Overall Design of “Event Management System”

# N- Tier Projects Structure:

## **Presentation Layer****: -**

**EventManagement -** For interacting with the application user

## **Business Access Layer: -**

**BusinessLogic** - To provide service and data to PL

**Business Object Layer: -**

**BusinessObjects** - To create business object for BAL

**Data Access Layer: -**

**DataAccess** - To handle data access read and write operations for BAL

**Data Base:-**

**Sql server 2012 -** DB with required tables and stored procedures required for this application

# Project Reference:

Within the solution, the project references work like this:

**EventManagement**

| \

| \

| \

**BusinessLogic** – – – – **BusinessObjects**

| /

| /

| /

**DataAccess /**

**|**

**|**

**Database**

**Classes by Layer:**

**EventManagement****: -**

SplashWindow.xaml –

1. This page is the Start of the Event management application

LoginWindow.xaml

1. This page allow user access to the Event management application

MainWindow.xaml

1. This page has 3 tabs which gives admin the over view of Customer list, Event booking list and User’s list

CustomerWindow.xaml –

1. Represents the view for Customers who will use Event Management Window
2. Can perform CRUD operations for adding Customers

CustomerWindow.xaml.cs –

1. The code behind page for CustomerWindow.xaml
2. Data binding and data fetching of customers profile are done through this window

EventBooking.xaml –

1. Represents the view for Event booking Window
2. Can perform CRUD operations to add different events for customers present in the system

EventBooking.xaml.cs –

1. Represents the code behind page for EventBooking.xaml
2. Data binding and data fetching of assigning and booking events for customers is done through this page

UserProfileWindow.xaml –

1. Represents the view for User Profile Window
2. Can perform CRUD operations for adding users of different roles

UserProfileWindow.xaml.cs –

1. The code behind page for UserProfileWindow.xaml
2. Data binding and data fetching of user profile are done through this window

**BusinessLogic: -**

UserProfileManager.cs –

This class is used for all business logic related to user profile

1. Fetching user details list
2. CRUD operations of users profile

CustomerManager.cs –

This class is used for all business logic related to Customer of Event management application

1. Fetching Customers list
2. CRUD operations of customers

EventBookingManager.cs –

This class is used for all business logic related to Event bookings this of application

1. Fetching Events list
2. CRUD operations of assigning events to customers etc.

SecurityManager.cs –

This class is used for authentication and authorization of users before logging into the application

HashMethods.cs

This class is used to encrypt and decrypt passwords of users within the system.

**BusinessObjects: -**

AccessToken.cs

Caterers.cs

Customer.cs

Enumerations.cs

EventBookings.cs

EventBookingsO.cs

EventStatus.cs

EventType.cs

Payment.cs

Roles.cs

UserProfile.cs

Venue.cs

**DataAccess: -**

CaterersAccessor.cs

CustomerAccessor.cs

DBConnection.cs

EventBookingsAccessor.cs

EventBookingsAccessorO.cs

EventTypeAccessor.cs

RoleAccessor.cs

UserProfileAccessor.cs

VenueAccessor.cs

**Business Logic Flow by Classes:**

**User:-**

List of all User

Browse a List of User

Select a User to see information

Edit a selected user information

Add a new user

Delete a user

**Customer:-**

List of all Customer

Browse a List of Customer

Select a Customer to see information

Edit a selected customer information

Add a new customer

Delete a customer from existing list

**Event Booking:**

List of all Event Bookings

Browse a List of Event bookings

Select event booking to see information

Edit a selected event booking information

Add a new event booking

**Methods by Classes:**

CustomerWindow.xaml.cs

**Private functions**

1. private void grdCustomerList\_SelectionChanged(object sender, SelectionChangedEventArgs e)

Function called on selection of row from customer list grid

1. private void btnGetCustomerData\_Click(object sender, RoutedEventArgs e)

Function called to get customer data

1. private void btnClose\_Click(object sender, RoutedEventArgs e)

Function called to close the current window

1. private void btnDeleteCustomer\_Click(object sender, RoutedEventArgs e)

Function to delete customer

1. private void DisplayCustomerData()

Function called internally to bind customer data grid

1. private void CleanEntryForm()

Function called internally to clean the form below the grid

1. private void FillEntryForm(Customer cust)

Function called to fill form based on grid row selection

1. private void EnableDisableForm(bool isEnable)

Function called to enable or disable entry form

1. private void btnSave\_Click(object sender, RoutedEventArgs e)

Function called to save new or edited customers data

1. private void btnCancel\_Click(object sender, RoutedEventArgs e)

Function called to cancel the current user action

1. private void btnEditCustomer\_Click(object sender, RoutedEventArgs e)

Function called on click of edit button on the page, this is used enable form entry

The above functions are also used with different business objects on “EventBookings.xaml.cs” and “UserProfile.xaml.cs”.

CustomerManager.cs

**Public Functions**

1. public List<Customer> GetCustomerList()

Function to get customer list on page load of Customer window

1. public Customer GetCustomerById(int customerId)

Function to get a particular customers data

1. public bool InsertNewCustomer(string firstName,string lastName,string emailID,string phoneNO1,string phoneNO2, string address1, string address2, string postalCode, string city, string state)

Function to insert customer details

1. public bool UpdateCustomer(Customer customer)

Function to update customer details

1. public bool DeleteCustomer(Customer customer)

Function to delete customer details

Similar functions are used in “EventBooking.cs” and “UserProfile.cs” class to perform similar functionality

CustomerAccessor.cs

**Public static functions**

1. public static List<Customer> FetchCustomerList()

Function used to access Database and fetch customer list

1. public static Customer FetchCustomerByID(int customerID)

Function used to access database and fetch a particular customers data based on customer id

1. public static int InsertCustomer(Customer customer)

Function used to access database to insert Customer details based on customer business object

1. public static int UpdateCustomer(Customer customer)

Function used to access database to update Customer details based on customer business object

1. public static int DeleteCustomer(int customerID)

Function used to access database to delete a customer based on customer id

Similar functionality is available in “EventBookingAccessor.cs” and “UserProfileAccessor.cs”