EVENT MANAGEMENT SYSTEM

Project submitted to the

SRM University – AP, Andhra Pradesh

Bachelor of Technology

In

Computer Science and Engineering

School of Engineering and Sciences

Submitted by

KANDULA LOHITH RANGANADHA REDDY

(AP20110010161)

Under the Guidance of

Inturi Anitha Rani

SRM University-AP

Neerukonda, Mangalagiri, Guntur

Andhra Pradesh - 522 240

[December, 2022]

EVENT MANAGEMENT SYSTEM

Project submitted for the

Course

Database Management System (CSE 304)

Certificate

Date: 19-Dec-22

This is to certify that the work present in this Project entitled "**EVENT MANAGEMENT SYSTEM**" has been carried out by **Kandula Lohith Rangandha Reddy** under my/our supervision. The work is genuine, original, and suitable for submission to the SRM University – AP.

Supervisor

(Signature)

Dr. Inturi Anitha Rani

Assistant Professor,

Department of CSE,

SRM University – AP,

Andhra Pradesh.

Acknowledgements

We would like to thank Dr. Inturi Anitha Rani for their support and guidance in completing our project on the topic **EVENT MANAGEMENT SYSTEM**. It was a great learning experience.

I would like to take this opportunity to express my gratitude to all of my group members. The project would not have been successful without their cooperation and inputs.

Kandula Lohith Rangandha Reddy

Table of Contents

Abstract

This project is designed to automate the process of booking a venue for the event, managing the whole process online. It will help in managing events on both small and large scales like conferences, weddings, festivals, formal parties, gatherings, get-togethers, etc.

Event management is a process of organizing a professional and focused event, for aparticular target audience. It involves visualizing concepts, planning, budgeting, organizing andexecuting events such as fashion shows, musical concerts, corporate seminars, exhibitions,wedding celebrations, theme parties, product launching etc.

Management of events & functions is a hectic process, right from business conferences and parties to family functions. Especially weddings are significant events for people and couples are often willing to spend a considerable amount of money to ensure that their weddings are well-organized.

List of Figures

Introduction

These days the world has become a digital world where everything is available in a single click or touch.

The definition of our problem lies in manual system and a fully automated system of Event Management System.

Manual system:

In Manual system is more prone to errors and sometimes it encounters various problems which are unstructured. Because things are managing by the human on the paper there might high chances to get mistakes as well as its time consuming and high money consuming.

Technical system:

The technical system comes with the advent of latest technology and user can access the application over the browser and Check the car and manage the things in just few clicks.

Used tools and technologies

We have a wide range of options of programming languages. From these options we can choose appropriate platform tools, technologies and languages for development of the airline reservation project.

Some of these are as following

Programming Languages: Java.

Relational Database: MYSQL.

SOFTWARE REQUIREMENTS:

Operating system	:	Windows Family
Front End	:	CSS, JSP,HTML
Back End	:	Servlet, JDBC.
Server	:	Tomcat Server

HARDWARE SPECIFICATIONS

Processor	:	(i3) Intel Pentium or more
Ram	:	4 GB
Hard disk	:	16 GB hard disk recommended

Methodology

To develop a system to Event Management, this will perform all the Event Management operation on an online platform.

To develop a system that has good management of data along with integrity and minimizing redundancy.

To develop a system that will be user friendly in all possible ways.

To provide better customer support for User.

Discussion

SOFTWARE ENGINEERING APPROACH

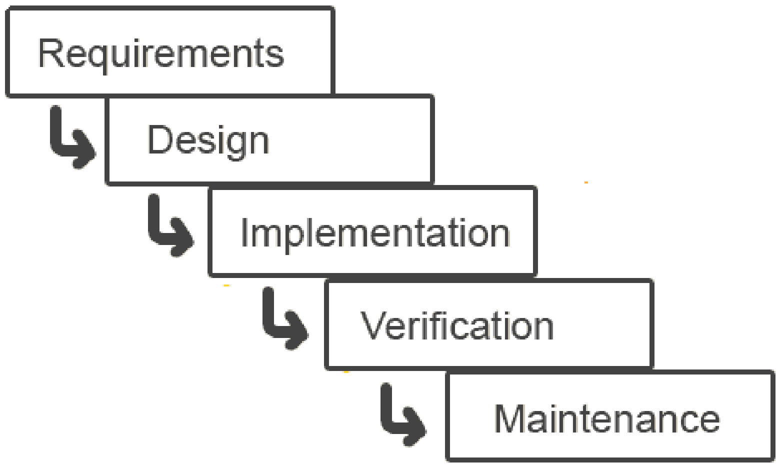


Figure 1

The field of software engineering is related to the development software in systematic manner unlike simple programs which can be developed in isolation and there may not be any systematic approach being followed. there is a big difference between programming and software engineering process. As it provides models that lead to the production of well documented software in a manner that is predictable.

For a planning process, it should be possible to determine in advance the time and effort will be required to develop the final product.

Themodel Ihave used is **Waterfall Model or Classic Life Cycle**. In thismodel first of all the existed system is observed. Then customer requirements are taken in consideration then planning, modeling, construction and finally deployment.

Analysis & system design

Requirement Analysis collects data through interviews, questionnaires, on-site observations, and procedural manuals and like. It is required to organize and convert the data through system flowcharts, data flow diagrams, structured decision tables and the like that support future development of the system.

The Data flow diagrams and various processing logic techniques show how, where, and when data are used or changed in an information system, but these techniques do not show the definition, structure and relationships within the data.

It is a way to focus on functions rather than the physical implementation. This is analogous to the architect’s blueprint as a starting point for system design. The design is a solution, a “how to” approach, compared to analysis, a “what is” orientation.

System design is a highly creative process. This system design process is also referred as data modeling. The most common formatted used the E-R notation explains the characteristics and structure of data independent of how the data may be stored in computer memories.

The process of system design can be divided into three stages. They are two

Structure design

- Database design
- Interface design

- As we know that system design is a solution to “How to approach to the creation of new system”. It provides the understudying and procedural details necessary for implementing the system. The steps involved during system design were as follow

DFD for Event Management System

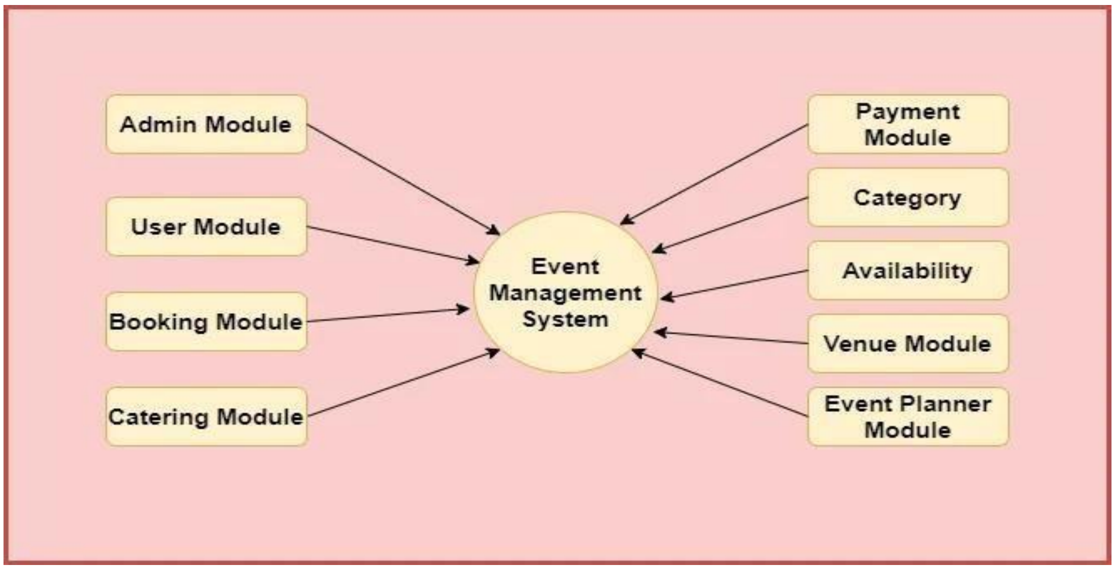


Figure 2

First Level of Data Flow Diagram for Admin

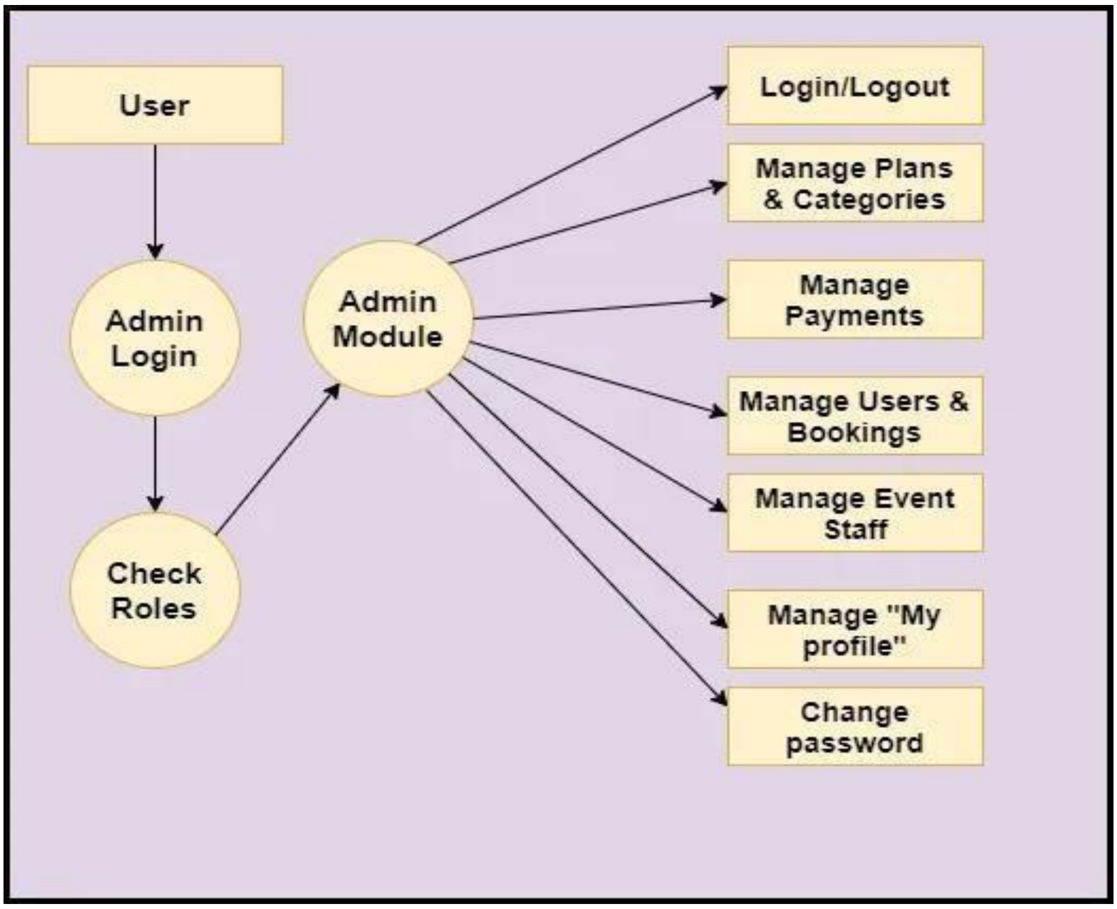


Figure 3

Second Level of Data Flow Diagram for User

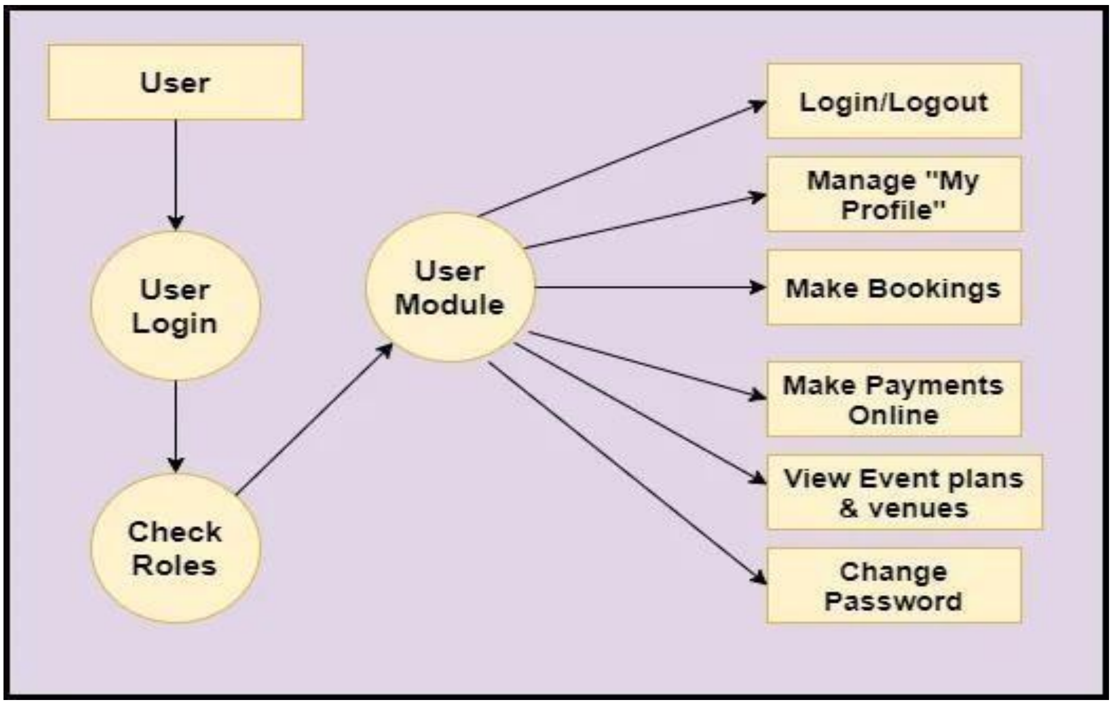


Figure 4

UML Diagrams

DFD diagrams:

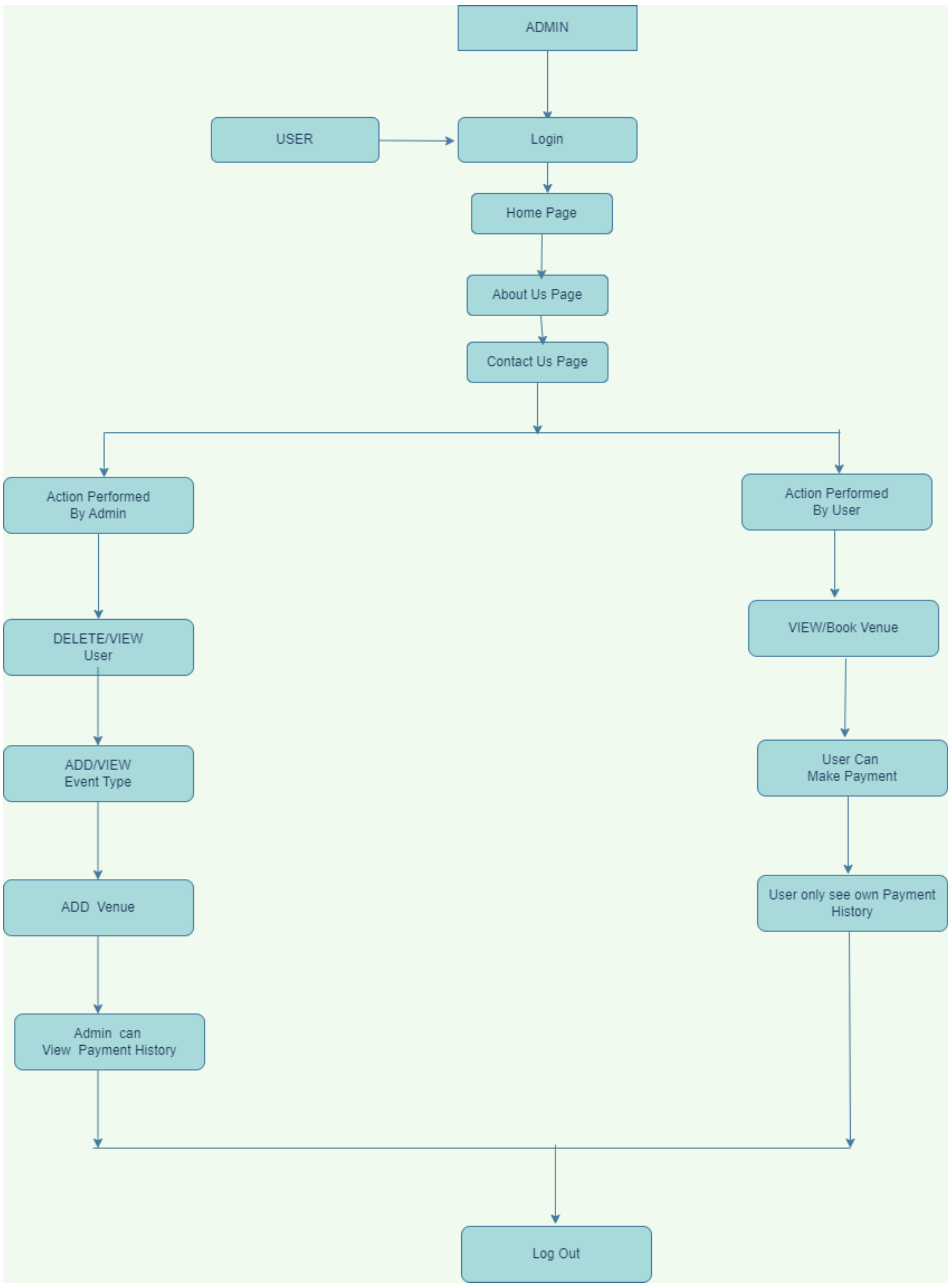


Figure 5

E-R Diagram

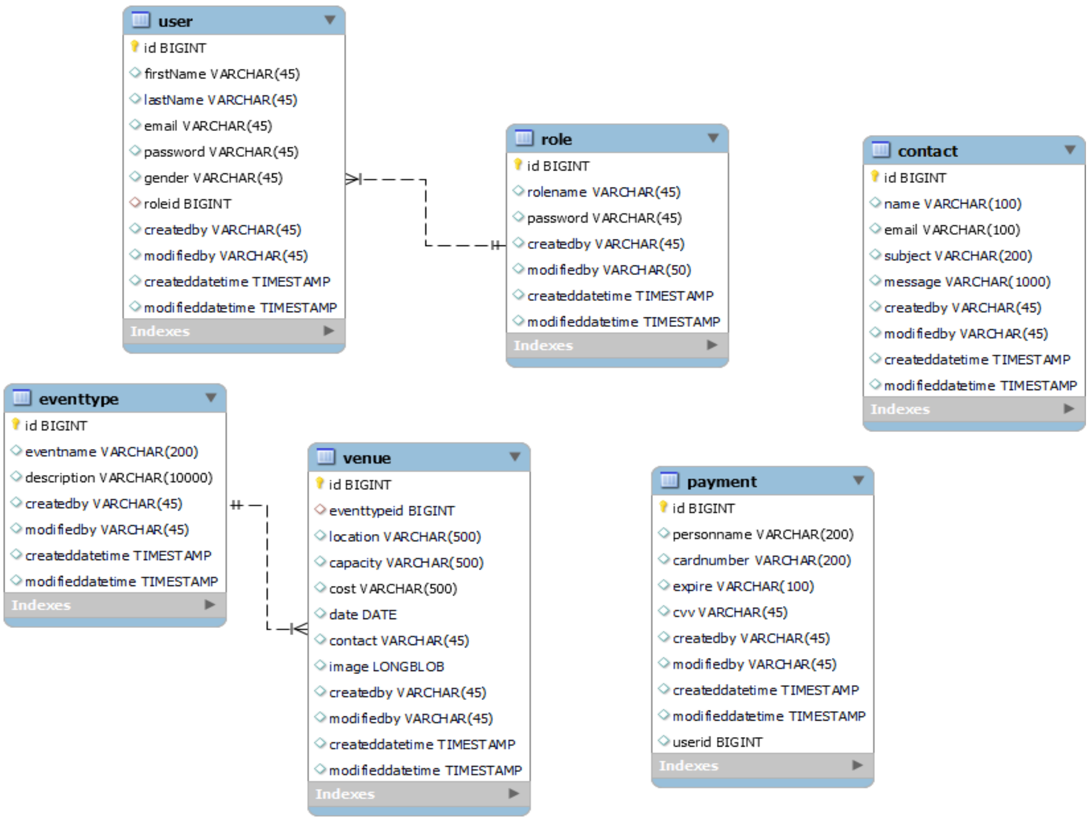


Figure 6

Logical and physical design

In the current physical system is thoroughly reviewed from point of view like how the data flow, what are file contents, its volumes, complexity and frequency etc.

After this input, output specifications security and control specification is prepare. It was also decided that how physical data information will flow through the system and a physical design walkthrough.

OUTPUT DESIGN

- The format of outputs is designed in user-friendly way that it is simple to read and interpret in the present output we have clearly labeled title, messages it contains date and time and all the fields are clearly mentioned as placeholders or label.

Screen Shorts

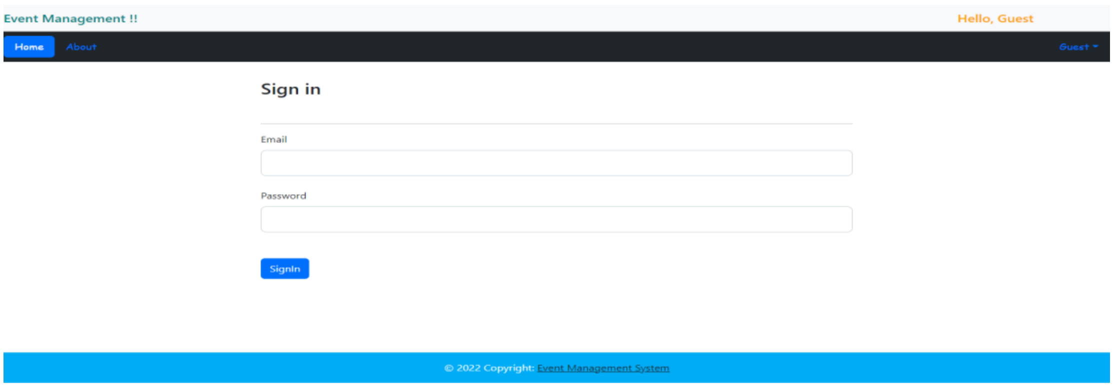
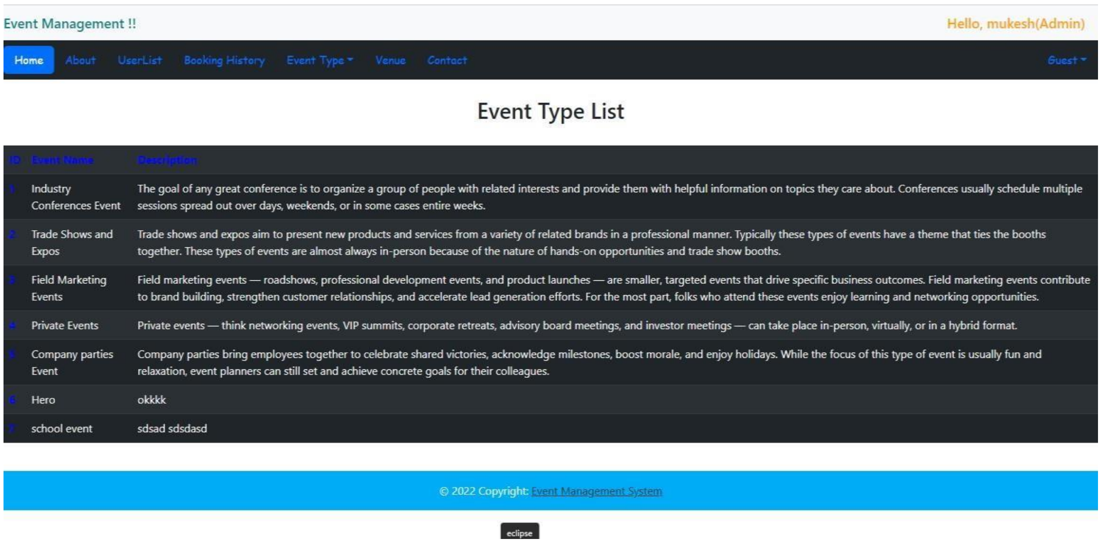


Figure 7



Event Type List

ID	Event Name	Description
1	Industry Conferences Event	The goal of any great conference is to organize a group of people with related interests and provide them with helpful information on topics they care about. Conferences usually schedule multiple sessions spread out over days, weekends, or in some cases entire weeks.
2	Trade Shows and Expos	Trade shows and expos aim to present new products and services from a variety of related brands in a professional manner. Typically these types of events have a theme that ties the booths together. These types of events are almost always in-person because of the nature of hands-on opportunities and trade show booths.
3	Field Marketing Events	Field marketing events — roadshows, professional development events, and product launches — are smaller, targeted events that drive specific business outcomes. Field marketing events contribute to brand building, strengthen customer relationships, and accelerate lead generation efforts. For the most part, folks who attend these events enjoy learning and networking opportunities.
4	Private Events	Private events — think networking events, VIP summits, corporate retreats, advisory board meetings, and investor meetings — can take place in-person, virtually, or in a hybrid format.
5	Company parties Event	Company parties bring employees together to celebrate shared victories, acknowledge milestones, boost morale, and enjoy holidays. While the focus of this type of event is usually fun and relaxation, event planners can still set and achieve concrete goals for their colleagues.
6	Hero	okkkk
7	school event	sdsad sdsdasd

Figure 8

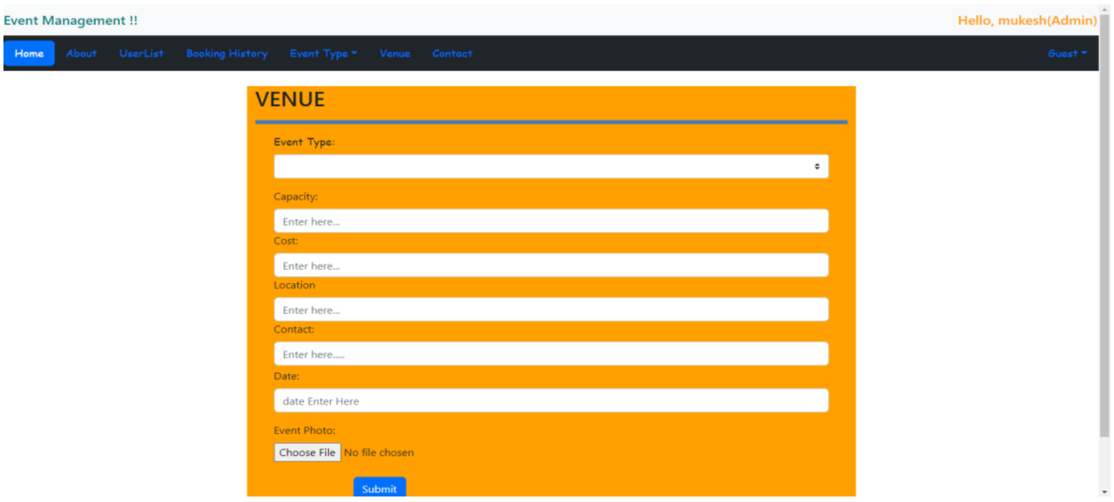
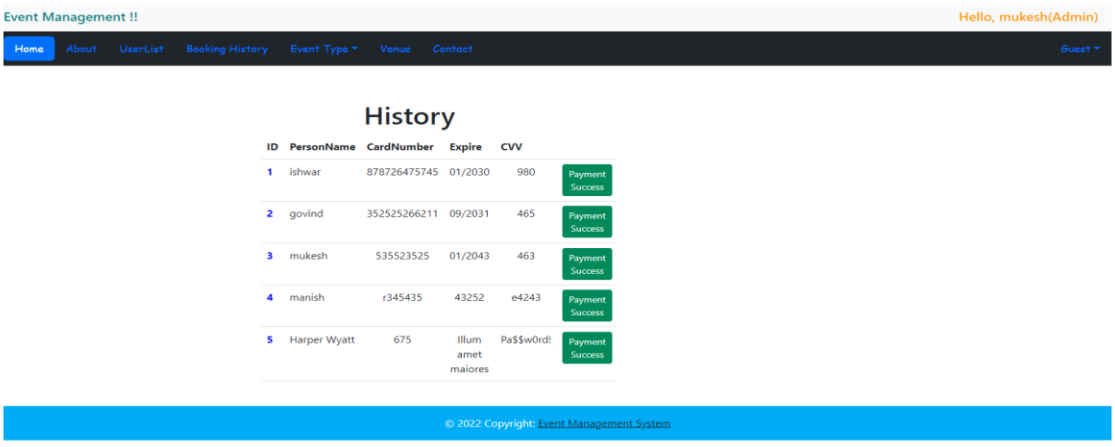


Figure 9

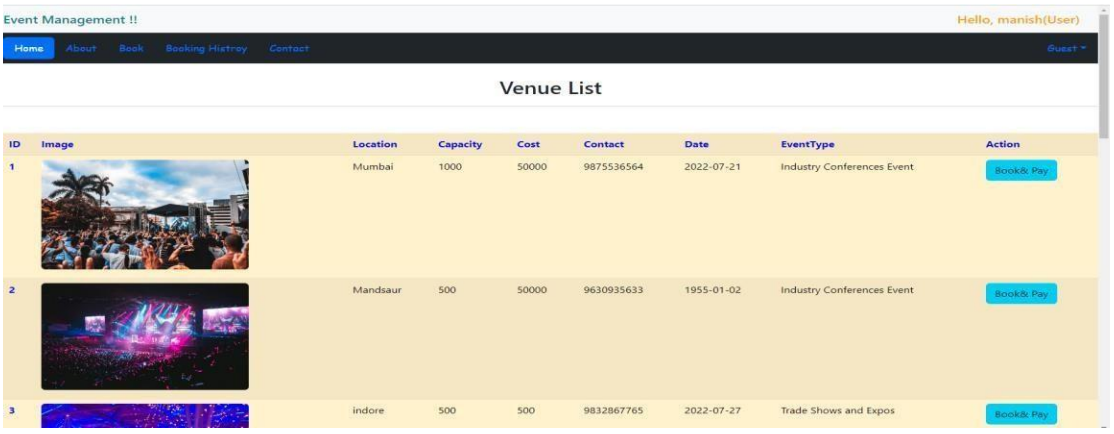


History

ID	PersonName	CardNumber	Expire	CVV	
1	ishwar	878726475745	01/2030	980	Payment Success
2	govind	352525266211	09/2031	465	Payment Success
3	mukesh	535523525	01/2043	463	Payment Success
4	manish	r345435	43252	e4243	Payment Success
5	Harper Wyatt	675	Illum amet maiores	Pa$$w0rd!	Payment Success

© 2022 Copyright: Event Management System

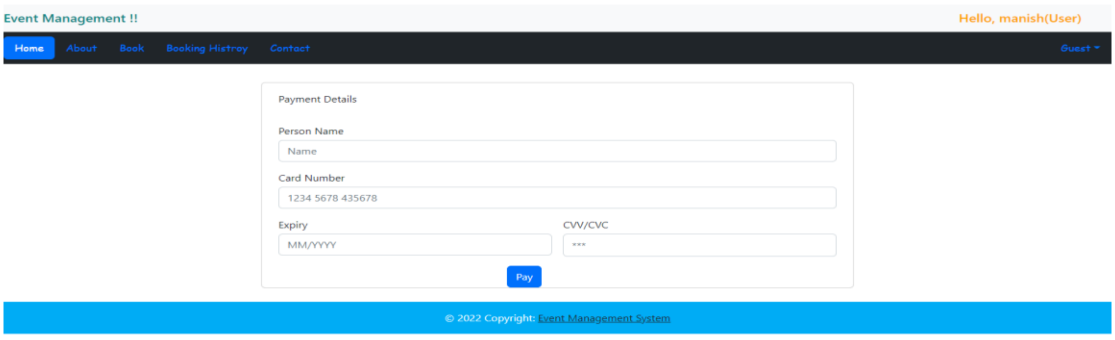
Figure 10



Venue List

ID	Image	Location	Capacity	Cost	Contact	Date	EventType	Action
1		Mumbai	1000	50000	9875536564	2022-07-21	Industry Conferences Event	Book& Pay
2		Mandsaur	500	50000	9630935633	1955-01-02	Industry Conferences Event	Book& Pay
3		indore	500	500	9832867765	2022-07-27	Trade Shows and Expos	Book& Pay

Figure 11



Payment Details

Person Name
Name

Card Number
1234 5678 435678

Expiry
MM/YYYY

CVV/CVC
***

Pay

© 2022 Copyright: Event Management System

Figure 12

Source Code

```
package in.co.online.Bean;


import java.sql.Blob;

import java.util.Date;


public class VenueBean extends BaseBean {


	private long eventtypeid;

	private String location;

	private String capacity;

	private String cost;

	private  Date date;

	private String contact;

	private Blob image;

	private String eventtype;


	public String getEventtype() {

		return eventtype;

	}

	public void setEventtype(String eventtype) {

		this.eventtype = eventtype;

	}

	public long getEventtypeid() {

		return eventtypeid;

	}

	public void setEventtypeid(long eventtypeid) {

		this.eventtypeid = eventtypeid;

	}
```

```
public String getLocation() {
	return location;
}
public void setLocation(String location) {
	this.location = location;
}
public String getCapacity() {
	return capacity;
}
public void setCapacity(String capacity) {
	this.capacity = capacity;
}
public String getCost() {
	return cost;
}
public void setCost(String cost) {
	this.cost = cost;
}
public Date getDate() {
	return date;
}
public void setDate(Date date) {
	this.date = date;
}
public String getContact() {
	return contact;
}
public void setContact(String contact) {
	this.contact = contact;
	}
	public Blob getImage() {
		return image;
	}
	public void setImage(Blob image) {
		this.image = image;
	}



}


package in.co.online.Controller;


importjava.io.IOException;
import java.io.InputStream;
import java.sql.Blob;
import java.util.List;


import javax.servlet.ServletException;
import javax.servlet.annotation.MultipartConfig;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.Part;
import javax.sql.rowset.serial.SerialBlob;


import in.co.online.Bean.BaseBean;
import in.co.online.Bean.EventTypeBean;
import in.co.online.Bean.UserBean;
import in.co.online.Bean.VenueBean;
import in.co.online.Model.EventTypeModel;
import in.co.online.Model.UserModel;
import in.co.online.Model.VenueModel;
import in.co.online.Utility.DataUtility;
import in.co.online.Utility.DataValidater;
import in.co.online.Utility.PropertyReader;
import in.co.online.Utility.ServletUtility;

/**
 * Servlet implementation class VenueCtl
 */
@WebServlet(name = "VenueCtl", urlPatterns = "/venue")
@MultipartConfig(maxFileSize = 16177215)
public class VenueCtl extends BaseCtl {
	private static final long serialVersionUID = 1L;
	publicstaticfinal String OP_SUBMIT = "Submit";



	protected boolean validate(HttpServletRequest request) {
		boolean pass = true;
		if (DataValidater.isNull(request.getParameter("eventtypeid"))) {
			request.setAttribute("eventtypeid", 
PropertyReader.getvalue("error.require", "EventTypeid"));
			pass = false;
		}
		if (DataValidater.isNull(request.getParameter("location"))) {
			request.setAttribute("location", 
PropertyReader.getvalue("error.require", "Location"));
```

```
                pass = false;
            }
            if (DataValidater.isNull(request.getParameter("capacity"))) {
                request.setAttribute("capacity",
PropertyReader.getvalue("error.require",  "Capacity"));
                pass = false;
            }
            if (DataValidater.isNull(request.getParameter("cost"))) {
                request.setAttribute("cost",
PropertyReader.getvalue("error.require", "Cost"));
                pass = false;
            }
            if (DataValidater.isNull(request.getParameter("date"))) {
                request.setAttribute("date",
PropertyReader.getvalue("error.require", "Date"));
                pass = false;
            }
            if (DataValidater.isNull(request.getParameter("contact"))) {
                request.setAttribute("contact",
PropertyReader.getvalue("error.require", "Contact"));
                pass = false;
            }
            return pass;
        }

        /**
         * @see HttpServlet#HttpServlet()
         */
        public VenueCtl() {
            super();
```

```
		// TODO Auto-generated constructor stub
	}


	/**
	 * @see HttpServlet#doGet(HttpServletRequest request,
HttpServletResponse
	 *      response)
	 */
	protected BaseBean populateBean(HttpServletRequest request){
		VenueBean bean = new VenueBean();
		bean.setId(DataUtility.getlong(request.getParameter("id")));

	bean.setEventtypeid(DataUtility.getlong(request.getParameter("eventt
ypeid")));

	bean.setLocation(DataUtility.getString(request.getParameter("location"
)));

	bean.setCapacity(DataUtility.getString(request.getParameter("capacity
")));

	bean.setCost(DataUtility.getString(request.getParameter("cost")));

	bean.setDate(DataUtility.getDate(request.getParameter("date")));

	bean.setContact(DataUtility.getString(request.getParameter("contact"))
);
		System.out.println("cost:"+bean.getCost());
		System.out.println("cost1:"+bean.getDate());
		System.out.println("cost2:"+bean.getCapacity());
		System.out.println("cost3:"+bean.getContact());
		System.out.println("cost4:"+bean.getEventtypeid());
		Blob blob = null;
		Part filepart;
		try {
			filepart = request.getPart("image");
			blob = medicinePacketUpload(filepart);
		} catch (Exception e) {

		}
		bean.setImage(blob);
		System.out.println("cost6:"+bean.getImage());
		populateDto(bean, request);
		return bean;
	}

	public Blob medicinePacketUpload(Part part) throws IOException {
		System.out.println("this si part :" + part);
		InputStream inputStream = null;
		Blob blob = null;
		inputStream = part.getInputStream();
		byte[] b = new byte[inputStream.available()];
		inputStream.read(b);
		try {
			blob = new SerialBlob(b);

		} catch (Exception e) {
		}

		return blob;
	}
```

```
        protected void doGet(HttpServletRequest request,
HttpServletResponse response)
                        throws ServletException, IOException {

                ServletUtility.forward(getView(), request, response);
        }

        /**
         * @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse
         *      response)
         */
        protected void doPost(HttpServletRequest request,
HttpServletResponse response)
                        throws ServletException, IOException {
                System.out.println("in do post");
                long id = DataUtility.getlong(request.getParameter("id"));
                String op =
DataUtility.getString(request.getParameter("operation"));
                VenueModel model = new VenueModel();
                 VenueBean bean = new VenueBean();
                if (OP_SUBMIT.equalsIgnoreCase(op)) {
                bean =(VenueBean) populateBean(request);
                try {
                        long pk = model.add(bean);
                        ServletUtility.setbean(bean, request);
                        ServletUtility.setSuccessMessage("Venue ADD
Successfully", request);
                        ServletUtility.forward(getView(), request,
response);
                        return;
                        } catch (Exception e) {
                                e.printStackTrace();
                        }

                }
                System.out.println("forword");
        ServletUtility.forward(getView(), request, response);
        }

        @Override
        protected String getView(){
                return EM_View.VENUE_VIEW;
        }

}



package in.co.online.Model;

import java.sql.Connection;
import java.sql.Date;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

import in.co.online.Bean.VenueBean;
import in.co.online.Exception.ApplicationException;
import in.co.online.Utility.JDBCDataSource;

public class VenueModel {

	public Integer nextpk() {
		Connection conn = null;
		int pk = 0;
		try {
			conn = JDBCDataSource.getconnection();
			PreparedStatement ps = conn.prepareStatement("SELECT
MAX(ID) FROM venue");
			ResultSet rs = ps.executeQuery();
			while (rs.next()) {
				pk = rs.getInt(1);
			}

		} catch (Exception e) {

		}
		return pk + 1;
	}

	public long add(VenueBean bean) throws Exception {
		System.out.println("in add method");
		Connection conn = null;
		int pk = 0;
		try {
			conn = JDBCDataSource.getconnection();
			pk = nextpk();
```

```
                        conn.setAutoCommit(false);

                        PreparedStatement ps = conn.prepareStatement("INSERT
INTO venue VALUES(?,?,?,?,?,?,?,?,?,?,?,?)");

                        ps.setLong(1, pk);

                        ps.setLong(2, bean.getEventtypeid());

                        ps.setString(3, bean.getLocation());

                        ps.setString(4, bean.getCapacity());

                        ps.setString(5, bean.getCost());

                        ps.setDate(6, new Date(bean.getDate().getTime()));

                        ps.setString(7, bean.getContact());

                        ps.setBlob(8, bean.getImage());

                        ps.setString(9, bean.getCreatedby());

                        ps.setString(10, bean.getModifiedby());

                        ps.setTimestamp(11, bean.getCreateddatetime());

                        ps.setTimestamp(12, bean.getModifieddatetime());

                        ps.executeUpdate();

                        conn.commit();

                        ps.close();


                } catch (Exception e) {

                        throw new ApplicationException("Exception : add
rollback exception " + e.getMessage());

                }finally {

                        JDBCDataSource.closeconnection(conn);

                }

                return pk;

        }


        public List list() throws Exception {

                ArrayList list = new ArrayList();
                try {
                        Connection conn = null;
                        conn = JDBCDataSource.getconnection();
                        PreparedStatement ps =
        conn.prepareStatement("SELECT
venue.id,eventtype.eventname,capacity,cost,image,date,contact,location
FROM venue INNER JOIN eventtype ON venue.eventtypeid=eventtype.id");
                        ResultSet rs = ps.executeQuery();
              while (rs.next()) {
                                VenueBean bean = new VenueBean();
                                bean.setId(rs.getLong(1));
                                bean.setEventtype(rs.getString(2));
                                bean.setCapacity(rs.getString(3));
                                bean.setCost(rs.getString(4));
                                bean.setImage(rs.getBlob(5));
                                bean.setDate(rs.getDate(6));
                                bean.setContact(rs.getString(7));
                                bean.setLocation(rs.getString(8));
                                list.add(bean);
              }

                } catch (Exception e) {
                        e.printStackTrace();
                }
                return list;
}
}
```

```
<%@page import="in.co.online.Bean.VenueBean"%>

<%@page import="in.co.online.Controller.VenueCtl"%>

<%@page import="java.util.List"%>

<%@page import="java.sql.ResultSet"%>

<%@page import="java.sql.PreparedStatement"%>

<%@page import="in.co.online.Utility.JDBCDataSource"%>

<%@page import="java.sql.Connection"%>

<%@page import="in.co.online.Utility.ServletUtility"%>

<%@page import="in.co.online.Utility.DataUtility"%>

<%@page import="java.util.Iterator"%>


<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

        pageEncoding="ISO-8859-1"%>

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet"

        href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css
">

<link rel="stylesheet"

        href="https://cdnjs.cloudflare.com/ajax/libs/bootstra
p- datepicker/1.9.0/css/bootstrap-datepicker.min.css">

<link rel="stylesheet"

        href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">

<script

        src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></scri
pt>

<script

        src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/js/bootstrap.bundle.mi
n.js"></script>
```

```
<script
        src="https://cdnjs.cloudflare.com/ajax/libs/bootstra
p- datepicker/1.9.0/js/bootstrap-
datepicker.min.js"></script>

<script type="text/javascript">

        $('.datepicker').datepicker();

</script>

<meta charset="ISO-8859-1">

<title>Venue</title>

</head>

<body>

        <%@include file="Header.jsp"%>

        <br>

        <div class="container">

                <div class="row">

                        <div class="col-2"></div>

                        <div class="col-8" style="background-color: orange;">

                                <form action="<%=EM_View.VENUE_CTL%>"
method="post"

                                        enctype="multipart/form-data">




                                        <h2>VENUE</h2>

                                        <hr class="border border-primary border-3
opacity-75">

                                        <h6 style="color:
red;"><%=ServletUtility.getErrorMessage(request)%></h6>

                                        <h6 style="color:
green;"><%=ServletUtility.getSuccessMessage(request)%></h6>

                                        <jsp:useBean id="bean" scope="request"

                                                class="in.co.online.Bean.VenueBean"/>
                                        <inputtype="hidden"name="id"
value="<%=bean.getId()%>"> <input

                                             type="hidden" name="createdby"
value="<%=bean.getCreatedby()%>">

                                        <input type="hidden" name="modifiedby"


     value="<%=bean.getModifiedby()%>"> <input type="hidden"

                                             name="createdDatetime"
value="<%=bean.getCreateddatetime()%>">

                                        <input type="hidden"

name="modifiedDateTime"


     value="<%=bean.getModifieddatetime()%>">



                                        <div class="container">

                                             <div class="col-md-12">

                                                  <label
for="exampleInputPassword1" style="font-family: cursive;">Event

                                                       Type:</label>

                                             <div class="form-group">

                                                  <select class="custom-select"

name=eventtypeid>

                                                       <%

                                                            Connection con
= JDBCDataSource.getconnection();

                                                            String sql =
"SELECT * FROM eventtype";


     PreparedStatement ps = con.prepareStatement(sql);

                                                            ResultSet rs =
ps.executeQuery();
```

```
                                                          while (rs.next())
{
                                                      %>


                                                      <option value="--------
Select------ "></option>

                                                      <option
value="<%=rs.getLong(1)%>"><%=rs.getString(2)%></option>


                                                      <%
                                                           }
                                                      %>
                                                 </select>
                                                 <div class="form-text"
style="color: red"><%=ServletUtility.getErrorMessage("eventtypeid",
request)%></div>
                                            </div>
                                       </div>


                                       <div class="col-12">
                                            <label for="inputAddress"
class="form-label">Capacity:</label> <input
                                                 type="text" class="form-
control"
                                                 name="capacity"
placeholder="Enter here..."

     value="<%=DataUtility.getStringData(bean.getCapacity())%>">
                                       </div>
                                       <font
color="red"><%=ServletUtility.getErrorMessage("capacity",
request)%></font>
```

```
                                        <div class="col-12">
                                             <label for="inputAddress"
class="form-label">Cost:</label>  <input
                                                  type="text" class="form-
control"  name="cost"
                                                  placeholder="Enter here..."

     value="<%=DataUtility.getStringData(bean.getCost())%>">
                                        </div>
                                        <font
color="red"><%=ServletUtility.getErrorMessage("cost", request)%></font>



                                        <div class="col-12">
                                             <label for="inputAddress"
class="form-label">Location</label> <input
                                                  type="text" class="form-
control" id="inputAddress"
                                                  name="location"
placeholder="Enter  here..."

     value="<%=DataUtility.getStringData(bean.getLocation())%>">
                                        </div>
                                        <font
color="red"><%=ServletUtility.getErrorMessage("location",
request)%></font>


<div class="col-md-12">
                                             <label
for="form_message">Contact:</label> <input
                                                  class="form-control"
type="text" name="contact"
                                                  placeholder="Enter  here......."
```

```
          value="<%=DataUtility.getStringData(bean.getContact())%>">
                                        </div>
                                        <div class="form-text" style="color:
red"><%=ServletUtility.getErrorMessage("contact", request)%></div>

<div class="col-md-12">

                                        <label for="form_message">Date:</label>

                                        <div class="form-group">
                                                <input type="text" class="form-
control" id="exampleInputEmail1"
                                                          aria-describedby="emailHelp"
name="date" id="datepicker"
                                                          data-provide="datepicker"

          value="<%=DataUtility.getStringData(bean.getDate())%>"
                                                          placeholder="date Enter
Here"> <font color="red"><%=ServletUtility.getErrorMessage("date",
request)%></font>
                                        </div>
</div>



                              <div class="col-md-12">
                                                <label
for="exampleFormControlInput1" class="form-label">Event
                                                          Photo:</label> <br><input
type="file" id="exampleFormControlInput1"
                                                          name="image">
                                                </div>


                                                <br>
                                                 <input type="submit" class="btn btn-
primary"
                                                        name="operation" style="margin-left:
130px;"

      value="<%=VenueCtl.OP_SUBMIT%>">
</div>
                                          </form>
                                    </div>
                              </div>
                        </div>



      <div style="margin-top: 2%;">
            <%@include file="Footer.jsp"%>
      </div>
</body>
                                          </html>
```

Testing
Software Testing

Software testing is a process of *verifying* and *validating* that a software application or program.

Meets the business and technical requirements that guided its design and development, and Works asexpected.

Software testing also identifies important *defects*, flaws, or errors in the application code that must be fixed. The modifier "important" in the previous sentence is, well, important because defects must be categorized by severity.

During test planning we decide what an important defect is by reviewing the requirements and design documents with an eye towards answering the question "Important to whom?" Generally speaking, an important defect is one that from the customer's perspective affects the usability or functionality of the application.

The quality assurance aspect of software development documenting the degree to which the developers followed corporate standard processes or best practices is not addressed in this paper because assuring quality is not a responsibility of the testing team.

The testing team cannot improve quality they can only measure it, although it can be argued that doing things like designing tests before coding begins will improve quality because the coders can then use that information while thinking about their designs and during coding and debugging.

Software testing has three main purposes: verification, validation, and defect finding.

The *verification* process confirms that the software meets its technical specifications. A "specification" is a description of a function in terms of a measurable output value given a specific input value under specific preconditions.

A simple specification may be along the line of "a SQL query retrieving data for a single account against the multi-month account-summary table must return these eight fields <list> ordered by month within 3 seconds of submission."

The *validation* process confirms that the software meets the business requirements. A simple example of a business requirement is "After choosing a branch office name, information about the branch's customer account managers will appear in a new window. The window will present manager identification and summary information about each manager's customer base: <list of data elements>."

Other requirements provide details on how the data will be summarized, formatted and displayed.

A *defect* is a variance between the expected and actual result. The defect's ultimate source may be traced to a fault introduced in the specification, design, or development (coding) phases.

Testing methods

Software testing methods are traditionally divided into black box testing and white box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

Black box testing

Black box testing treats the software as a "black box"—without any knowledge of internal implementation. Black box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, fuzz testing, model-based testing, traceability matrix, exploratory testing and specification-based testing.

Specification-based testing: Specification-based testing aims to test the functionality of software according to the applicable requirements. Thus, the tester inputs data into, and only sees the output from, the test object. This level of testing usually requires thorough test cases to be provided to the tester, who then can simply verify that for a given input, the output value (or behavior), either "is" or "is not" the same as the expected value specified in the test case.

Specification-based testing is necessary, but it is insufficient to guard against certain risks.

Advantages and disadvantages:

The black box tester has no "bonds" with the code, and a tester's perception is very simple: a code *must* have bugs. Using the principle, "Ask and you shall receive," black box testers find bugs where programmers do not.

But, on the other hand, black box testing has been said to be "like a walk in a dark labyrinth without a flashlight," because the tester doesn't know how the software being tested was actually constructed.

As a result, there are situations when 1 a tester writes many test cases to check something that could have been tested by only one test case, and/or 2 some parts of the back-end are not tested at all.

Therefore, black box testing has the advantage of "an unaffiliated opinion," on the one hand, and the disadvantage of "blind exploring," on the other.

White box testing

White box testing is when the tester has access to the internal data structures and algorithms including the code that implement these.

Types of white box testing

API testing (application programming interface) - Testing of the application using Public and Private APIs

Code coverage - creating tests to satisfy some criteria of code coverage (e.g., the test designer can create tests to cause all statements in the program to be executed at least once)

Fault injection methods - improving the coverage of a test by introducing faults to test codepaths

Mutation testing methods

Static testing White box testing includes all static testing

A sample testing life cycle

Although variations exist between organizations, there is a typical cycle for testing:

Requirementsanalysis: Testingshouldbeginintherequirementsphaseof the software development life cycle. During the design phase, testers work with developers in determining what aspects of a design are testable and with what parameters those testswork.

Test planning: Test strategy, test plan, tested creation. Since many activities will be carried out during testing, a plan isneeded.

Test development: Test procedures, test scenarios, test cases, test datasets, test scripts to use in testing software.

Testexecution: Testersexecutethesoftwarebasedontheplansandtestsand report any errors found to the development team.

Test reporting: Once testing is completed, testers generate metrics and make final reports on their test effort and whether or not the software tested is ready for release.

Test result analysis: Or Defect Analysis, is done by the development team usually along with the client, in order to decide what defects should be treated, fixed, rejected (i.e. found software working properly) or deferred to be dealt with later.

Defect Retesting: Once a defect has been dealt with by the development team, it is retested by the testingteam.

Regression testing: It is common to have a small test program built of a subset of tests, for each integration of new, modified, or fixed software, in order to ensure that the latest delivery has not ruined anything, and that the software product as a whole is still working correctly.

Test Closure: Once the test meets the exit criteria, the activities such as capturing the key outputs, lessons learned, results, logs, documents related to the project are archived and used as a reference for future projects.

Concluding Remarks

Event Management System is user friendly and cost effective system, it is customized with activities related to event management life-cycle.

It provides a new edge to management industry. Solution Dot always keep your objectives and goals on top priority while developing any plan of work.

In this project, we made attempt to effectively introduce the concept of event management systems already existing in the society. We then explain the concept of online event management systems which are already present. We describe the proposed system and explain the features implemented by our proposed system .We also give a brief overview of the technologies used during the development of our proposed system. This project can be further refined and extended by introducing new and more innovative features.

Future Work

Event management system project also keeps track of the number of participants in the event and their registration accordingly. It is a remarkable application that can efficiently manage an event, store and process necessary information related to event, and uses this information for future events. It can be further modified to execute for large or small events. Event management system is largely being used by event mangers worldwide.

References

1. https://www.vertabelo.com/blog/how-to-plan-and-run-events-an-event-management-data-model/
2. https://learn.microsoft.com/en-us/sql/ssms/agent/manage-events?view=sql-server-ver16
3. https://github.com/PuneethReddyHC/event-management/blob/master/database/events.sql
4. https://docs.oracle.com/cd/A57673_01/DOC/sysman/doc/A55898_01/ch_ems.htm
5. https://code-projects.org/event-management-system-in-php-css-javascript-and-mysql-free-download/