively.NET Core comes with an improved JIT compiler, called RyuJIT.

**Assemblies:**

Compiled CIL code is stored in CLI assemblies. As mandated by the specification, assemblies are stored in Portable Executable (PE) file format, common on Windows platform for all DLL and EXE files. Each assembly consists of one or more files, one of which must contain a manifest bearing the metadata for the assembly. The complete name of an assembly (not to be confused with the file name on disk) contains its simple text name, version number, culture, and public key token. Assemblies are considered equivalent if they share the same complete name, excluding the revision of the version number.

A private key can also be used by the creator of the assembly for strong naming. The public key token identifies which private key an assembly is signed with. Only the creator of the key pair( typically .NET developer signing the assembly) can sign assemblies that have the same strong name as a previous version assembly, since the creator is in possession of the private key. Strong naming is required to add assemblies to Global Assembly Cache.

**C++/CLI:**

Microsoft introduced C++/CLI in Visual Studio 2005, which is a language and means of compiling Visual C++ programs to run within the .NET Framework. Certain portions of the C++ program still run within an unmanaged Visual C++ Runtime, while specially modified portions are translated into CIL code and run with the .NET Framework's CLR.

Assemblies compiled using the C++/CLI compiler are known as mixed-mode assemblies, since they contain native and managed code within the same DLL, Such assemblies are also difficult to reverse engineer, since .NET de compilers such as .NET Reflector only reveal the managed code.

**Security:**

.NET Framework has its own security mechanism with two general features: Code Access Security (CAS), and validation and verification. CAS is based on evidence that is associated with a specific assembly. Typically the evidence is the source of the assembly (whether it is installed on the local machine or has been downloaded from the intranet or Internet). CAS uses evidence to determine the permissions granted to the code. Other code can demand that calling code be granted a specified permission. The demand causes CLR to perform a call stack walk: every assembly of each method in the call stack is checked for the required permission; if any assembly is not granted the permission a security exception is thrown.

Managed CIL byte code is easier to reverse-engineer than native code, unless obfuscated. NET decompile programs enable developers with no reverse-engineering skills to view the source code behind unobfuscated .NET assemblies. In contrast, apps compiled to native machine code are much harder to reverse-engineer and source code is almost never produced successfully, mainly because of compiler optimizations and lack of reflection. One concern is over possible loss of trade secrets and the bypassing of license control mechanisms. To mitigate this, Microsoft has included Dot fuscator Community Edition with Visual Studio .NET since 2002.[b] Third-party obfuscation tools are also available from vendors such as vm \ware, V.i. Labs, Xenocode, Red Gate Software. Method-level encryption tools for .NET code are available from vendors such as Safe Net.

**Memory management:**

CLR frees the developer from the burden of managing memory (allocating and freeing up when done); it handles memory management itself by detecting when memory can be safely freed. Instantiations of .NET types (objects) are allocated from the managed heap; a pool of memory managed by CLR. As long as there exists a reference to an object, which might be either a direct reference to an object or via a graph of objects, the object is considered to be in use. When there is no reference to an object, and it cannot be reached or used, it becomes garbage, eligible for collection.

.NET Framework includes a garbage collector (GC) which runs periodically, on a separate thread from the application's thread, that enumerates all the unusable objects and reclaims the memory allocated to them. It is a non-deterministic, compacting, mark-and-sweep garbage collector. GC runs only when a certain amount of memory has been used or there is enough pressure for memory on the system. Since it is not guaranteed when the conditions to reclaim memory are reached, GC runs are non-deterministic. Each .NET application has a set of roots, which are pointers to objects on the managed heap (managed objects). These include references to static objects and objects defined as local variables or method parameters currently in scope, as well as objects referred to by CPU registers. When GC runs, it pauses the application, and for each object referred to in the root, it recursively enumerates all the objects reachable from the root objects and marks them as reachable. It uses CLI metadata and reflection to discover the objects encapsulated by an object, and then recursively walk them. It then enumerates all the objects on the heap (which were initially allocated contiguously) using reflection. All objects not marked as reachable are garbage. This is the mark phase. Since the memory held by garbage is not of any consequence, it is considered free space. However, this leaves chunks of free space between objects which were initially contiguous. The objects are then compacted together to make used memory contiguous again. Any reference to an ect invalidated by moving the object is updated by GC to reflect the new location. The application is resumed after the garbage collection is over.

GC used by .NET Framework is also generational. Objects are assigned a generation; newly created objects belong to Generation 0. The objects that survive a garbage collection are tagged as Generation 1, and the Generation 1 objects that survive another collection are Generation 2 objects. .NET Framework uses up to Generation 2 objects. Higher generation objects are garbage collected less frequently than lower generation objects. This helps increase the efficiency of garbage collection, as older objects tend to have a longer lifetime than newer objects. Thus, by eliminating older (and thus more likely to survive a collection) objects from the scope of a collection run, fewer objects need to be checked and compacted.

**Performance:**

When an application is first launched, the .NET Framework compiles the CIL code into executable code using its just-in-time compiler, and caches the executable program into the .NET Native Image Cache. Due to caching, the application launches faster for subsequent launches, although the first launch is usually slower. To increase speed of the first launch, developers may use the Native Image Generator utility to manually compile and cache any .NET application, ahead-of-time.

The garbage collector, which is integrated into the environment, can introduce unanticipated delays of execution over which the developer has little direct control. "In large applications, the number of objects that the garbage collector needs to deal with can become very large, which means it can take a very long time to visit and rearrange all of them.

.NET Framework provides support for calling Streaming SIMD Extensions (SSE) via managed code from April 2014 in Visual Studio 2013 Update 2. However, Mono has provided support for SIMD Extensions as of version 2.2 within the Mono.Simd namespace; before. Mono's lead developer Miguel de Icaza has expressed hope that this SIMD support will be adopted by CLR's ECMA standard.Streaming SIMD Extensions have been available in x86 CPUs since the introduction of the Pentium III. Some other architectures such as ARM and MIPS also have SIMD extensions. In case the CPU lacks support for those extensions, the instructions are simulated in software.

**CHAPTER 3**

**SYSTEM REQUIREMENTS**

**HARDWARE REQUIREMENTS:**

* System :      Pentium IV 2.4 GHz.
* Hard Disk :     40 GB.
* Floppy Drive : 44 Mb.
* Monitor : 15 VGA Colour.
* Ram : 512 Mb.

**SOFTWARE REQUIREMENTS:**

* Operating system : Windows XP/7.
* Coding Language : c#
* IDE : Visual Studio 2010
* Database : SQL Server 2008 R2

**3.1 FUNCTIONAL REQUIREMENTS:**

View Menus: There has a list of menu to allow customer to view the menu items .The menu is show in the data list view that displays the picture, name, price of items.

* Check Orders: This check list is use by admin and It will show the all the Order details.
* Give Feedback: Customers allow to write their feedback by click the Feedback. This form can use by customers to give us the feedback for either the services ,quality of food, facility, or give the recommendation for us to make improvement and increase customer satisfaction
* View all Menu: admin can view all the menu items.
* Update Menu: Admin also enable to update or edit the menu list.
* Add new item: When the food court created or produce a new item admin may need to add the new item category and add the information.
* **Remove item:** If the product is not in good sales, and the food court decide to stop selling the item. Admin can delete that item.

**3.2 NON FUNCTIONAL REQUIREMENTS:**

Following are the non-functional requirements

* Consistency: The system provides consistency user interface design to end user. The designs of the screen are standardize and consistent that make the end user feel comfortable to use it.
* Convenience: The system may give convenience to the end user to make reserve table via online platform. Customers no need to go out or phone call to make reserve table from the Food court.
* Availability: The system provides the end user to login the system to browse or make reserve table on 24 hours.
* Usability: The system is ready to use system the end user may feel easy to use the system.
* Security: The system provide the password access control to avoid unauthorized user to login to the system. The system also authenticates the staff level to access to some of the admin part.
* Reliability: The system provide effective method to maintain the back-end of the system such as generate report. All of the feedbacks and reserve table details are manage by this system effectively.

**CHAPTER 4**

**SYSTEM ANALYSIS**

.**4.1 EXISTING SYSTEM:**

* The system provides so many functionality for the user to book any event.
* The system is useful as it calculates an exact cost for all resources required during the event.

**DISADVANTAGES OF EXISTING SYSTEM:**

* Cost is more and it is time consuming.
* It takes more time for login and logout.

**4.2 PROPOSED SYSTEM:**

* The system provides most of the basic functionality required for an event.
* The system stored the data in the database.

**ADVANTAGES OF PROPOSED SYSTEM:**

* The user gets all the resource at a single place instead of wandering around for these.
* This system is effective and saves time and cost of the users.

**CHAPTER 5**

**DESIGN**

**5.1 UML DIAGRAMS**

The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing and documenting of a software intensive system. The UML gives a standard way to standard way to write a system blueprints, covering conceptual things, such as classes written in a specified programmed language, database schemas and reusable software components.

There are types of diagrams

* Class Diagram
* Use case Diagram
* Sequence Diagram
* Collaboration Diagram
* Activity Diagram

**5.1.1 USE CASE DIAGRAM:**

* Use case is a description of set of sequence of actions that a system performs that yields an observable results. These are important in organizing the behaviors of a system.
* Use case is used to structure the behavioral things in a model.
* A Use case is realized by collaboration.
* Graphically , a use case is rendered as a ellipse with solid lines, usually including only its name

**ACTOR**- also referred to as roles. Name and stereotype of an actor can be changed in its properties lab

**Use Cases-** these can have extension points.



Fig 5.1.1(a) Admin Use Case Diagram



Fig 5.1.1(b) User Use Case Diagram

**5.1.2 CLASS DIAGRAM**

The class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing , describing documenting different aspects of a system but also for constructing executable code of the software application. The class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object oriented system because they are the only UML diagrams which can be mapped directly with object oriented languages.

The class diagram shows a collection of classes, interfaces, associations, collaboration and constraints. It is also known as ***structural diagram***.



Fig 5.1.2 Class Diagram

**5.1.3 SEQUENCE DIAGRAM**

A Sequence diagram is an interaction diagram that emphasizes the time ordering of messages. A Sequence diagram shows a set of objects and messages sent and received by those objects. The objects are typically named or anonymous instances of classes, but may also represent instances of other thing such as collaboration, components and nodes. These are used to illustrate the dynamic view of a system. It is isomorphic, which means taking any one of them in to consideration, and can be transformed to another.

A Sequence diagram shows, as parallel vertical lines (*Life lines*), a different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in the in which they occur. This allows the specification of simple runtime scenarios in a graphical manner. In order to show the sequence of time ordering of messages involved in the working of PHC a sequence diagram was used.



Fig 5.1.3 Sequence Diagram

**5.1.4 COLLABORATION DIAGRAM**

A collaboration diagram is an interaction diagram that emphasizes the structural organization of the objects that send and receive messages. Collaboration diagrams are isomorphic, meaning that you can take one and transform it into the other.

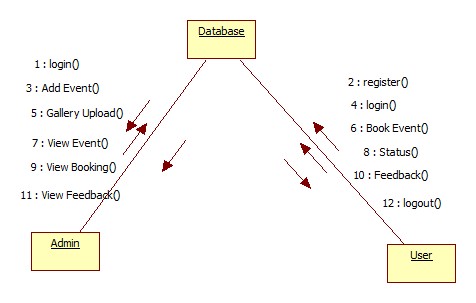
****

Fig 5.1.4 Collaboration Diagram

**5.1.5 ACTIVITY DIAGRAM**

An Activity diagram shows the flow from activity to activity within a system it emphasizes the flow of control among objects. Activity diagrams are graphical representation of workflows of step wise and actions with support for choice, iteration and concurrency in the unified modeling language, activity diagrams are intended to model both computational and organizational process(i.e workflows).Activity diagrams show the overall flow of control.

****

Fig 5.1.5(a) Admin Activity Diagram

****

Fig 5.1.5(b) User Activity Diagram

**CHAPTER-6**

**IMPLEMENTATION**

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implementation stage involved careful planning, investigation of the existing system and it’s constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

Implementation is the process of converting a new system design into operation. It is the phase that focuses on user training, site preparation and file conversion for installing a candidate system. The important factor that should be considered here is that the conversion should not disrupt the functioning of the organization.

**6.1 MODULES**:

**Admin & User:**

Admin will login with his own credentials and create new events where user needs to select those events. Admin can view all the user details and if required delete the unauthorized users and check the user’s booking status. User will register and login with his own credentials and book an event, gives the feedback about that event.

**6.2 CODING:**

**6.2.1 User Register code**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

public partial class Registration : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

protected void TextBox3\_TextChanged(object sender, EventArgs e)

{

}

protected void Button1\_Click(object sender, EventArgs e)

{

try

{

string qry = "insert into Reg values('" + txtfirstname.Text + "','"+txtlastname.Text+"','" + txtemailid.Text + "','" + RadioButton1.SelectedItem.Text

+ "','"+RadioButton2.SelectedItem.Text+"','"+txtage.Text+"','" +txtusername.Text + "','" +txtpassword.Text + "')";

int i = obj.inupdel(qry);

if (i > 0)

{

Response.Write("<script>alert('Registration Sucessfully Completed')</script>");

txtfirstname.Text=txtlastname.Text=txtemailid.Text=txtage.Text=txtusername.Text=txtpassword.Text="";

}

else

{

Response.Write("<script>alert('Not Yet Completed')</script>");

}

}

catch (Exception ex)

{

Response.Write("<script>alert('" + ex.Message + "')</script>");

}

}

}

**6.2.2 Login Code**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

public partial class Login : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

Class1 cs = new Class1();

protected void Button1\_Click(object sender, EventArgs e)

{

if(DropDownList1.SelectedItem.Text=="User")

{

string qry = "select \* from batch6 where Select Type='"+DropDownList1+"' EmailId='" + txtuname.Text+ "' and Password='" + txtpass.Text+ "'";

DataSet ds = cs.select(qry);

if (ds.Tables[0].Rows.Count > 0)

{

Session["Id"] = ds.Tables[0].Rows[0][0].ToString();

Session["Select type"] = ds.Tables[0].Rows[0][1].ToString();

Session["Email"] = ds.Tables[0].Rows[0][2].ToString();

Session["Password"] = ds.Tables[0].Rows[0][3].ToString();

Response.Redirect("~/User/UserHome.aspx");

}

else

{

Response.Write("<script>alert('Invalid Login')</script>");

txtuname.Text = txtpass.Text = "";

}

}

else if (DropDownList1.SelectedItem.Text == "Admin")

{

if(txtuname.Text=="admin"&&txtpass.Text=="admin")

{

Response.Redirect("~/Admin/AdminHome.aspx");

}

}

}

}

**6.2.3 Forget Password**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

using System.Data.SqlClient;

public partial class ForgotPwd : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

Sql obj = new Sql();

protected void Button1\_Click(object sender, EventArgs e)

{

try

{

string qry = "select Password from Regi where Email='" + TextBox1.Text + "'";

DataSet ds = obj.select(qry);

if (ds.Tables[0].Rows.Count > 0)

{

MailMessage msg = new MailMessage();

msg.From = new MailAddress("mayorpharmacy3@gmail.com");

msg.IsBodyHtml = true;

msg.To.Add(new MailAddress(TextBox1.Text));

msg.Subject = "A Mail Regarding Password";

msg.Body = "Your Password from Blood Donor Tarcker Website is : " + ds.Tables[0].Rows[0][0].ToString();

SmtpClient smtp = new SmtpClient("smtp.gmail.com", 587);

NetworkCredential nc = new NetworkCredential("mayorpharmacy3@gmail.com", "pharmacy333");

smtp.UseDefaultCredentials = false;

smtp.Credentials = nc;

smtp.EnableSsl = true;

smtp.Send(msg);

Response.Write("<script>alert('Mail Sent Successfully')</script>");

TextBox1.Text = "";

}

else

{

//lblmsg.Text = "MailId Doesn't Exist";

Response.Write("<script>alert('MailId not existed')</script>");

}

}

catch (Exception)

{

//lblmsg.Text = "Failure in sending Mail";

Response.Write("<script>alert('Failure in sending Mail')</script>");

}

}

}

**6.2.4 Gallery Code**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

public partial class Gallery : System.Web.UI.Page

{

Class1 obj = new Class1();

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

bind();

}

}

public void bind()

{

string qry = "select \* from Table\_2";

DataSet ds = obj.select(qry);

if (ds.Tables[0].Rows.Count > 0)

{

DataList1.DataSource = ds;

DataList1.DataBind();

}

else

{

Response.Write("<script>alert('No Queries Present')</script>");

}

}

protected void DataList1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

}

**6.2.5 Book Event**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

public partial class User\_BookEvent : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

string qry = "select \* from Regi where EmailId = '"+ Session["id"].ToString() +"' and Password = '"+ Session["pwd"].ToString() +"'";

DataSet ds = obj.Display(qry);

if (ds.Tables[0].Rows.Count > 0)

{

Session["fname"] = ds.Tables[0].Rows[0][1].ToString();

Session["mno"] = ds.Tables[0].Rows[0][6].ToString();

Session["add"] = ds.Tables[0].Rows[0][7].ToString();

}

}

Sql obj = new Sql();

protected void DropDownList2\_SelectedIndexChanged(object sender, EventArgs e)

{

try

{

string qry = "select \* from Events where EventName = '" + DropDownList1.SelectedItem.Text + "' and Type = '" + DropDownList2.SelectedItem.Text + "'";

DataSet ds = obj.Display(qry);

if (ds.Tables[0].Rows.Count > 0)

{

TextBox1.Text = ds.Tables[0].Rows[0][3].ToString();

Image1.ImageUrl = ds.Tables[0].Rows[0][4].ToString();

TextBox3.Text = ds.Tables[0].Rows[0][5].ToString();

}

}

catch (Exception ex)

{

Response.Write(ex.Message);

}

}

protected void Button1\_Click(object sender, EventArgs e)

{

try

{

string qry = "insert into Booking values('" + Session["fname"].ToString() + "','" + Session["id"].ToString() + "','" + Session["mno"].ToString() + "','" + Session["add"].ToString() + "','"+ DropDownList1.SelectedItem.Text +"','"+ DropDownList2.SelectedItem.Text +"','"+ TextBox1.Text +"','"+ TextBox3.Text +"','Not Accepted')";

int i = obj.InUpDel(qry);

if (i == 1)

{

Response.Write("<script>alert('Event Booked Sucessfully...!!!')</script>");

TextBox1.Text = TextBox3.Text = "";

}

else

{

Response.Write("<script>alert('Event not Booked...!!!')</script>");

}

}

catch (Exception ex)

{

Response.Write(ex.Message);

}

}

}

**6.2.6 Status**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

public partial class User\_Status : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

Bind();

}

Sql obj = new Sql();

void Bind()

{

string qry = "select \* from Booking where EmailId = '" + Session["id"].ToString() + "'";

DataSet ds = obj.Display(qry);

if (ds.Tables[0].Rows.Count > 0)

{

GridView1.DataSource = ds;

GridView1.DataBind();

}

else

{

Response.Write("<script>alert('Booking Status is not Available...!!!')</script>");

}

}

}

**6.2.7 Feedback**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

public partial class User\_Feedback : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

Sql obj = new Sql();

protected void ImageButton1\_Click(object sender, ImageClickEventArgs e)

{

try

{

string qry = "insert into Feedback values('" + TextBox1.Text + "','" + TextBox2.Text + "','" + TextBox3.Text + "')";

int i = obj.InUpDel(qry);

if (i == 1)

{

Response.Write("<script>alert('Feedback Submitted Sucessfully...!!!')</script>");

TextBox1.Text = TextBox2.Text = TextBox3.Text = "";

}

else

{

Response.Write("<script>alert('Feedback Not Submitted...!!!')</script>");

}

}

catch (Exception ex)

{

Response.Write(ex.Message);

}

}

}

**6.2.8 Add Event**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.IO;

public partial class Admin\_AddEvent : System.Web.UI.Page

{

Sql obj = new Sql();

protected void Page\_Load(object sender, EventArgs e)

{

}

protected void DropDownList1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

protected void Button1\_Click(object sender, EventArgs e)

{

try

{

if (FileUpload1.HasFile)

{

string ext = Path.GetExtension(FileUpload1.PostedFile.FileName);

if (ext.ToUpper() == ".PNG" || ext.ToUpper() == ".JPEG" || ext.ToUpper() == ".JPG")

{

string path = "~/Images/" + FileUpload1.PostedFile.FileName;

string qry = "insert into Events values('" + DropDownList1.SelectedItem.Text + "','" + DropDownList2.SelectedItem.Text + "','" + TextBox1.Text + "','" + path + "','" + TextBox3.Text + "')";

int i = obj.InUpDel(qry);

if (i == 1)

{

FileUpload1.SaveAs(Server.MapPath(path));

Response.Write("<script>alert('Event Added Sucessfully...!!!')</script>");

TextBox1.Text = TextBox3.Text = "";

}

else

{

Response.Write("<script>alert('Event is not Added...!!!')</script>");

}

}

else

{

Response.Write("<script>alert('Please Enter only Image file having extension .JPEG , .PNG or .JPG')</script>");

}

}

else

{

Response.Write("<script>alert('Please select the Image...!!!')</script>");

}

}

catch (Exception ex)

{

Response.Write(ex.Message);

}

}

}

**6.2.9 Upload**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

public partial class Admin\_Upload : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

protected void AjaxFileUpload1\_UploadComplete(object sender, AjaxControlToolkit.AjaxFileUploadEventArgs e)

{

Sql obj = new Sql();

string filename = System.IO.Path.GetFileName(e.FileName);

string path = "~/Img/" + filename;

AjaxFileUpload1.SaveAs(Server.MapPath(path));

string qry = "insert into Gallery values('" + path + "')";

int i = obj.InUpDel(qry);

if (i > 0)

{

lblmsg.Text = "Images Inserted Succesfully";

//Response.Write("<script>alert('Image Uploaded Sucessfully...')</script>");

}

else

{

lblmsg.Text = "Image Not Uploaded";

// Response.Write("<script>alert('Image Not Uploaded...')</script>");

}

}

}

**6.2.10 Booking Details**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

public partial class Admin\_BookHistory : System.Web.UI.Page

{

Sql obj = new Sql();

protected void Page\_Load(object sender, EventArgs e)

{

try

{

BookingDetails();

}

catch (Exception ex)

{

Response.Write(ex.Message);

}

}

void BookingDetails()

{

string qry = "select \* from Booking";

DataSet ds = obj.Display(qry);

if (ds.Tables[0].Rows.Count > 0)

{

//Session["fname"] = ds.Tables[0].Rows[0][1].ToString();

//Session["email"] = ds.Tables[0].Rows[0][2].ToString();

GridView1.DataSource = ds;

GridView1.DataBind();

}

else

{

Response.Write("<script>alert('Booking Dteails Not Available...!!!')</script>");

}

}

protected void Button1\_Click(object sender, EventArgs e)

{

try

{

Button btn = (Button)sender;

GridViewRow gr = (GridViewRow)btn.NamingContainer;

Label lbl1 = (Label)gr.FindControl("Label1");

string qry = "update Booking set Status ='Accepted' where Id = '" + lbl1.Text + "'";

int i = obj.InUpDel(qry);

if (i == 1)

{

Response.Write("<script>alert('Status Updated Sucessfully...!!!')</script>");

BookingDetails();

}

else

{

Response.Write("<script>alert('Status Not Updated...!!!')</script>");

}

}

catch (Exception ex)

{

Response.Write(ex.Message);

}

}

}

**6.2.11 Feedback**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data;

public partial class Admin\_Feedback : System.Web.UI.Page

{

Sql obj=new Sql();

protected void Page\_Load(object sender, EventArgs e)

{

try

{

Feedback();

}

catch (Exception ex)

{

Response.Write(ex.Message);

}

}

void Feedback()

{

string qry = "select \* from Feedback";

DataSet ds = obj.Display(qry);

if (ds.Tables[0].Rows.Count > 0)

{

GridView1.DataSource = ds;

GridView1.DataBind();

}

else

{

Response.Write("<script>alert('Feedback Data not Available...!!!')</script>");

}

}

}

**CHAPTER 7**

**TESTING**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**7.1 FEATURES TO BE TESTED**

**DATABASE:**

Data consistency is very important in web applications. Check for the data integrity and errors while we do any data base related functionality. Check if all the data base queries are executing correctly, data is retrieved correctly and also updated correctly.

**UNIT TESTING:**

Unit testing focuses verification effort on the smallest unit of software design that is the module. Using procedural design description as a guide, important control paths are tested to uncover errors within the boundaries of the module. The unit test is normally white box testing oriented and the step can be conducted in parallel for multiple modules.

**SYSTEM TESTING:**

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different purpose, all work to verify that all system elements have been properly integrated to perform allocated functions.

**SECURITY TESTING:**

Attempts to verify the protection mechanisms built into the system.

**PERFORMANCE TESTING:**

This method is designed to test runtime performance of software within the context of an integrated system.

**7.2 FEATURES NOT TO BE TESTED**

**TESTING’S DONE AT SERVER SIDE:**

The server side tests are not done as this project is a sample representation of the web project and this is an stand alone IEEE project.

**INTEGRATION TESTING:**

Integration testing is a systematic technique for constructing the program structure, while conducting test to uncover errors associated with the interface. The objective is to take unit tested methods and build a program structure that has been dictated by design.

**TOP-DOWN INTEGRATION:**

Top down integrations is an incremental approach for construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main control program. Modules subordinate to the main program are incorporated in the structure either in the breath-first or depth-first manner.

**BOTTOM-UP INTEGRATION:**

This method as the name suggests, begins construction and testing with atomic modules i.e., modules at the lowest level. Because the modules are integrated in the bottom up manner the processing required for the modules subordinate to a given level is always available and the need for stubs is eliminated.

**VALIDATION TESTING:**

At the end of integration testing software is completely assembled as a package. Validation testing is the next stage, which can be defined as successful when the software functions in the manner reasonably expected by the customer. Reasonable expectations are those defined in the software requirements specifications. Information contained in those sections form a basis for validation testing approach.

**7.3 TESTING TOOLS AND ENVIRONMENT**

**BLACK BOX TESTING:**

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

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

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

**WHITE BOX TESTING:**

In this test cases are generated on the logic of the each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It has been uses to generate the test cases in the following cases.

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

**7.4 DESIGN OF TEST CASES AND SCENARIOS**

**TEST CASES**

A test case is a set of conditions or variables and inputs that are developed for a particular goal or objective to be achieved on a certain application to judge its capabilities or features.

It might take more than one test case to determine the true functionality of the application being tested. Every requirement or objective to be achieved needs at least one test case. Some software development methodologies like Rational Unified Process (RUP) recommend creating at least two test cases for each requirement or objective; one for performing testing through positive perspective and the other through negative perspective.

**TEST CASE STRUCTURE**

A formal written test case comprises of three parts -

**INFORMATION**

Information consists of general information about the test case. Information incorporates Identifier, test case creator, test case version, name of the test case, purpose or brief description and test case dependencies.

**ACTIVITY**  
 Activity consists of the actual test case activities. Activity contains information about the test case environment, activities to be done at test case initialization, activities to be done after test case is performed, and step by step actions to be done while testing and the input data that is to be supplied for testing.

**Chapter 8: RESULT**

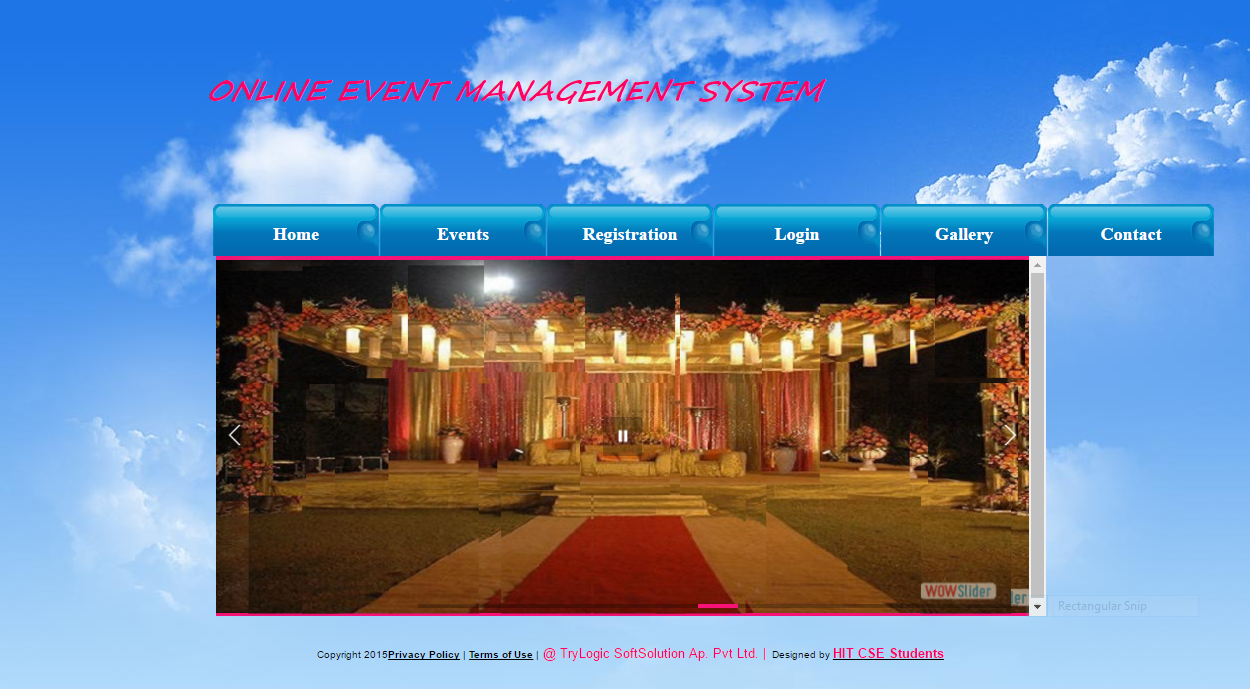
****

Fig 8.1.1 Admin Home page

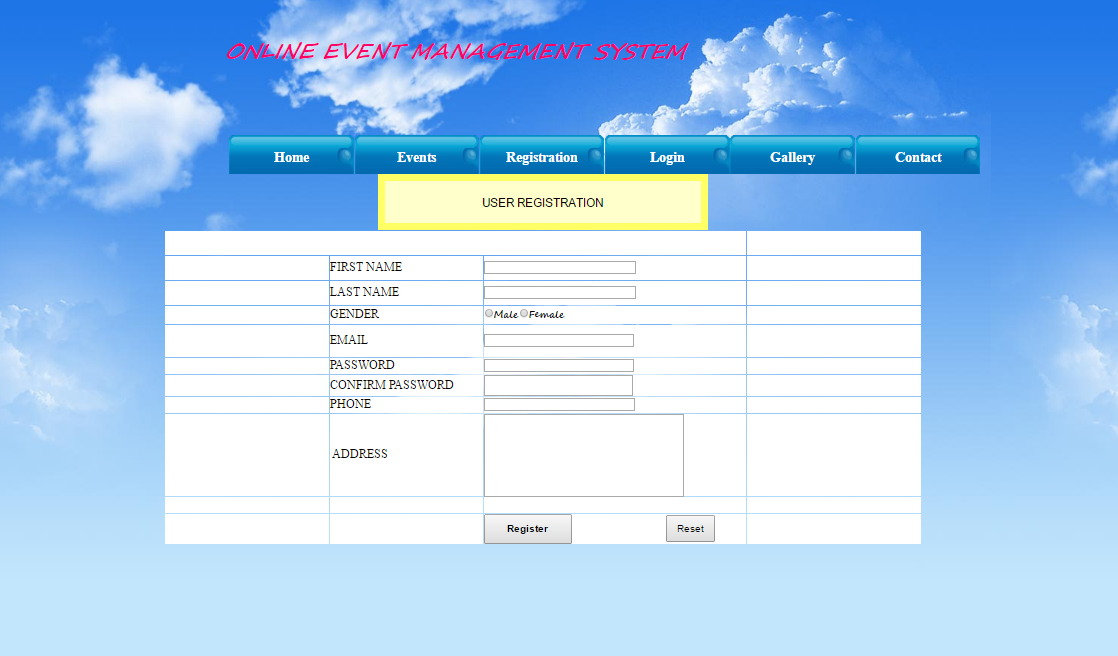


Fig 8.1.2 Registration page

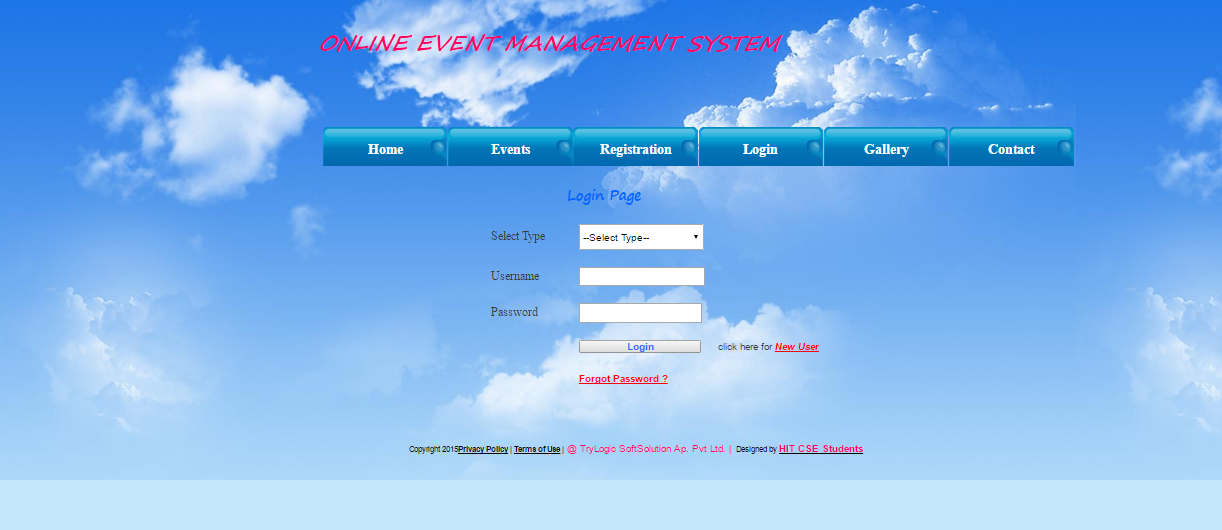
****

Fig8.1.3 Admin login page

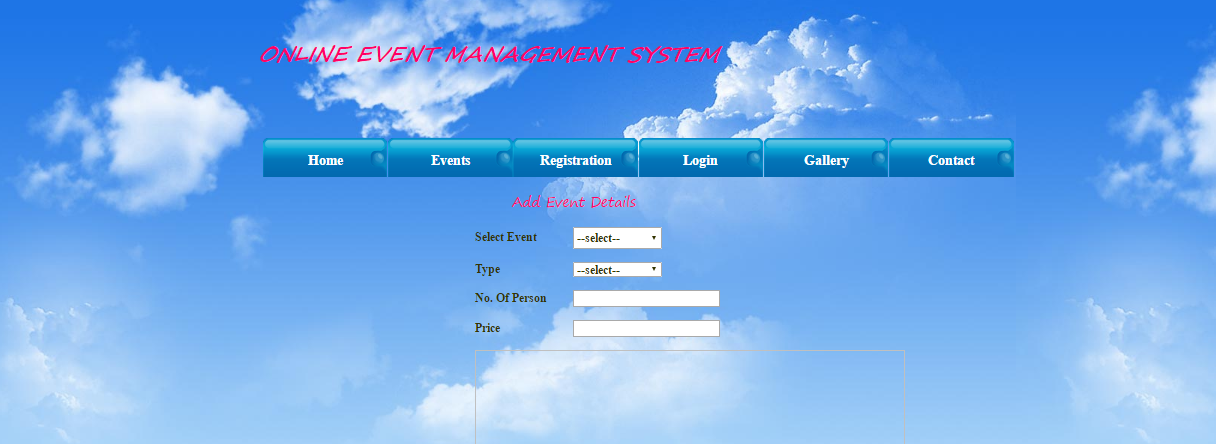
****

Fig 8.1.4 Admin Add Event page

****

Fig 8.1.5 Gallery page

****

Fig 8.1.6 Contact us page

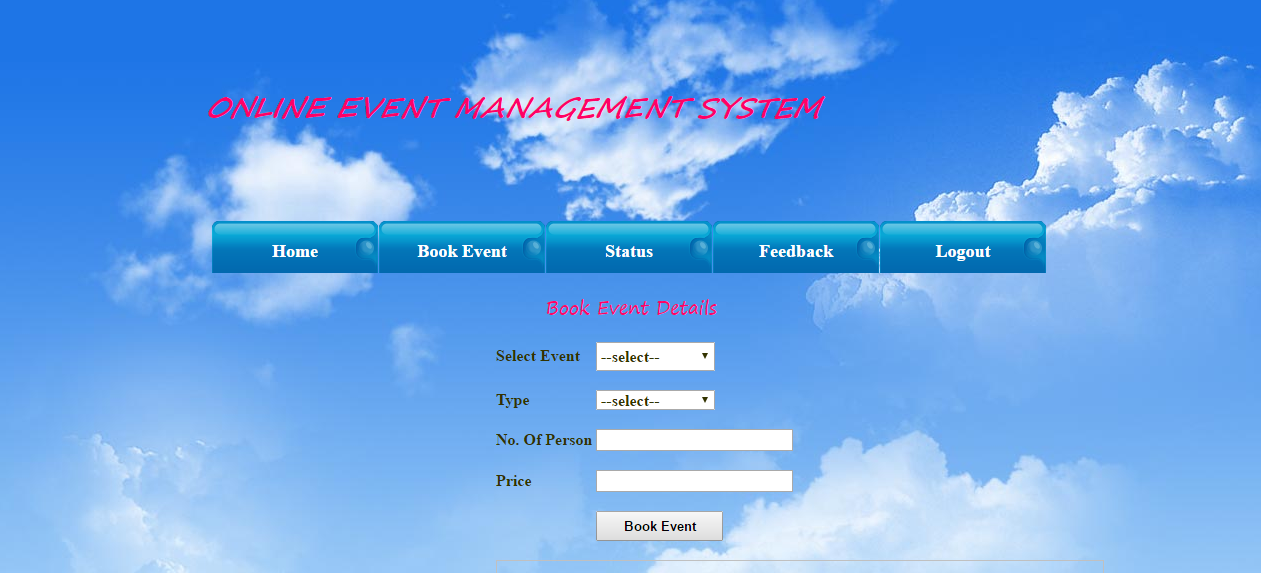
****

Fig8.1.7 User Book Event page

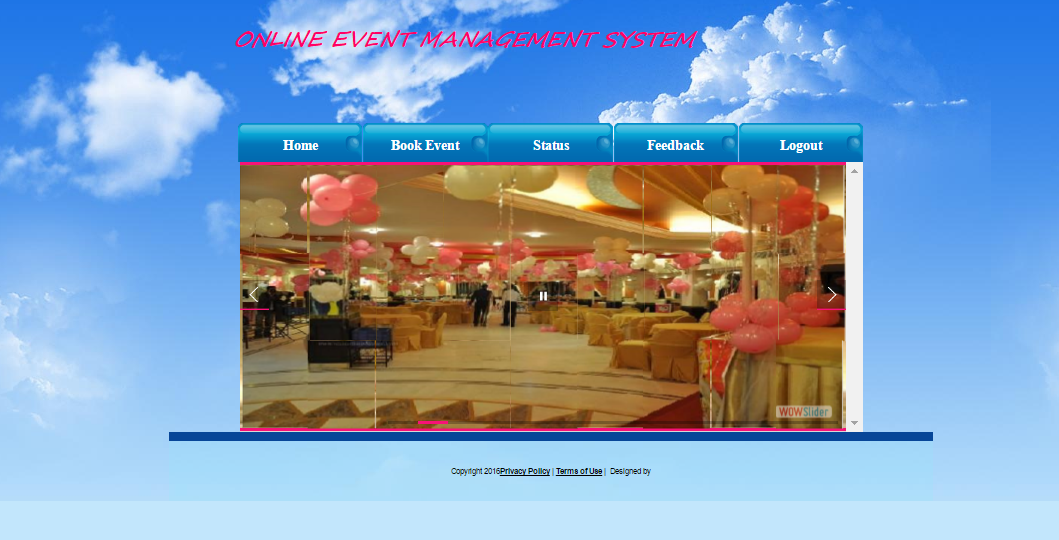
****

Fig 8.1.8 User home page

****

Fig 8.1.9 Booking Status page

****

Fig 8.1.10 Booking details page

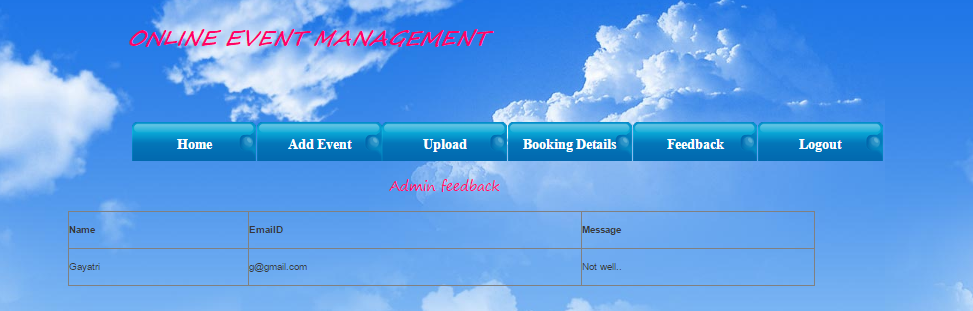
****

Fig 8.1.11 Admin Feedback page

**9. CONCLUSION**

In this project will provide most of the basic functionality required for any event. The purpose of event management is to prepare and organize an event in advance so that the event runs smoothly. By these the outlook of the event is that might be an integral practice for event like marriages, parties, annual functions not only can they lower the operations but also they can rip the benefits of the partnering with relatively related event organizer to enable their event to scale to a greater heights. These type of an online event is useful for many users where ever they want they can booked the event very easily and can take the benefits from the event management.

**10. FUTURE ENHANCEMENT**

* Online Event Management system provides most of the basic functionality required for an event.
* The system stored the data in the database.
* The user gets all the resource at a single place instead of wandering around for these.
* This system is effective and saves time and cost of the users.

**11. BIBLIOGRAPHY**

[1] Applying UML and patterns: An Introduction to object-oriented Analysis and Design by Carig Larman; PHI; Third Edition;ISBN-10:0131489062,ISBN-13:978-0131489066

[2] UML Distilled:A brief Guide to standard object modeling by Fowler;2011 Edition(2007),Perason;ISBN-10:8131715655,ISBN-13:978-8131715659

[3] Database Modeling and Design, by Toby J.Teorey ,Sam S.Lightstone, Tom Nadeau;5th Edition;Morgan Kaufmann;ISBN-10:0123820200,ISBN-13:978-0123820204

[4] Database Design using Entity-Relationship Diagrams by Richard Earp;2nd Edition, April 16,2007;Auerbach Publications;ISBN-10:1439861765,ISBN-13:978-1439861769

[5] Structured systems Analysis and Design: Data flow Approach, by V.B Kaujalgi,2nd Edition ;Orient Blackswan;ISBN-10:0863113230,ISBN-13:978-0863113239

[6] Software Testing, by Ron patton;2nd Edition; Sams publishing;ISBN-10:0672327988,ISBN-13:978-0672327988

[7] Software Testing concepts and Tools, by NageshwarRao Pusluri;2nd Edition ;Dreamtech press;ISBN-10:8177227122,ISBN-13:978-8177227123

[8] Professional ASP.NET by Jon Galloway, K.ScottAllen ,David Maison;1st Edition; Wrox Publisher;ISBN-10:1118794753,ISBN-13:978-1118794753

[9]C# in Depth, By Jon Skeet;3rd Edition; Manning publications;ISBN-10:161729134,ISBN-13:978-1617291340

[10] The Unified Modeling Language User Guide, by Grady Booch, James Rumbaugh; 2nd Edition; Addison-Wesley Professional; ISBN-10: 0321267974, ISBN-13: 978-0321267979

[11] Concurrent Programming on Windows**,** by Joe Duffy**;** 1st Edition**;** Addison-Wesley Professional**;** ISBN-10: 032143482X,ISBN-13: 978-0321434821