

ABSTRACT

Event management is a process of organizing a professional and focused event, for a particular target audience. It involves visualizing concepts, planning, budgeting, organizing and executing events such as wedding, musical concerts, corporate seminars, exhibitions, birthday celebrations, theme parties, and many more.

An Event management system software project serves the functionality of an event manager. The system allow registered user login and new user are allowed to register on the application. The system helps in the management of events, users and the aspects related to them. This proposed to be a web application. The project provides most of the basic functionality required for an event type e.g. [Marriage, Dance Show Birthday party, College Festival, etc.], the system then allows the user to select date and time of event, place and the event equipment. All the data is logged in the database and the user is given a receipt number for his booking. The data is then sent to administrator (website owner) and they may interact with the client as per his requirement.

TABLE OF CONTENTS

Chapter No.	Contents	Page No.
	Acknowledgement	I
	Abstract	II
	Table of Contents	III
	List of Figures	V
Chapter 1	INTRODUCTION	1
1.1	Overview	1
1.2	Problem statement	1
1.3	Databse management system	2
1.4	SQL	2
1.5	HTML / Javascript	3
1.6	Php Connections	3
Chapter 2	Requirements Specification	4
2.1	Overall Description	4
2.2	Specific Requirements	4
2.2.1	Software Requirements	4
2.2.2	Hardware Requirements	4
2.2.3	Technology	5
Chapter 3	Detailed Design	6
3.1	System Design	6
3.2	Entity Relationship Diagram	8
3.3	Relational Schema	9
3.4	Description Of Tables	10
Chapter 4	Implementation	12
4.1	Module And Their Roles	12
4.2	Stored Procedure	15
4.3	Result	15
Chapter 5	Testing	16

5.1	Software Testing	16
5.2	Module Testing And Integration	16
5.3	Limitations	16
Chapter 6	SNAP SHOTS	17
6.1	Login Page	17
6.2	Registration Page	17
6.3	Home Page	18
6.4	Event Package List	18
6.5	User Event Booking	19
6.6	Event History Of The User	19
6.7	Admin Login Page	20
6.8	Admin Dashboard Page	20
6.9	Admin Package Creation Page	21
6.10	Admin Manage Booking Page	21
Chapter 7	Conclusion	22
Chapter 8	Future Enhancements	23
	References	24

LIST OF FIGURES

Figure No.	Figure Name	Page No.
3.1	JSP Architecture	6
3.2	ER diagram of Event Management System	8
3.3	Schema diagram	9

INTRODUCTION

1.1 OVERVIEW

Event management system is developed for managing the needs of the customers on an Online platform. It has been developed to override the problems prevailing in the practicing manual systems.

As people spend lots of money on weddings, parties, and other events by involving themselves in each and every affair in such away that, at the end of the day they feel that they have not seen the events or could not enjoy the event which took place. That is why, an “Event Management System” is required to make people comfortable on the day of the event.

1.2 PROBLEM STATEMENT

The main aim of “event management system” is to make an easy interface for customers. To bring all factors together to produce a workable event.

1.3 DATABASE MANAGEMENT SYSTEM

A database management system (DBMS) is system software for creating and managing databases. The DBMS provides users and programmers with a systematic way to create, retrieve, update and manage data. The DBMS essentially serves as an interface between the database and end users application programs, ensuring that data is consistently organized and remains easily accessible.

The DBMS manages three important things: the data, the database engine that allows data to be accessed, locked and modified, and the database schema, which defines the database’s logical structure. These three foundational elements help to provide concurrency, security, data integrity and uniform administration procedures. Typical database administration tasks supported by the DBMS include change management, performance monitoring/tuning and backup and recovery. Many database management systems are also responsible for automated rollbacks, restarts and recovery as well as the logging and auditing of activity.

1.4 SQL

SQL is a standard language for storing, manipulating and retrieving data in databases. Originally based upon relational algebra and tuple relational calculus, SQL consists of a data definition language, data manipulation language, and data control language. The scope of SQL includes data insert, query, update and delete, schema creation and modification, and data access control.

SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987.[13] Since then, the standard has been revised to include a larger set of features. Despite the existence of such standards, most SQL code is not completely portable among different database systems without adjustments.

1.5 HTML / JavaScript

HTML is a markup language used for structuring and presenting content on the web and the fifth and current major version of the HTML standard. HTML5 includes detailed processing models to encourage more interoperable implementations; it extends, improves and rationalizes the markup available for documents, and introduces markup and application programming interfaces (APIs) for complex web applications.

JavaScript often abbreviated as JS, is a high-level, interpreted programming language. It is a language which is also characterized as dynamic, weakly typed, prototype-based and multi-paradigm.

Alongside HTML and CSS, JavaScript is one of the three core technologies of the World Wide Web. JavaScript enables interactive web pages and thus is an essential part of web applications. The vast majority of websites use it, and all major web browsers have a dedicated JavaScript engine to execute it.

1.6 PHP CONNECTIONS

The database is one of the important components of any programming language. To deal with a dynamic project and the data management, database is required. PHP supports various kinds of database connections with it. MySQL is one of the most widely used relational databases and it is mostly used with PHP as well. Considering the term database connection in PHP, MySQL itself have various way to make connections in an application to play with the database operations. After making the connection of PHP-MYSQL various things can be done like – insertion of records; deletion of records; updating etc.

REQUIREMENTS SPECIFICATION

A computerized way of handling information about property and users' details is efficient, organized and time saving, compared to a manual way of doing so. This is done through a database driven web application whose requirements are mentioned in this section.

2.1 OVERALL DESCRIPTION

A reliable and scalable database driven web application with security features that is easy to use and maintain is the requisite.

2.2 SPECIFIC REQUIREMENTS

The specific requirements of the event management System are stated as follows:

2.2.1 SOFTWARE REQUIREMENTS

- ☐ Front end – html, CSS, JavaScript, bootstrap.
- ☐ Web Browser – Firefox 50 or later, Google Chrome 60 or later
- ☐ Database support - MySQL
- ☐ Operating system – Windows 10

2.2.2 HARDWARE REQUIREMENTS

- ☐ Processor – Intel core i5
- ☐ RAM – 2 GB or more
- ☐ Hard disk – 1 TB
- ☐ Monitor – VGA of 1024x768 screen resolution
- ☐ Keyboard and Mouse

2.2.3 TECHNOLOGY

- HTML is used for the front end design. It provides a means to structure text based information in a document. It allows users to produce web pages that include text, graphics and hyperlinks.
- CSS (Cascading Style Sheets) is a style sheet language used for describing the presentation of a document written in a markup language. Although most often used to set the visual style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any XML document.
- SQL is the language used to manipulate relational databases. It is tied closely with the relational model. It is issued for the purpose of data definition and data manipulation.
- Java Server pages is a simple yet powerful technology for creating and maintaining dynamic-content web pages. It is based on the Java programming language. It can be thought of as an extension to servlet because it provides more functionality than servlet. A JSP page consists of HTML tags and JSP tags. The jsp pages are easier to maintain than servlet because we can separate designing and development.
- We require a JDBC connection between the front end and back end components to write to the database and fetch required data.

DETAILED DESIGN

3.1 SYSTEM DESIGN

The web server needs a JSP engine, i.e., a container to process JSP pages. The JSP container is responsible for intercepting requests for JSP pages. A JSP container works with the Web server to provide the runtime environment and other services a JSP needs. It knows how to understand the special elements that are part of JSPs. This server will act as a mediator between the client browser and a database.

The following diagram shows the JSP architecture.

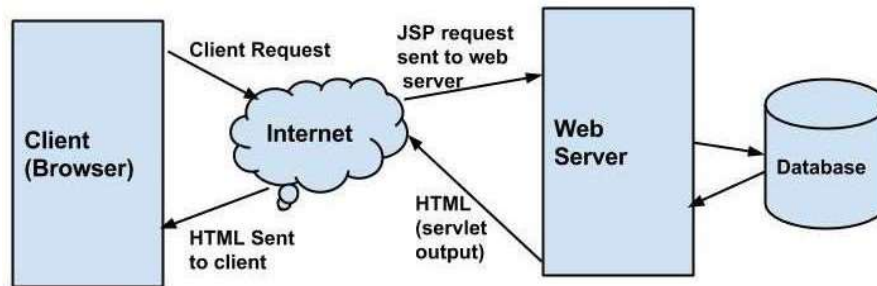


Fig. 3.1: JSP Architecture

Three-tier Client / Server database architecture is commonly used architecture for web applications. Intermediate layer called Application server or Web Server stores the web connectivity software and the business logic (constraints) part of application used to access the right amount of data from the database server. This layer acts like medium for sending partially processed data between the database server and the client. Database architecture focuses on the design, development, implementation and maintenance of computer programs that store and organize information for businesses, agencies and institutions. A database architect develops and implements software to meet the needs of users. Several types of databases, including relational or multimedia, may be created. Additionally, database architects may use one of several languages to create databases, such as structured query language.

3.2 ENTITY RELATIONSHIP DIAGRAM

An entity–relationship model is usually the result of systematic analysis to define and describe what is important to processes in an area of a business.

An E-R model does not define the business processes; it only presents a business data schema in graphical form. It is usually drawn in a graphical form as boxes (entities) that are connected by lines (relationships) which express the associations and dependencies between entities.

Entities may be characterized not only by relationships, but also by additional properties (attributes), which include identifiers called "primary keys". Diagrams created to represent attributes as well as entities and relationships may be called entity-attribute-relationship diagrams, rather than entity-relationship models.

An ER model is typically implemented as a database. In a simple relational database implementation, each row of a table represents one instance of an entity type, and each field in a table represents an attribute type. In a relational database a relationship between entities is implemented by storing the primary key of one entity as a pointer or "foreign key" in the table of another entity.

There is a tradition for ER/data models to be built at two or three levels of abstraction. Note that the conceptual-logical-physical hierarchy below is used in other kinds of specification, and is different from the three schema approach to software engineering. While useful for organizing data that can be represented by a relational structure, an entity-relationship diagram can't sufficiently represent semi-structured or unstructured data, and an ER Diagram is unlikely to be helpful on its own in integrating data into a pre-existing information system.

Cardinality notations define the attributes of the relationship between the entities. Cardinalities can denote that an entity is optional.

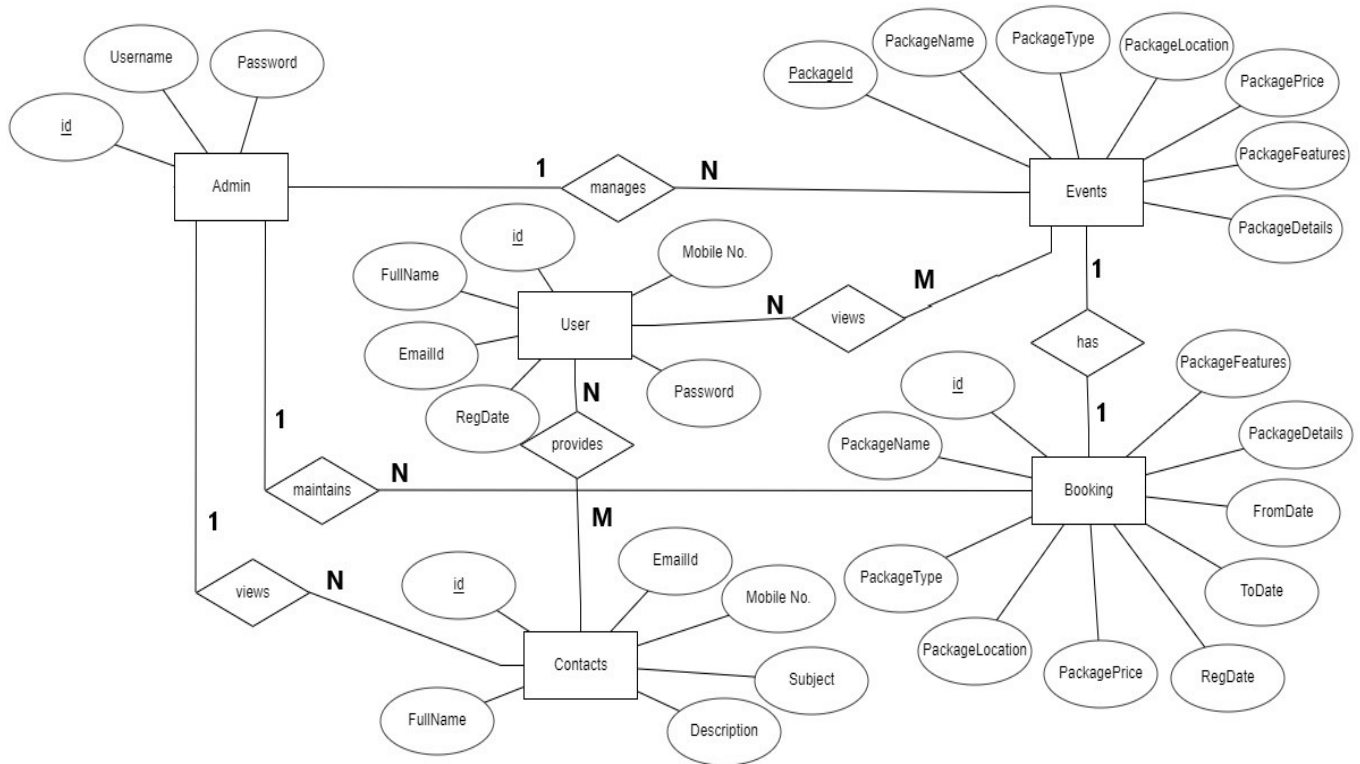


Fig.3.2 ER diagram of Event Management System

3.3 RELATIONAL SCHEMA

The term "schema" refers to the organization of data as a blueprint of how the database is constructed. The formal definition of a database schema is a set of formulas called integrity constraints imposed on a database. A relational schema shows references among fields in the database. When a primary key is referenced in another table in the database, it is called a foreign key. This is denoted by an arrow with the head pointing at the referenced key attribute. A schema diagram helps organize values in the database. The following diagram shows the schema diagram for the database.

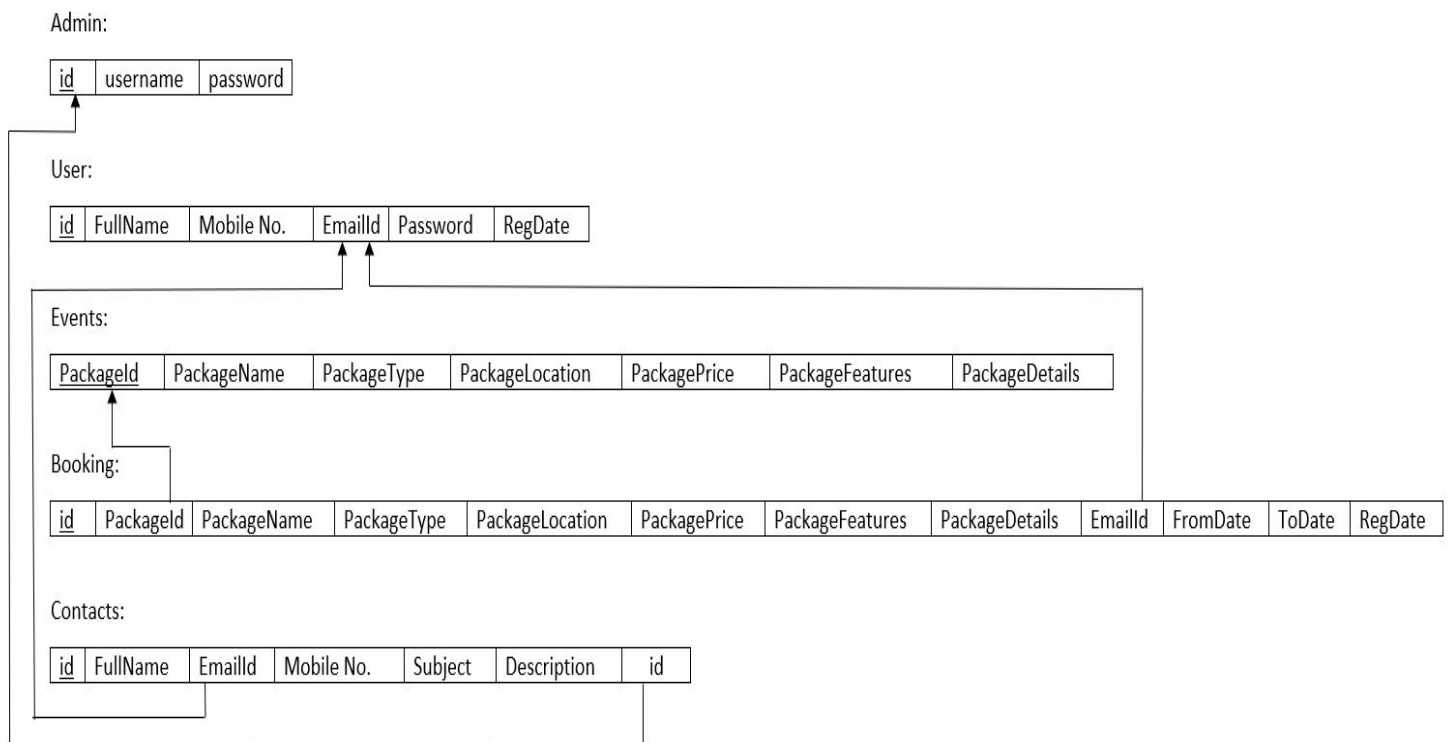


Fig. 3.3: Schema diagram

3.4 DESCRIPTION OF TABLES

The database consists of six tables:

1. Admin: It stores the admin details.
 - ☐ Admin_id: Unique admin id done by auto increment.
 - ☐ Username: Name of the user.
 - ☐ Password: Password associated with admin to login into system

2. User: It stores the user details.
 - ☐ User_id: Unique user id done by auto increment
 - ☐ Fullname: name of the user.
 - ☐ Mobile No: phone number of the user
 - ☐ Email Id: email of the user.
 - ☐ Password: Password associated with user to login into system
 - ☐ Regdate: the users registration date.

3. Events: Stores the information about the events
 - ☐ Package_id: Unique package_id done by auto increment
 - ☐ Package_name: Name of the package
 - ☐ Package_type: Type of the package given to the user.
 - ☐ Package_location: location of the event to take place.
 - ☐ Package_price: price of the event.
 - ☐ Package_features: features of the event.
 - ☐ Package_details: complete details of the event.

4. Booking: It stores the booking details of the events.
 - ☐ Booking_id: Unique Symbol of the company
 - ☐ Email_id: email of the user.
 - ☐ From date: event start date.
 - ☐ To date: event end date.
 - ☐ Regdate: the registration date.
 - ☐ Package_id: Unique package_id done by auto increment
 - ☐ Package_name: Name of the package
 - ☐ Package_type: Type of the package given to the user.
 - ☐ Package_location: location of the event to take place.
 - ☐ Package_price: price of the event.
 - ☐ Package_features: features of the event.
 - ☐ Package_details: complete details of the event

5. Contacts: It stores the contact details of the user.

- ☐ Contact_id: Unique contact_id done by auto increment
- ☐ Fullname: Name of the user.
- ☐ Email_id: email_id of the user.
- ☐ Mobile no: mobile no of the user.
- ☐ Subject: the subject given by the user.
- ☐ Description: the related description provided by the user.

IMPLEMENTATION

4.1 MODULES AND THEIR ROLES

4.1.1 CONFIGURATON CODE:

```
<?php
// DB credentials.
define('DB_HOST','localhost');
define('DB_USER','root');
define('DB_PASS','');
define('DB_NAME','tms');
// Establish database connection.
try
{
$dbh = new PDO("mysql:host=".DB_HOST.";dbname=".DB_NAME,DB_USER,
DB_PASS,array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES 'utf8'"));
}
catch (PDOException $e)
{
exit("Error: " . $e->getMessage());
}
?>
```

4.1.2 ADMIN LOGIN:

```
<?php
session_start();
include('includes/config.php');
if(isset($_POST['login']))
{
$username=$_POST['username'];
$password=md5($_POST['password']);
$sql ="SELECT UserName,Password FROM admin WHERE UserName=:uname and
Password=:password";
$query= $dbh -> prepare($sql);
$query-> bindParam(':uname', $username, PDO::PARAM_STR);
$query-> bindParam(':password', $password, PDO::PARAM_STR);
$query-> execute();
$results=$query->fetchAll(PDO::FETCH_OBJ);
if($query->rowCount() > 0)
{
$_SESSION['alogin']=$_POST['username'];
```

```

        echo "<script type='text/javascript'> document.location = 'dashboard.php'; </script>";
    }else{
        echo "<script>alert('Invalid Details');</script>";
    }
}
?>

```

4.1.3 USER SIGN UP:

```

<?php
error_reporting(0);
if(isset($_POST['submit']))
{
    $fname=$_POST['fname'];
    $mnumber=$_POST['mobilenumber'];
    $email=$_POST['email'];
    $password=md5($_POST['password']);
    $sql="INSERT INTO tblusers(FullName,MobileNumber,EmailId>Password)
VALUES(:fname,:mnumber,:email,:password)";
    $query = $dbh->prepare($sql);
    $query->bindParam(':fname',$fname,PDO::PARAM_STR);
    $query->bindParam(':mnumber',$mnumber,PDO::PARAM_STR);
    $query->bindParam(':email',$email,PDO::PARAM_STR);
    $query->bindParam(':password',$password,PDO::PARAM_STR);
    $query->execute();
    $lastInsertId = $dbh->lastInsertId();
    if($lastInsertId)
    {
        $_SESSION['msg']="You are Scuccessfully registered. Now you can login ";
        header('location:thankyou.php');
    }
    else
    {
        $_SESSION['msg']="Something went wrong. Please try again.";
        header('location:thankyou.php');
    }
}
?>

```

4.1.4 PACKAGE CREATION:

```

<?php
session_start();
error_reporting(0);
include('includes/config.php');
if(strlen($_SESSION['alogin'])==0)
{
    header('location:index.php');
}
else{

```

```

if(isset($_POST['submit']))
{
    $pname=$_POST['packagename'];
    $ptype=$_POST['packagetype'];
    $plocation=$_POST['packagelocation'];
    $pprice=$_POST['packageprice'];
    $pfeatures=$_POST['packagefeatures'];
    $pdetails=$_POST['packagedetails'];
    $pimage=$_FILES["packageimage"]["name"];
    move_uploaded_file($_FILES["packageimage"]["tmp_name"],"packageimages/".$_FILES["packageimage"]["name"]);
    $sql="INSERT INTO
tblEventpackages(PackageName,PackageType,PackageLocation,PackagePrice,PackageFeatures,PackageDetails,PackageImage)
VALUES(:pname,:ptype,:plocation,:pprice,:pfeatures,:pdetails,:pimage)";
    $query = $dbh->prepare($sql);
    $query->bindParam(':pname',$pname,PDO::PARAM_STR);
    $query->bindParam(':ptype',$ptype,PDO::PARAM_STR);
    $query->bindParam(':plocation',$plocation,PDO::PARAM_STR);
    $query->bindParam(':pprice',$pprice,PDO::PARAM_STR);
    $query->bindParam(':pfeatures',$pfeatures,PDO::PARAM_STR);
    $query->bindParam(':pdetails',$pdetails,PDO::PARAM_STR);
    $query->bindParam(':pimage',$pimage,PDO::PARAM_STR);
    $query->execute();
    $lastInsertId = $dbh->lastInsertId();
    if($lastInsertId)
    {
        $msg="Package Created Successfully";
    }
    else
    {
        $error="Something went wrong. Please try again";
    }
}
?>

```

4.1.5 USER LOGOUT:

```

<?php
session_start();
$_SESSION = array();
if (ini_get("session.use_cookies")) {
    $params = session_get_cookie_params();
    setcookie(session_name(), "", time() - 60*60,
        $params["path"], $params["domain"],
        $params["secure"], $params["httponly"]
    );
}
unset($_SESSION['login']);
session_destroy(); // destroy session
header("location:index.php");
?>

```

4.2 STORED PROCEDURES

Stored procedure in admin:

```
CREATE DEFINER='root'@'localhost' PROCEDURE `admin` (IN `id` INT(20), IN  
`username` VARCHAR(255), IN `password` VARCHAR(255))  
BEGIN  
INSERT into admin VALUES  
(id, username, password)  
END;
```

4.3 RESULT

The resulting system is able to:

- ☐ Authenticate user credentials during login.
- ☐ Salted encryption for security of user passwords.
- ☐ Register new users and link to their accounts.
- ☐ Allow users to view the list of packages.
- ☐ Allow user to see their individual event history which is being booked.
- ☐ Ability to cancel/confirm the event if they want to.

TESTING

5.1 SOFTWARE TESTING

Testing is the process used to help identify correctness, completeness, security and quality of developed software. This includes executing a program with the intent of finding errors. It is important to distinguish between faults and failures. Software testing can provide objective, independent information about the quality of software and risk of its failure to users or sponsors. It can be conducted as soon as executable software (even if partially complete) exists. Most testing occurs after system requirements have been defined and then implemented in testable programs.

5.2 MODULE TESTING AND INTEGRATION

Module testing is a process of testing the individual subprograms, subroutines, classes, or procedures in a program. Instead of testing whole software program at once, module testing recommend testing the smaller building blocks of the program. It is largely white box oriented. The objective of doing Module testing is not to demonstrate proper functioning of the module but to demonstrate the presence of an error in the module. Module testing allows implementing of parallelism into the testing process by giving the opportunity to test multiple modules simultaneously.

The final integrated system too has been tested for various test cases such as duplicate entries and type mismatch.

5.3 LIMITATIONS

- ☐ Does not track markets at the live end and database is not fully up to date.
- ☐ User's session timing is not recorded.
- ☐ Better secure interfaces needed for communication with the event planners.

SNAPSHOTS

This chapter consists of working screenshots of the project.

6.1 LOGIN PAGE

This is the login page for existing users.

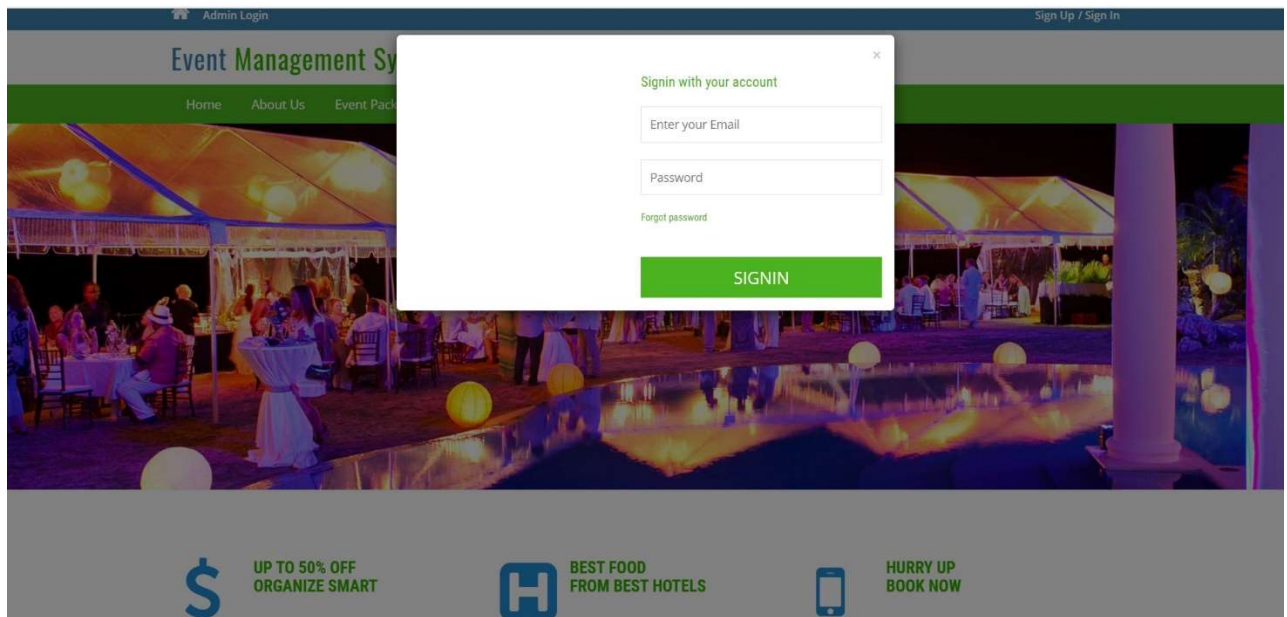


Fig 6.1: Login page

6.2 REGISTRATION PAGE

This is the registration page for any new users.

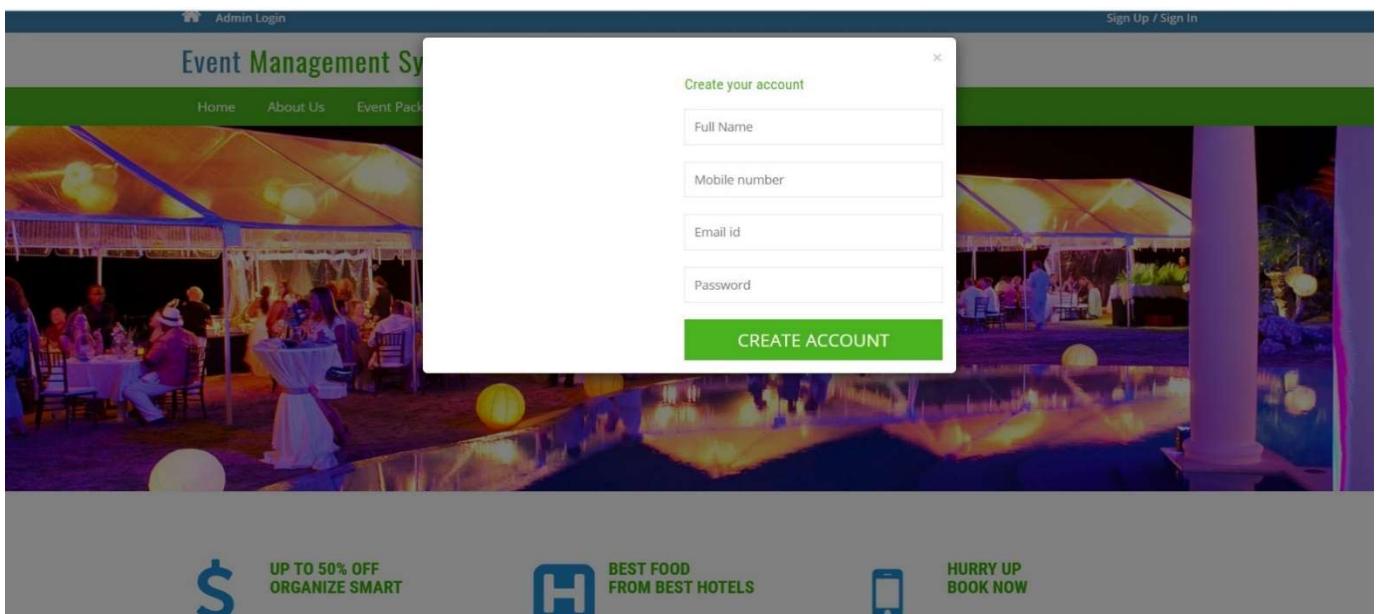


Fig 6.2: Registration page

6.3 HOME PAGE

First home page shown to customers.

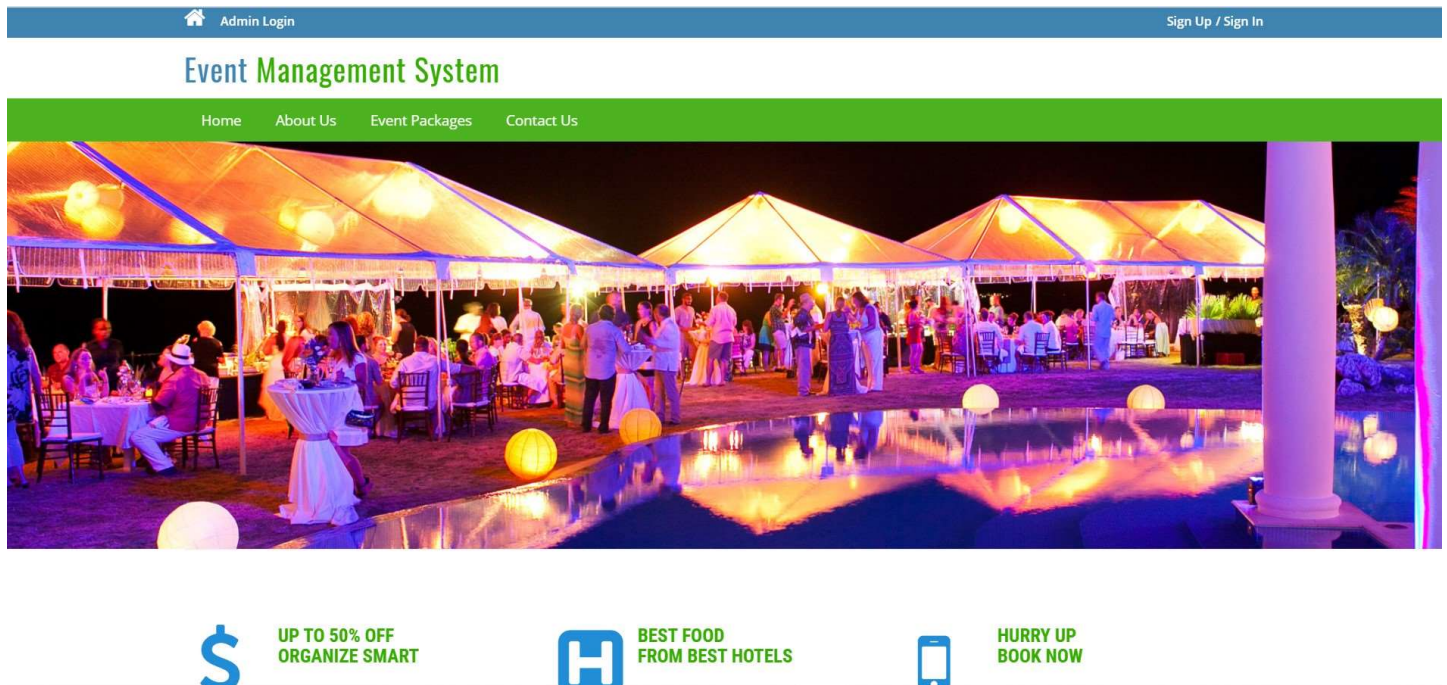


Fig 6.3: Home Page

6.4 EVENT PACKAGE LISTS

List of event packages used in our project.

Package List





	Package Name: WEDDING Package Type : Indoor / Outdoor Package Location : India Features PHOTOGRAPHY:(Drone, Normal) , FOOD:(Veg/Non-Veg) , DECORATIONS:(Flower, Lighting) , ENTERTAINMENT:(DJ, Light, Music)	INR 500000 Details
	Package Name: BIRTHDAY Package Type : Indoor / Outdoor Package Location : India Features PHOTOGRAPHY , FOOD:(Veg/Non-Veg) , DECORATION: Based on themes , ENTERTAINMENT:(Kids Games, Magic Show, Puppet Show) , RETURN GIFT	INR 100000 Details
	Package Name: COLLEGE EVENT Package Type : Indoor / Outdoor Package Location : India Features FOOD:VEG, DECORATIONS:(STAGE SETUP,LIGHTINGS) ENTERTAINMENT:DJ PHOTOGRAPHY (drone, normal)	INR 200000 Details

Fig 6.4: List of Companies

6.5 USER EVENT BOOKING

This page shows the allowing of the user booking event.



WEDDING

#PKG-1

Package Type : Indoor / Outdoor

Package Location : India

Features PHOTOGRAPHY:(Drone, Normal) , FOOD:(Veg/Non-Veg) , DECORATIONS:(Flower, Lighting) , ENTERTAINMENT:(DJ, Light, Music)

from

To

dd-mm-yyyy

dd-mm-yyyy

EVENT DETAILS

Kindly mail the following details: 1. Wedding location---> 2. Approximate number of guests---> 3. Your preferred contact information---> 4: preferred timings---> 5. Few photos of couple(optional)---> 6. The desired wedding theme.

Comment

Book

Fig 6.5: Event booking page

6.6 EVENT HISTORY OF THE USER.

This allows users to see the history of the event registered by them.

[My Profile](#)
[Change Password](#)
[My Event History](#)

Welcome : abc@gmail.com / Logout

Event Management System

[Home](#)
[About Us](#)
[Event Packages](#)



My Event History

#	Booking Id	Package Name	From	To	Comment	Status	Booking Date	Action
1	#BK1	WEDDING	2020-12-17	2020-12-31	aaa	Pending	2020-12-31 19:52:50	Cancel
2	#BK4	WEDDING	2020-12-21	2020-12-25		Confirmed	2020-12-31 19:53:20	Cancel
3	#BK2	BIRTHDAY	2020-12-19		make this as a big event	Pending	2020-12-31 19:52:59	Cancel
4	#BK3	COLLEGE EVENT	2020-12-22	2020-12-26		Pending	2020-12-31 19:53:09	Cancel

Fig 6.6: Event history of the user.