# VISVESVARAYA TECHNOLOGICAL UNIVERSITY
## "JNANA SANGAMA", BELAGAVI-590018

*A MINI PROJECT REPORT ON*

# EVENT MANAGEMENT SYSTEM

***Submitted in partial fulfillment of the requirement***

***For the award of degree of***

**Bachelor of Engineering**

In

**Computer Science and Engineering**

By


**MAHESH B.V**

**[1KS18CS044]**


**Under the guidance of**


**Mrs.Sougandika Narayan**

Ass.Prof,Dept.of CSE

**Mr.Prashanth.H.S**

Ass.**Prof**,Dept.of CSE




Department of Computer Science & Engineering

K.S.INSTITUTE OF TECHNOLOGY

#14,Raghuvanahalli,Kanakapura Main Road,Bengaluru-560109


2020-21

# K.S.INSTITUTE OF TECHNOLOGY

#14, Raghuvanahalli, Kanakapura Main Road, Bengaluru-560109

**Department of Computer Science & Engineering**

## CERTIFICATE

This is to certify that mini project work entitled "**EVENT DATABASE MANAGEMENT SYSTEM**" carried out by **Mr. MAHESH B.V** bearing USN **1KS18CS044** bonafide student of **K.S. Institute of Technology** in the partial fulfilment for the award of the **Bachelor of Engineering** in **Computer Science & Engineering** of the **Visvesvaraya Technological University, Belagavi,** during the year 2020-21. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The mini project report has been approved as it satisfies the academic requirements in respect of mini Project work prescribed for the said degree for the 5th semester.

**Dr.Rekha B Venkatapur**
Prof & HOD,CS & E Department

**Dr.K V A Balaji**
Prinicipal/CEO,KSIT

**Mrs.Sougandika Narayan**
Asst.Prof,Dept.of CSE

**Mr.Prashanth H S**
Asst.Prof,Dept.of CSE

**Name of the Examiners** **Signature with date**

**1.** _____

**2.** _____

# ACKNOWLEDGEMENT

The successful presentation of the DBMS LABORATORY WITH MINI PROJECT (18CSL58) would be incomplete without the mention of the people who made it possible and whose constant guidance crowned my effort with success.

We would like to extend our gratitude to the MANAGEMENT, KAMMAVARI SANGHAM, Bengaluru, for providing all the facilities to present the DBMS LABORATORY WITH MINI PROJECT.

We would like to extend our gratitude to Dr . K.V.A BALAJI the Principal/CEO, K.S. Institute of Technology, Bengaluru, for facilitating us to present the DBMS LABORATOY WITH MINI PROJECT.

We thank Dr. REKHA B VENKATAPUR, Professor and Head, Department of Computer Science and Engineering, K.S. Institute of Technology, Bengaluru, for her encouragement. We would like to than our Project Guide's, Mrs.SOUGANDIKA NARAYAN & Mr.PRASHANTH.H.S Department of Computer Science and Engineering, K.S. Institute of Technology, Bengaluru, for their constant guidance and inputs.

We would like to thank all the Teaching Staff and Non-Teaching Staff of the college for their co-operation. Finally, we extend my heart-felt gratitude to my family for their encouragement and support without which we would not have come so far. Moreover, we thank all my friends for their invaluable support and cooperation.

**MAHESH B.V**
**(1KS18CS044)**

# ABSTRACT

The main objective of this project is to develop a College Events Management System for the entire college's students, professors and the college staff. This will make the management of events, event venues and participants easier. The college events management system even gives the admin options to view, edit, add and delete events, set the number of participants and view the participants. And to the student the options to view the list of events and participate in them are provided. This database provides details about the event such as the name, description, venue and maximum number of participants allowed.

# TABLE OF CONTENTS

# Chapter 1

# INTRODUCTION

## 1.1 Project Overview

Event Management primarily refers to any assistance offered to the customers which involve planning, organizing, advertising and execution of various events. An event can be of any kind, from a college or university events or private party to a corporate conference. Event management is in its own an artform that cannot clearly be defined. Over the years various event management has boomed. The College Event Management System is developed for the colleges to effortlessly manage annual events in the college. It is an overview of the college events and the events related data contained in the database.

## 1.2 Project Description

With the launch and increase in sales of web-based devices over the last few years, people are using web browsers to get their work done, which makes their lives easier. Websites comprise various different categories such as Events, Entertainment, Sports, Lifestyle, Games, Food and Drink, Health and Fitness, Finance, etc.

Event Management primarily refers to any assistance offered to the customers which involve planning, organizing, advertising and execution of various events. An event can be of any kind, from a college or university events or private party to a corporate conference. Event management is in its own an artform that cannot clearly be defined. Over the years various event management has boomed. The College Event Management System is developed for the colleges to effortlessly manage annual events in the college. It is an overview of the college events and the events related data contained in the database.

The application's interface is designed using custom art elements, the functionality is implemented using HTML, CSS and JS with backend storage MYSQL, and the phase of testing the product was accomplished successfully. The website is not much user intensive but just comprises of having them enter the events, date, category, participants limit and other optional attributes (adding subcategories to the categories).

## 1.3 Scope Of The Project

This database is especially useful for those arguments regarding when an event is going to happen, its venue, total number of participants their respective details. This database provides a clear record of college events and participants. This database will prove useful to college staff, coordinators and students. Coordinators can use this to effectively manage college events, participants, and control the number of entries of participating students.

## 1.4 Aim Of The Project

- The objective of this database management system is to manage the college events efficiently.

- Managing college events and its participants through a single web portal. The college will host the portal on the internet.

- This database management system also manages the production house that the label company hires.

- This database management system helps us maintain a clear record of all the songs in an album.

# Chapter 2

# REQUIREMENT SPECIFICATIONS

## 2.1 Details of Software and Languages

### 2.1.1 Introduction to JavaScript

JavaScript is a very powerful client-side scripting language. JavaScript is used mainly for enhancing the interaction of a user with the webpage. In other words, you can make your webpage more lively and interactively, with the help of JavaScript. JavaScript is also being used widely in game development and mobile application development.

Being a scripting language, JavaScript cannot run on its own. In fact, the browser is responsible for running JavaScript code**.** When a user requests an HTML page with JavaScript in it, the script is sent to the browser and it is up to the browser to execute it. The main advantage of JavaScript is that all modern web browsers support JavaScript. So, you do not have to worry about whether your site visitor uses Internet Explorer, Google Chrome, Firefox or any other browser. JavaScript will be supported. Also, JavaScript runs on any operating system. Thus, JavaScript overcomes the main disadvantages VBscript (Now deprecated) which is limited to just IE and Windows.

### 2.1.2 Introduction to MYSQL

MySQL Workbench simplifies database design and maintenance, automates time-consuming and error-prone tasks, and improves communication among DBA and developer teams. It enables data architects to visualize requirements, communicate with stakeholders, and resolve design issues before a major investment of time and resources is made. It enables model-driven database design, which is the most efficient methodology for creating valid and well-performing databases, while providing the flexibility to respond to evolving business requirements. Model and Schema Validation utilities enforce best practice standards for data modeling, also enforce MySQL-specific physical design standards so no mistakes are made when building new ER diagrams or generating physical MySQL databases. MySQL database connection with JavaScript is simple and easy with Node.js. Using embedded JavaScript we can use queries to manipulate HTML using MYSQL results.

### 2.1.3 Introduction to Node.js

Node.js was written initially by Ryan Dahl in 2009, about thirteen years after the introduction of the first server-side JavaScript environment, Netscape's LiveWire Pro Web. The initial release supported only Linux and Mac OS X. Its development and maintenance was led by Dahl and later sponsored by Joyent.

NodeJS is an open-source, cross-platform JavaScript runtime environment developed on Chrome's V8 JavaScript directly into the native machine code. It is a lightweight framework used to develop server-side web applications. It is mostly used for creating large scale application development, mostly used for video streaming websites, single page application, and other web applications. Node.js uses event- driven, non-blocking Input- Output Model that makes it right for data-intensive real- time applications.

## Features of NodeJS

**Open Source**

Node.js is an open-source platform. It means that the copyright holder has given various rights of studying, editing and distributing the software to anyone for any purpose. **High Scalability**

Since it uses an event mechanism, Node.js is highly scalable and helps the server in a non-blocking response.

**Simple and Fast**

As Node.js is built on Google Chrome's V8 JavaScript engine, its libraries are highly advanced and hence able to run the code at a faster speed.

**No Buffering**

Node.js is blessed with a special feature and that is, it does not buffer any data.

**Single-Threaded**

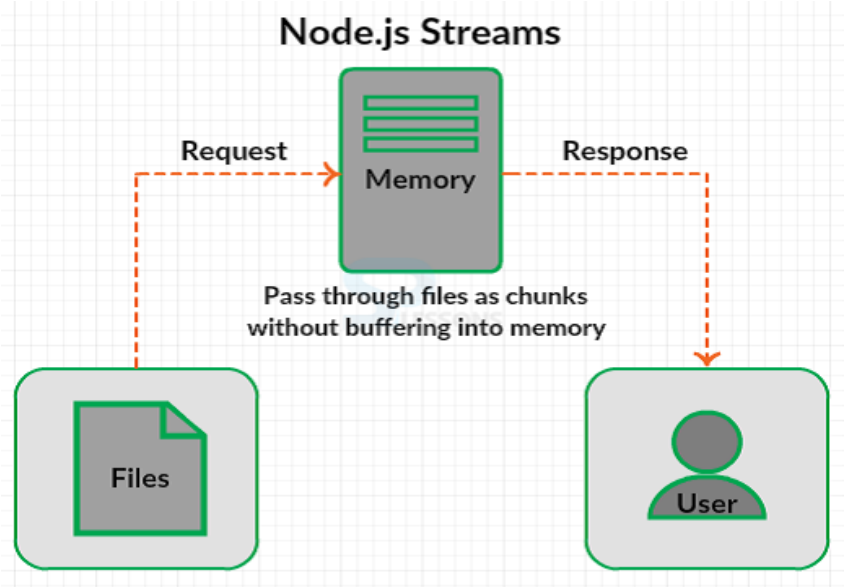
As it uses a process of event looping, Node.js is able to follow the single-threaded model. It helps a single user to handle more than one requests.

**Asynchronous**

Node.js has asynchronous libraries. It is quite helpful as Node.js servers need not wait for an API to send the response and move on to the next API.

**Cross-Platform**

Node.js can be easily created and deployed on several platforms like Windows, Mac, and Linux.




## 2.2 System Requirements

### 2.2.1 Hardware Interface

- A Visual Display unit such as a Monitor
- Intel Pentium or AMD-64 instruction set supporting processor such as AMD A series
- APUs or Intel Core Processor having a clock speed of at least 2,0 GHz
- At least 100 MB of additional free memory or RAM
- At least 80 GB Hard Disk Drive

### 2.2.2 Software Interface

- Client on Internet: Web Browser
- Operating System (any)
- Web Server: Node.js Express Server
- Data Base Server: MySQL
- Front End: HTML,CSS,JAVASCRIPT
- Back End: MYSQL,JAVASCRIPT

# Chapter 3

# SYSTEM DESIGN

## 3.1 Schema Diagram

A database schema is the skeleton that represents the logical view of the entire database. It defines how the data is organized and how the relation among them are associated. It formulates all the constraints that are to be applied on the data. A database schema defines its entities and relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It's the database designer who design the schema to help programmer understand the database and make it useful.

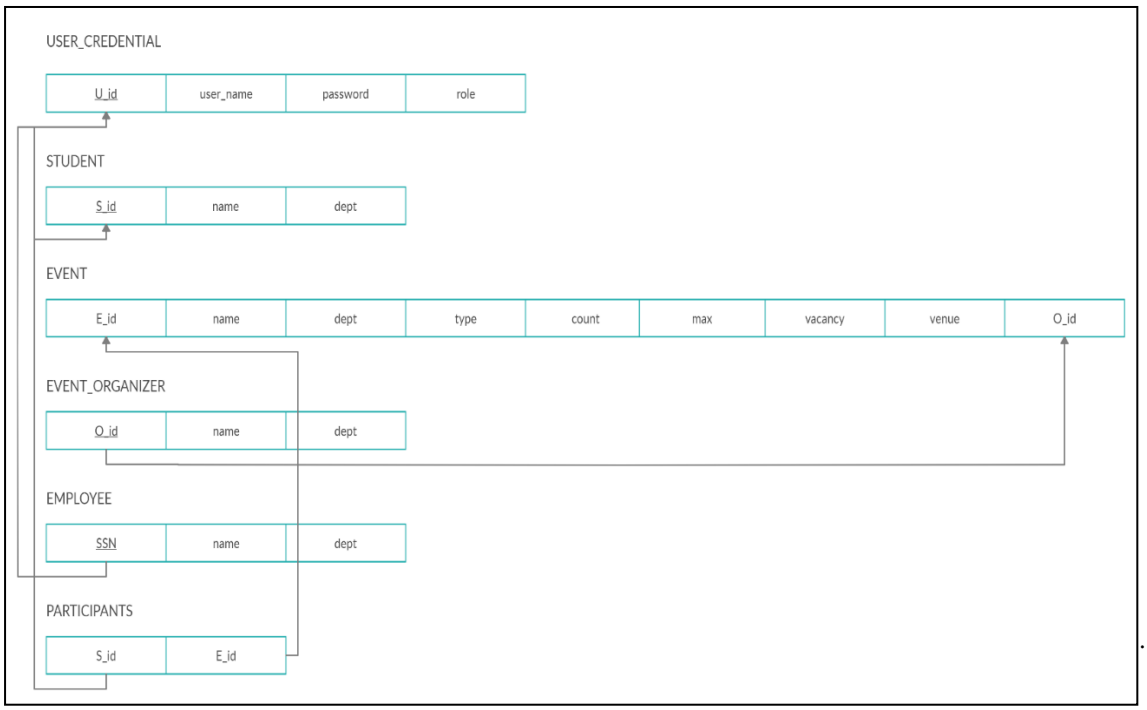



**Fig 3.1: Schema Diagram**

### 3.1.1 EMPLOYEE Table

**Table 3.1.1: EMPLOYEE Table**

| NAME | NULL | TYPE |
|---|---|---|
| EID | NOT NULL | INT(11) |
| EUID | NOT NULL | INT(11) |
| ENAME | NOT NULL | VARCHAR(255) |
| EDEPT | NOT NULL | VARCHAR(255) |

### 3.1.2 EVENTS Table

**Table 3.1.2: EVENTS Table**

| NAME | NULL | TYPE |
|---|---|---|
| EVID | NOT NULL | INT(11) |
| EVNAME | NOT NULL | VARCHAR(255) |
| EVDEPT | NOT NULL | VARCHAR(255) |
| EVTYPE | NOT NULL | VARCHAR(255) |
| EVDATE | NOT NULL | VARCHAR(255) |
| EVDESC | NOT NULL | VARCHAR(1000) |
| EVVENUE | NOT NULL | VARCHAR(255) |
| EVCOUNTMAX | NOT NULL | INT(11) |
| EVCOUNT | NOT NULL | INT(11) |
| EVAVAILABILITY | NOT NULL | TINYINT(11) |
| EVORGID | NOT NULL | INT(11) |

### 3.1.3 EVENT_ORGANIZER Table

**Table 3.1.3: EVENTS_ORGANIZER Table**

| NAME | NULL | TYPE |
|---|---|---|
| EOID | NOT NULL | INT(11) |
| EONAME | NOT NULL | VARCHAR(255) |
| EODEPT | NOT NULL | VARCHAR(255) |
| EOEID | NOT NULL | INT(11) |

### 3.1.4 STUDENT Table

**Table 3.1.4: STUDENT Table**

| NAME | NULL | TYPE |
|---|---|---|
| SID | NOT NULL | INT(11) |
| SUID | NOT NULL | INT(11) |
| SNAME | NOT NULL | VARCHAR(255) |
| SDEPT | NOT NULL | VARCHAR(255) |

### 3.1.5 USER_CREDENTIALS Table

**Table 3.1.5: USER_CREDENTIALS Table**

| NAME | NULL | TYPE |
|---|---|---|
| UID | NOT NULL | INT(11) |
| UUSERNAME | NOT NULL | VARCHAR(255) |
| UPASSWORD | NOT NULL | VARCHAR(255) |
| UROLE | NOT NULL | INT(11) |

### 3.1.6 PARTICIPANTS Table

**Table 3.1.6: PARTICIPANTS Table**

| NAME | NULL | TYPE |
|---|---|---|
| PSID | NOT NULL | INT(11) |
| PEID | NOT NULL | INT(11) |

### 3.1.7 LOGS Table

**Table 3.1.7: LOGS Table**

| NAME | NULL | TYPE |
|---|---|---|
| LID | NOT NULL | INT(11) |
| LNAME | NOT NULL | VARCHAR(255) |
| LNAME | NOT NULL | VARCHAR(255) |
| LTIME | NOT NULL | DATETIME |

# 3.2 ER Diagram

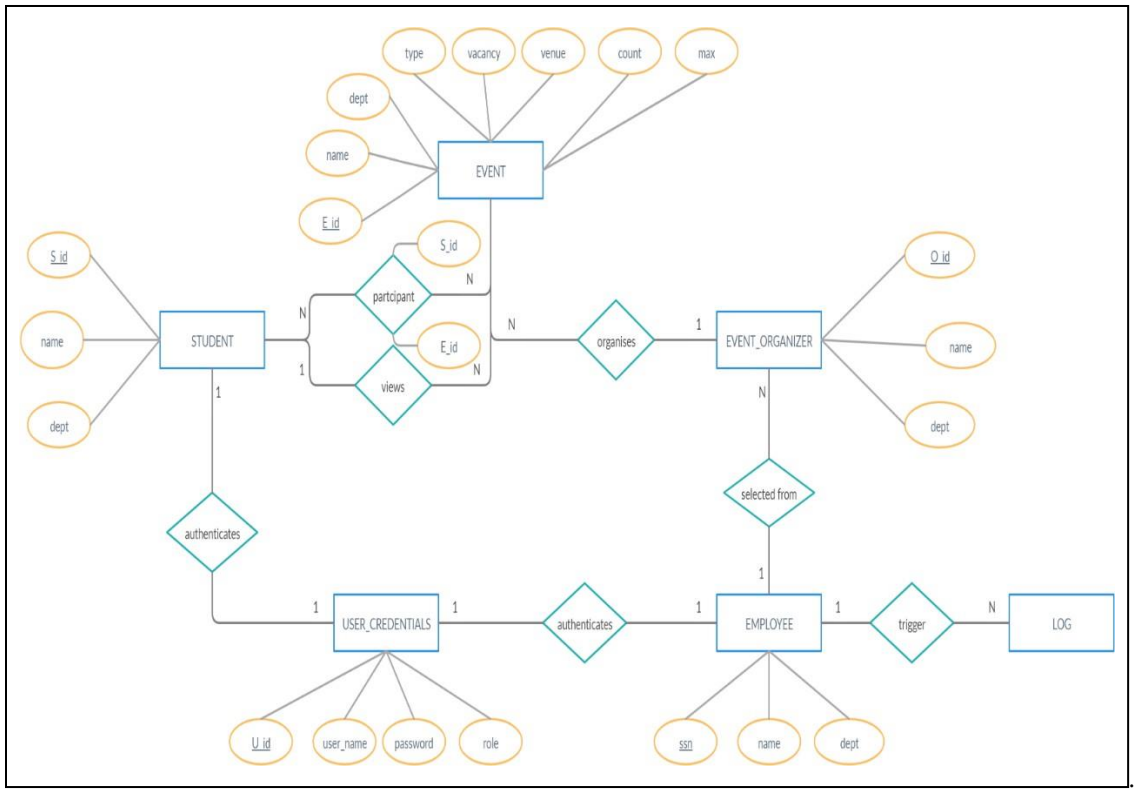



**Fig 3.2: ER Diagram**

# Chapter 4

# SYSTEM IMPLEMENTATION

System implementation uses the structure created during architectural design and the results of system analysis to construct system elements that meet the stakeholder requirements and system requirements developments development in the early life cycle phases. These system elements are then integrated to form intermediate aggregates and finally the complete system of interest (SOI).

Implementation is the process that actually yields the lowest-level system elements in the system hierarchy (system break down structure). System elements are made, bought or reused. Production involves the hardware fabrication processes of forming, removing, joining and finishing. The software realization process of coding and testing or operational procedures development processes for operator roles. If implementation involves a production process a manufacturing system which uses the established technical and management processes may be required.

## 4.1. Connection with Database

In order to establish the connection with the database following code is used. Here, localhost is the host name, root is the username and UEM is the name of the database created which stores all the tables and stored procedures.

**Source code**:

```
const express = require("express") const mysql = require("mysql");
//DataBase Credentials
const edb = mysql.createConnection({ host: 'localhost',
user: 'root', password: '', database: 'uem', port: '3306'
})
                //DataBase connection db.connect((err) => { if(err) throw err;
                        console.log(`Connected to uem Database`)
                 //Listening Port app.listen(port , () => {console.log(`Listening to Port 7000);
                 console.log(`Listening to 3306 `); )};
```

## 4.2. Login Page

In this page user is asked to enter the username and the password for authentication purpose. If the username and password is valid then the user is successfully logged in and redirected to the home page. If the username and password do not match then the user is shown a popup with an unsuccessful message and asked to re-enter user name and password.

### Source Code:

```
 getAuthCheck: (req, res) => {
 var username = req.body.username;
      var password = req.body.password;
db.query('SELECT Uid , URole FROM user_credencials WHERE Uusername = ?
AND Upassword = ?', [username,password], function(error, results, fields, rows) {
if (results.length > 0) {
                    req.session.loggedin = true;
                    req.session.username = username;
                    req.session.userid = results[0].Uid
                    req.session.userrole = results[0].URole

                    CheckID = results[0].URole
                    if(CheckID === 2)
                    {
                       db.query('SELECT * FROM employee', [username,password],
function(error, results, fields, rows) {
                          req.session.participants = results.length
                          res.redirect('/adminevents');
                       });
                    }
                    else if(CheckID === 1)
                    {
                       db.query('SELECT * FROM employee', [username,password],
function(error, results, fields, rows) {
                             req.session.participants = results.length
                             res.redirect('/events');});}
                    else res.send('<h1>SERVER DOWN <small> Please try
again<small><h1>')
                } else {
                    req.session.loggedin = false;
                    res.redirect('/loginFail');  }});}
```

## 4.3. Employee Events Page

This page lists the events and allows the employee to add new events and edit, delete, view particular event details and view the participants. It even shows the username of the organizer and their organization name. It even provides a search button to search for events by their names.

### Source Code:

```
  getAdminEventPage: (req,res) =>{
    if(req.session.loggedin === true)
    {
      var userID = req.session.userid
      db.query('SELECT E.Ename AS Ename,E.Edept AS Edept, EO.EOname AS
EOname FROM employee E , events_organizer EO WHERE E.EUid = ? AND E.Eid =
EO.EOid', [userID], function(error, results, fields, rows) {
        req.session.Ename = results[0].Ename
        req.session.EOname = results[0].EOname
        db.query('SELECT * FROM events', [userID], function(error, results, fields,
rows) {
          length = results.length
          res.render('Aevents.ejs',{
            title: 'AdminEvents',
            totalParticipants: req.session.participants,
            userName : req.session.Ename,
            userOrganizer : req.session.EOname,
            results,length
            })
        });
      });
    }
    else
    {
      res.redirect('/login');
    }
  }
```

## 4.4. Student Events Page

This page lists the events and allows the students to view event details. It even shows the username of the student. It even provides a search button to search for events by their names.

### Source Code:

```
getEventPage: (req,res) =>{
    if(req.session.loggedin === true)
    {
        var userID = req.session.userid
        db.query('SELECT * FROM student WHERE SUid = ?', [userID], function(error,
                results, fields, rows) {
            userName = results[0].Sname
            userDept = results[0].Sdept
            req.session.name = userName;
            lengthP = results.length
            db.query('SELECT * FROM events', [], function(error, results, fields, rows) {
                length = results.length
                db.query('SELECT * FROM student', [], function(error, resultsP, fields, rows) {
                    lengthPP = resultsP.length
                    res.render('events.ejs',{
                        title: 'Events',
                        userName : userName,
                        userDept : userDept,
                        totalParticipants : lengthPP,
                        results,length,resultsP,lengthPP})});});});}
         else{
        res.redirect('/login');
        }}
```

## 4.5. View Event Page

This page opens up when the employee or the student clicks on the "View" button in front of the event name in the event list. It shows the event details such as event name, description, set number of participants, select department, event type, set events venue , event date, count of participants and the availability such as the event is open or closed.

### Source Code:

```
getViewPage: (req,res) =>{
    var userID = req.session.userid
    var eventID = req.body.eventID
    if(req.session.loggedin === true && eventID)
    {
            db.query('SELECT * FROM student WHERE SUid = ?', [userID],
   function(error, results, fields, rows) {
        req.session.userPartID = results[0].Sid
        userPartID = req.session.userPartID
            db.query('SELECT * FROM participants WHERE Psid = ? AND Peid = ?',
   [userPartID,eventID], function(error, results, fields, rows) {
          if(results.length === 0) UserPartFlag = 0;
          else UserPartFlag = 1;
              db.query('SELECT * FROM events WHERE EVid = ?', [eventID],
   function(error, results, fields, rows) { length = results.length
            result = results[0]
                db.query('SELECT * FROM participants WHERE Peid = ?',
   [eventID], function(error, resultsPart, fields, rows) {
              lengthPart = resultsPart.length
              res.render('details.ejs',{
                title: 'EventsView',
                userName: req.session.username,
                result,length,eventID,UserPartFlag,lengthPart
                }) }); }); }); }); }
    else{ res.redirect('/login');}  }
```

## 4.6. Stored Procedure in MySQL

The stored procedure is saved in the backend MySQL. The given stored procedure takes zero input parameters**.** It retrieves LOGS table data to display the logged data.

**Source Code**:

```
CREATE PROCEDURE LogList();
SELECT * FROM logs;
```

## 4.7. Calling Stored Procedure

This page can be used to call the stored procedure in the server. The stored procedure **LogList()** takes zero input parameters. The stored procedure will then retrieve the Lid, Lname, Laction and Ltime.

**Source Code**:

```
getLogPage: (req,res) =>{
   if(req.session.loggedin === true)
   {
      db.query('LogList ()', [], function(error, results, fields, rows) {
         length = results.length
         res.render('logs.ejs',{
            title: 'AdminLogPage',
            userName : req.session.Ename,
            userOrganizer : req.session.EOname,
            results,length
            })
      });
   }
   else
   {
      res.redirect('/login');
   }
}
```

## 4.8. Trigger In MySQL

A trigger has been used in the LOGS table. When any of the record is inserted into EVENTS Table is i.e. insertion event on EVENTS happens then trigger in initiated after insertion, updation or delete takes place.

### Source Code:

```
//When events are INSERTED

DELIMITER $$
CREATE TRIGGER log_newEvents AFTER INSERT
ON events
FOR EACH ROW BEGIN
    INSERT INTO logs (Lname, Laction, Ltime) VALUES (NEW.EVname,"Event
Created", NOW());
END $$


//When events are UPDATED

DELIMITER $$
CREATE TRIGGER log_updateEvents AFTER UPDATE
ON events
FOR EACH ROW BEGIN
    INSERT INTO logs (Lname, Laction, Ltime) VALUES (NEW.EVname,"Event
Updated", NOW());
END $$


//When events are DELETED

DELIMITER $$
CREATE TRIGGER log_deleteEvents BEFORE DELETE
ON events
FOR EACH ROW BEGIN
    INSERT INTO logs (Lname, Laction, Ltime) VALUES (OLD.EVname,"Event
Deleted", NOW());
END $$
```

# Chapter 5

# SCREENSHOTS

## 5.1 Login Page

Login Page welcomes to College Events Management System where the user is given option to register by entering the credentials such as username, password, name and department, or if account exist then user can sign in using his username and password

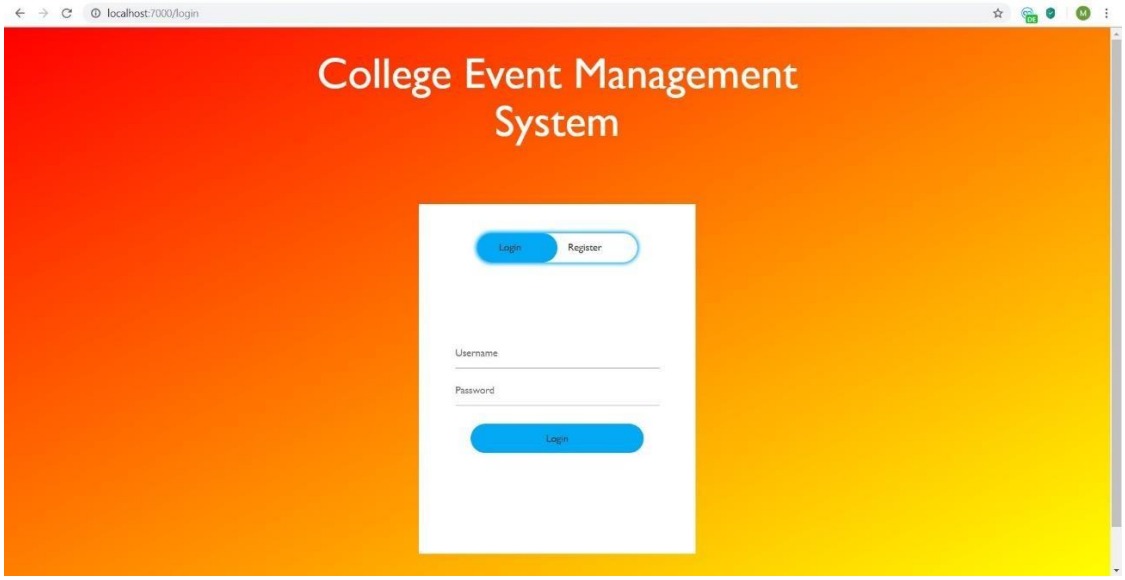



**Figure 5.1: Screenshot of Login Page**

## 5.2 Employee Events Page

This page contains the list of events followed by actions that can be taken upon them such as view, edit, delete and view the participants of that event. It even provides a search bar to search for the events by their names. Further, it displays the total count of events and organizers.

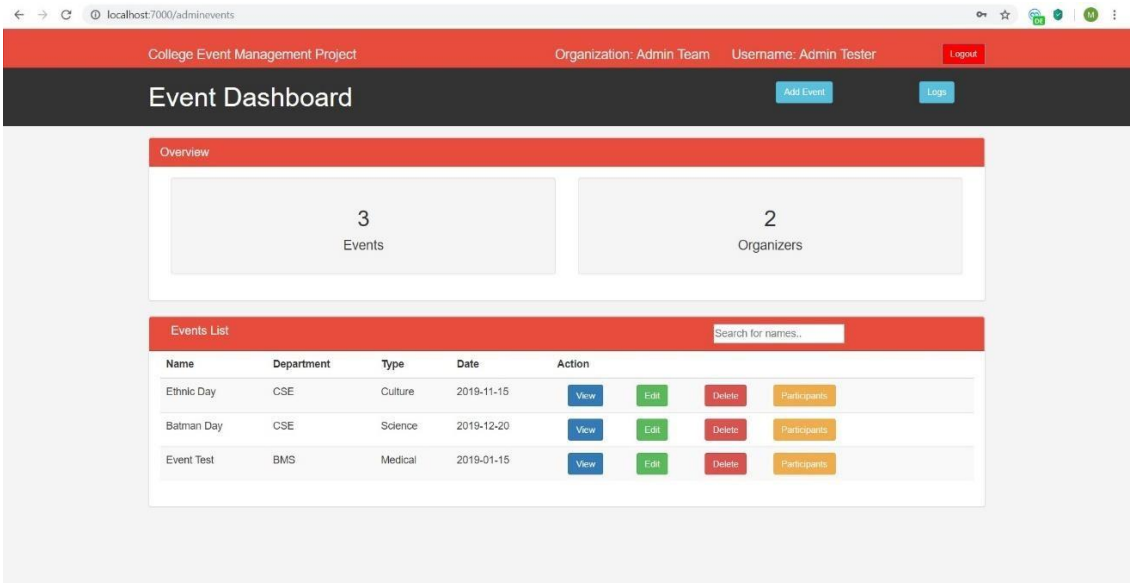



**Figure 5.2: Screenshot of Employee Events Page**

## 5.3 Student Events Page

This page contains the list of events followed by the action that can be taken upon it, i.e. view the event details. It even provides a search bar to search for the events by their names. Further, it displays the total count of events and students registered.

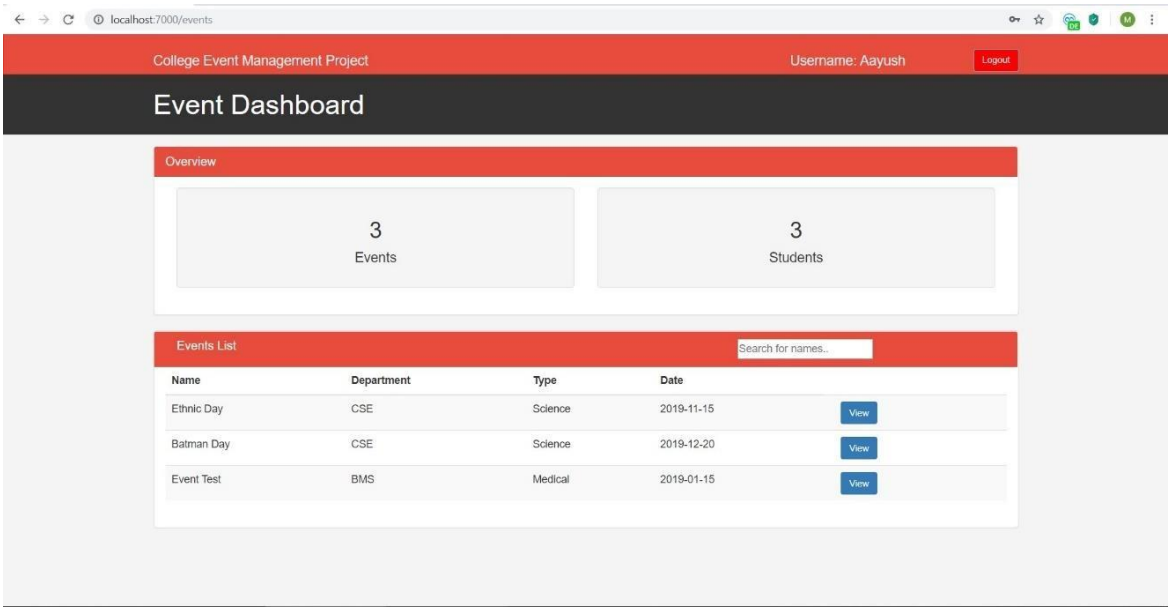



**Figure 5.3: Screenshot of Student Events Page**

# Chapter 6

# CONCLUSION

Managing events is now as simpler than ever before. With all the UI/UX being made for ease of managing and viewing Events and Income using the COLLEGE EVENTS MANAGEMENT SYSTEM. There are some limitations to this project that could be looked into such as getting the participants list in a PDF format or the ability to share as a mail. Limited options of departments it could be diversified to add more options deep to discover more options. Entry of these events data is still a manual process but in a digital form.

With websites being easily accessible, managing events with COLLEGE EVENTS MANAGEMENT SYSTEM would help out the college event managers and event participants to keep track of their events.

## Future Enhancement:

There is still additional room for future upgradation that would improve quality of the software and make it more useful and user friendly. Some of the features that can be enhanced are:

- Results Page can be added for students to easily check the result for competitive events.
- Profile could update all student details such as Semester and Overall Rating.
- Pop up message when the student is confirming his participation.
- Participate status at the student event view page
- Option to opt out of the event.

# BIBLIOGRAPHY

## Book References:

1. Database Systems Models, Languages, Design and Application Programming, RamezElmasri and Shamkant B. Navathe, 6th Edition, 2017, Pearson.
2. Database management systems, Ramakrishnan, and Gehrke, 3rd Edition, 2014, McGraw Hill
3. Fundamentals of Database System by Elmasri Ramez , Navathe Shamkant

## Website References:

- HTML: http://www.tutorialspoint.com/html http://www.w3schools.com/html
- CSS: https://www.w3schools.com/css http://www.htmldog.com/guides/css/beginner/margins/
- MySQL: http://tutorialspoint.com/mysql/index.htm https://www.w3schools.com/sql/default.asp
- JavaScript: http://www.w3schools.com/js