**DATABASE MANAGEMENT SYSTEMS**

*PROJECT REPORT ON:*

**COLLEGE EVENT MANAGEMENT SYSTEM**

***PROJECT BY:***

**VAISHNAVI RATHOD (E016)**

**RIYA TENDULKAR(E047)**

***UNDER THE GUIDANCE OF:***

**PANKTI DOSHI**

Department of Computer Engineering

Mukesh Patel School of Technology, Management, and Engineering



**MUKESH PATEL SCHOOL OF TECHNOLOGY**

**MANAGEMENT &ENGINEERING**

SVKM's

NARSEE MONJEE INSTITUTE OF MANAGEMENT STUDIES (Declared as Deemed-to-be University Under Section 3 of the UGC Act, 1956)

V. L. Mehta Road, Vile Parle (West)

MUMBAI -400056

**March 2020**

Subject: Database Management Systems

Semester IV, Year: II

**Academic Year: 2019-2020**

**ABSTRACT**

Productivity is never an accident. It is always a result of a commitment to excellence, intelligent planning, and focused effort. To maximise production of any task, a complete planning goes into it. With our project, we aim to make the life of students here at MPSTME, a tad bit easier. Event planning and managing is no kid’s play. A meticulous plan, organising team of 100+ people and working day and night results into our college events that are known far and wide in the city.

Every type of event is made up of numerous parts that fit together like pieces of a puzzle. All of those pieces ultimately come together to create an event. Successful events have all those related pieces coming together at the right place, and right time, smoothly, efficiently and according to plan. It is a given that just one person can’t organise an event at such a big scale. We have considered various aspects for the event management for our college like events, committees and their members, guests, resources, sponsors, and audience.

Event Management through database management systems is the new, upcoming method to get events done perfectly saving time, paper, money and human mechanical effort.

**TABLE OF FIGURES**

[Figure 1: ER Diagram 14](#_Toc36727421)

[Figure 2: Schema Diagram 17](#_Toc36727422)

[Figure 3: SQLite3 Software 26](#_Toc36727423)

[Figure 4: Flow of Scripts 29](#_Toc36727424)

**LIST OF TABLES**

[Table 1: Table of contents 4](#_Toc36727662)

[Table 2: Member Details 6](#_Toc36727663)

[Table 3: Add data table 10](#_Toc36727664)

[Table 4: User Table 12](#_Toc36727665)

[Table 5: Front End and Back End Details 21](#_Toc36727666)

[Table 6: SQLite3 Library Commands 25](#_Toc36727667)

**LIST OF ACRONYMS**

Acronym 1: DBMS- Database Management Systems

Acronym 2: SQL-Structured Query Language

Acronym 3: RDBMS-Relational Database Management System

Acronym 4: ER- Entity Relationship Diagram

Acronym 5: MPSTME- Mukesh Patel School Of Technology Management And Engineering

Acronym 6: GUI-Graphical User Interface

Acronym 7: NF- Normal Form

Acronym 8: BCNF- BOYCE AND CODD NORMAL FORM (BCNF)

Acronym 9: **IDLE**- **Integrated DeveLopment Environment**

Acronym 10: Ide: Integrated Development Environment

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TOPIC** | **PAGE NO.** |
| **1.** | **INTRODUCTION TO THE SYSTEM** | D |
| 1.1 | INTRODUCTION | D |
| 1.2 | PROBLEM STATEMENT | D |
| 1.3 | FUNCTIONAL REQUIREMENTS OF THE SYSTEM | D |
| 1.4 | USERS OF THE SYSTEM | D |
| **2.** | **SYSTEM DESIGN AND CONSTRAINTS** |  |
| 2.1 | ER MODEL | d |
| 2.2 | REDUCTION OF ER MODEL TO RELATIONAL MODEL | d |
| 2.3 | SCHEMA DIAGRAM | d |
| 2.4 | CONSTRAINTS |  |
| 2.5 | NORMALIZATION TECHNIQUES APPLIED ON RELATIONAL MODEL | d |
| **3.** | **IMPLEMENTATION** |  |
| 3.1 | HARDWARE AND SOFTWARE DETAILS (FRONT END AND BACK END DETAILS) | d |
| 3.2 | TOOLS OR LIBRARY USED | d |
| 3.3 | SCREENSHOTS AND DESCRIPTION |  |
| 3.4 | DATABASE STRUCTURE |  |
| **4.** | **CONCLUSION AND FUTURE WORK** | d |
| **5.** | **APPENDIX** |  |

Table : Table of contents

**CHAPTER 1: INTRODUCTION TO THE SYSTEM**

**1.1 INTRODUCTION TO DBMS**

### The developer builds an application or software, but software needs data to perform day to day operations and analytics over processed data and data is something which is driving the business nowadays to excel in their respective areas of operations. So a developer needs a database management system.

### A database management system (DBMS) refers to the technology for creating and managing databases. DBMS is a software tool to organize(create, retrieve, update and manage) data in a database. The main aim of a DBMS is to supply a way to store up and retrieve database information that is both convenient and efficient.

### The main components of the database systems are:

### 1.Hardware- all the physical devices that are being used in DBMS operations are called the hardware. For example if we run MySQL server, then the hard disk, RAM, the keyboard comes under the hardware component.

### 2.Software-this is one of the most important components as it contains all the programs that will drive the DBMS functionalities. The job of this software is to understand the database access language and interpret into actual database commands to execute them.

### 3.Data- the database management system collects, store, process and reads the data. This is yet another important component of the DBMS. The DBMS contains actual, operation or the metadata.

### 4.Procedures- procedure is the general instructions and rules that help in using a DBMS.

### 5.Database Access Language- it is a query language which is used to write commands to perform CRUD operations like create, read, update and delete.

**1.1 INTRODUCTION TO THE PROJECT**

### Running an event is a complex and costly exercise so the preparation and management of every detail counts. Keeping a proper database of the events is important to keep a record of specific details. It provides creative ideas and proven event formats that will work, **expert planning and budget management to ensure overall event success. It specifies technology that gives the attendees a professional and slick journey, providing a fulfilling ride in the events they attend, managing all risks and keeping them safe.**

### The main purpose of event management system is to provide a platform for the users to view the information about the events that took place in the past and the ones which are about to take place in the near future.

MPSTME has a lot of events happening throughout the year, both technical and non-technical. Other than these events we also have multiple fests like Sattva, Editorial Project, Taqneeq, etc. Storing these records physically and in file systems gets tedious, it is time consuming and inefficient considering the huge number of events held every year and large no of people attending them. Other than the events happening, it is required to store the information of the audience and performers. Being just students, the committees don’t have enough time to invest in maintaining a proper manual data in physical files. The aim of our project is to create a event management database system in which we make storing and accessing of data easier. The main entities of this database will be- Events, Committees, Guests, Audience, Sponsors and Resources.

The advantages of event managing database system is:

* Data of events can be shared between authorized user of database.
* It maintains data consistency. If a person needs to update some details about the events, it is much efficient in this system.
* The updated values are immediately available to users
* If a person needs to search for some events happening, this system makes this idea possible
* It provides a clear head for designing and managing the events, members organising it, guests performing in it and audience that attends it.

**Objectives of the project:**

* To create a model for database
* Add functionalities for the users
* Create appropriate views for users.
* To create a front end
* To link front end and back end

The members of this project and their contribution is:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Roll No.** | **Name** | **Contribution** | **Pages/Tables** | **Functionalities** |
| E016 | Vaishnavi Rathod | * Formatting GUI * Adding Database * Screenshots of outputs | * Admin * Login * College Audience * External Audience * Guests * Sponsors * Resources * Guests\_Events * Sponsors\_Events * Resources\_Events | * Search Event * Sort Event * Upcoming Events * Login |
| E047 | Riya Tendulkar | * Connecting All Scripts of Code * Report | * Events * Committee * Members * External * AudienceAudience\_Events * College\_Audience Events * Committee\_Members * Committee Main Page * Database Add Page * External Main Page * Main Page | * Total Cost * Remove Member * Popularity Page * Sponsor Display * Guest Display |

Table : Member Details

**1.2 PROBLEM STATEMENT**

Our first entity is **Events**. Every event has an event id which is the primary key. Every event will have a name, topic, date, budget, location, the year in which it happened, charge, website. The event can be academic or non-academic.

The event will be hosted by a **Committee**. Every committee has a committee id which is unique for each committee, a name, website, email-id, funds allocated to it and the head. For the committee to work every committee have some members who work for them.

**Committee Members** have a committee id and name, branch, course, year, phone number, email-id, birthdate, from the birthdate we can derive the age of the member, and the date of joining of the member. For every event to take place we need resources, sponsors and guests.

Every event has many **Guests** performing at the event. The guests have an id, name, phone number, email-id. To perform for each event the guests charge some amount to the committee.

Similarly, each event can have many **Sponsors** and the sponsors are repeated for many events. They have an id, name, website, address, and the type of sponsor they are. For any event to occur we need some resources.

The **Resource** has an id and name. Every event needs a certain quantity of resources.

Last but not the least we have the **Audience** who attend these events. Before attending the event, every person has to register. He/she will have an unique id. We will be required to store name, course, branch, year, email-id, phone number.

**1.3 FUNCTIONAL REQUIREMENTS OF THE SYSTEM**

We have 10 main functionalities in our project:

1. Total Cost of an Event
2. Remove Member
3. Search Events
4. Sort Events
5. Popularity of Events
6. Upcoming Events
7. Display Guests
8. Display Sponsors
9. Add Details
10. Login

**FUNCTIONALITY 1: TOTAL COST OF EVENT**

Money in events is a very important aspect. So, we give the committee the option to analyse the inflow and outflow of money.

BRIEF FLOW:

1. Calculate money spent on guests
2. Calculate money spent on resources
3. Add the total money spent
4. Calculate money received by charges paid by audience
5. Calculate to total money spent and balance (step 4-step 3)
6. Compare with the budget of the event
   1. If money left, add to the committee funds
   2. If less money, subtract money from committee funds

First, we display the events hosted by the committee and input the event whose monetary analysis is to be done.

From the ER diagram we know that every guest charges to appear for an event. From the guests\_events table we find the guests who had performed for that event and then calculate the total amount spent for guests.

Every event requires multiple resources and a specific quantity of them. So now for the event we find the resources required and the quantity ordered from the resource\_event table. We find the price per resource item from the resource item, multiply it with the quantity and calculate the total price spent on resources.

We add the prices spent on guests and resources, this is the total amount spent for the event.

From the audience tables, we calculate the total amount which the audience has paid. This will be the income for the event.

Now we subtract this value from the budget assigned to the event.

If money is left in the budget for event, it is added to the total budget for the event.

If final money is in negative values (more spent than the budget) then we subtract this amount from the budget of the committee.

**FUNCTIONALITY 2: REMOVE MEMBERS**

We are giving exclusive rights to the committee to remove members from the committee. The committee can search members from the committee depending on the following two parameters:

* Position- The committee will be first shown the distinct positions held by members in the committee. Then it has to select the position of which the member he wants to delete. It will be shown the members in the committee who hold the post. The committee can select the id of the member to be deleted.
* Year- The committee can select which year’s members he wants to delete- 1st, 2nd ,3rd,4th and 5th. The members of the committee in the selected year are shown and can be deleted.

**FUNCTIONALITY 3: SEARCH**

In this functionality, the drop down menu gives the user a choice to search for events based on their prices or their committees. In the first menu, we have events that are free of cost, events with a specific price that will be requested by the user and any price that will arrange the prices of events in the ascending or increasing order. In the next menu we have the option to search the events based on the committee the user enters.

**FUNCTIONALITY 4: SORT**

For this functionality we have a menu drop down with the options of past and future events. It carefully arranges and displays to the user the events that were held in the past of the year entered by the user. It also can display the future or next events coming up after the year the user inputs.

**FUNCTIONALITY 5: POPULARITY OF EVENT**

In this we display how many people had attended the event. We first take the input of the name of the event and the year it took place. It is important to take the year of the event as many events are annual.

So, from the event name and year, we extract the e\_id of the event. Then from the external and college audience table we count the number of people who had attended the event.

**FUNCTIONALITY 6: UPCOMING EVENTS**

This functionality displays the upcoming events of the college. If any external user wants to see the events (of any type) he can use this functionality to see the events. This functionality will search all the events from the events table happening in that particular year and the years ahead of that. It uses the current date of the system when the functionality is used and displays the events after that registered in the database.

**FUNCTIONALITY 7: DISPLAY GUESTS**

Any external user can see the guests who will be performing at a specific event. We take the event name and year as input and from that we fetch the e\_id from the events table, then we retrieve the guests from the guest\_event table.

**FUNCTIONALITY 8: DISPLAY SPONSORS**

**@riya add**

**FUNCTIONALITY 9: ADD DETAILS**

Using this functionality, we have the option of adding the data directly to the database. The adding of data has restrictions and only admin is allowed to enter the details of the entities. The add pages along with the user who is allowed to do access it is as follows:

|  |  |
| --- | --- |
| **TABLE** | **USER** |
| Audience | Admin, External User(to register) |
| Sponsors to Events | Committee |
| Resources to Events | Committee |
| Guests to Events | Committee |
| Event | Admin |
| Committee | Admin |
| Members | Admin |
| Guests | Admin |
| Audience | Admin |
| Sponsors | Admin |
| Resources | Admin |
| Admin | Admin |

Table : Add data table

**Add sponsors/resources/guests to events:**

When any event is hosted, there are sponsors, resources and guests. The details of these entities (id, name, number, etc) are already stored in individual tables. When any event is hosted, they will have different guests, sponsors and resources. So, we have separate tables wherein we have the ids of sponsors, resources and guests and the event id. These tables keep a track of which event had what and how many sponsors, guests and resources. This functionality is given exclusively to the committee as they host and plan the events.

**FUNCTIONALITY 10: LOGIN**

Adding the details to the database has exclusive access only and so we first need to confirm if the admin is authorized. For that we take input the username and password of the admin and then cross check the details from the database. If the credentials entered are correct then the admin is given further access.

**1.4 USERS OF THE SYSTEM**

We have 3 users for this system- **External User, Committee and Admin**.

**External User**- The external user is any person who wants to see the event details. Our entire database is a collection of events- of the past and upcoming and so any external user should be able to see these events and its details. The user is given an option to search and sort the events according to certain parameters. After seeing the events, if the user wishes to register for the event, he can do so.

**Committee**- Committee is the entity which hosts the event and so it is necessary to give them the access of the database. Like the external user can see the specific details of the event, the committee can see the entire backend of the event- committee members, sponsors, resources, audiences, guests and the respective details of all of them. Also, we give them the option of deleting members from the database, that is, if they feel any committee member needs to be removed, they have the authority to do so.

**Admin**- The admin is the core user of this database and to access this part there is a security check. The admin can add any data to the database and has no restriction.

|  |  |  |
| --- | --- | --- |
| **USER** | **ACCESS** | **FUNCTIONALITY** |
| External | View and Register for events only | * View events * Search events * Sort events * Register * See guests, sponsors |
| Committee | View and delete | * View entire database * Delete committee   Members   * Calculate the costing of an event |
| Admin | Add only | * Can add data to the database |

Table : User Table

**CHAPTER 2: SYSTEM DESIGN AND CONSTRAINTS**

**2.1 ER DIAGRAM**

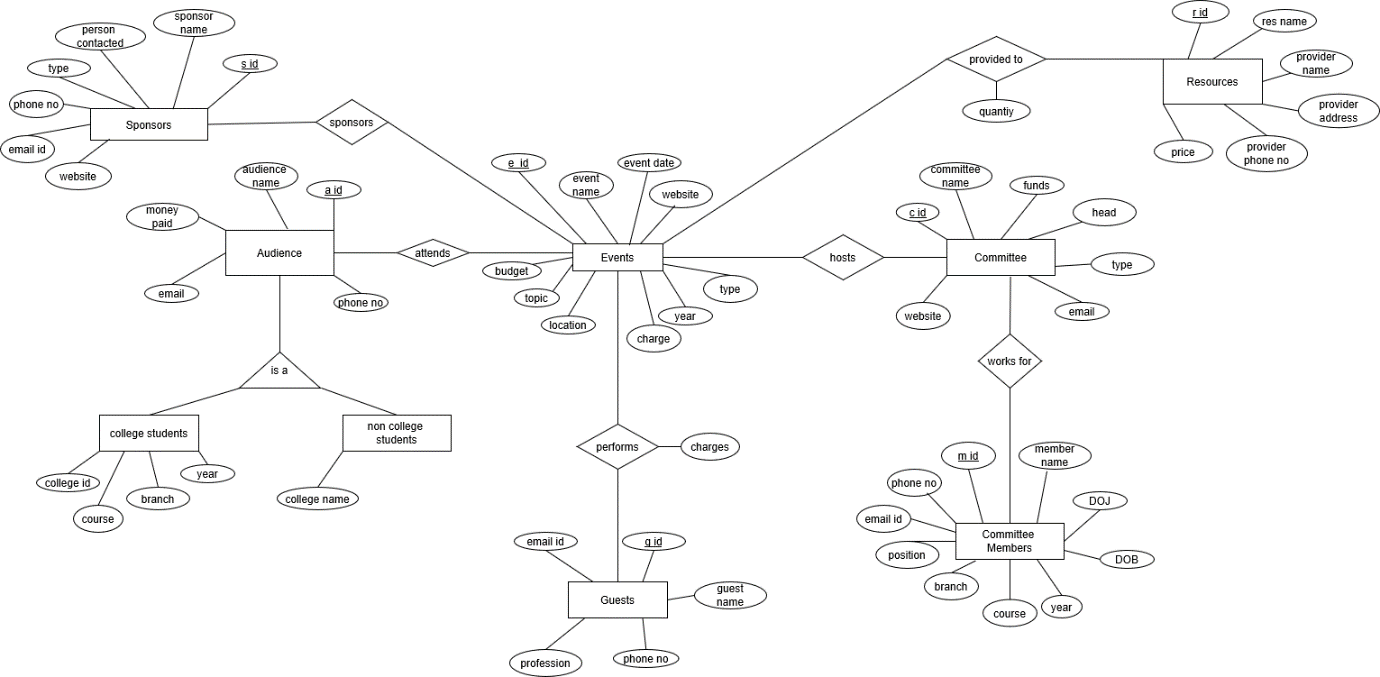


Figure : ER Diagram

As per the model, we have different entities with their attributes that duly entail for the event management database system. We have carefully picked up the requirement of every detail in the system according to the working of any event management that takes place in the college. The entities and their attributes are:

* **EVENT** which contains e\_id (primary key) which is an autoincrement attribute that will store the unique number given or any event stored in the database. Event\_name stands for the name of the event it is called by,decided by the people organising it. event\_date is when the event did or would take place. We use attributes to store email id, phone no, website, budget, the location, charge, type (technical or non-technical) and year of the event.
* **COMMITTEE** hosts the events and contain c\_id (primary key) autoincrement attribute to store unique nos of committees in the college. It has other attributes like name, funds, website, email id, type (student council or not) of the committee and who is it headed by.
* **COMMITTEE MEMBERS** are literally the backbone of any event. M\_id(primary key) stores autoincremented unique no to identify any committee member of the college. The other attributes are their name, date of birth, the date they joined the committee, phone no, email id, position in the committee, course, branch and their year.
* **RESOURCES** like chairs, lights etc are required for any event. It has r\_id is the primary key with the name of the resource, its price, the name, phone no and address of the provider for the resources.
* **GUEST** are invited to the event to make it more spectacular and contain attributes like g\_id(primary key), guest name, their email id and phone no and the profession they belong to. Sometimes a guest might charge a particular fee like charge to perform at the event.
* **AUDIENCE** are the ones that come to watch the event and based on the turn out its decided whether the event was good, bad or amazing. They have attributes like a\_id(primary key), audience name, phone no, email id and the money they paid for the event. There can be 2 types of audiences here: belonging to our college or outside the college. Our college students will have attributes like college id(like sap no), branch, course and year. The non-college students will have to state their college name.
* **SPONSORS** sponsor the event to advertise their own businesses and help out the students with the event costs. We have s\_id(primary key), the name of the sponsors, type, person contacted at the firm for a particular event, the phone no, email id, website of the sponsors

**CHAPTER 2.2 RELATIONAL MODEL**

Upon analysing the ER diagram, we make the following relational model:

1. Events ( e\_id, name, website, date, year, budget, topic, location, charge, type, c\_id)
2. Committee ( c\_id, name, head, funds, email id, website, type)
3. Members ( m\_id, name, phone\_no, emailId, position, branch, course, year, DOB, DOJ, c\_id )
4. Guests (g\_id, name, phone no, profession, email\_id)
5. Sponsors (s\_id, name, address, phone number, email\_id, website, type)
6. Resources (r\_id, resource name, price, provider name, provider phone number, provider address)
7. College\_Audience ( a\_id, name, email\_id, phone no, branch, year, course, money paid, college\_id, e\_id)
8. External\_Audience ( ea\_id, name, email\_id, phone no, college\_name, money paid, e\_id)
9. Guests\_Events (g\_id, e\_id, charge)
10. Resource\_Event (r\_id,e\_id,quantity)
11. Sponsor\_Event (e\_id,s\_id)
12. Committee\_Members (c\_id,m\_id)
13. CAudience\_Event ( a\_id, e\_id)
14. EAudience\_Event (ea\_id, e\_id)

**CHAPTER 2.3 SCHEMA DIAGRAM**

**Guests**

* g\_id
* name
* phone no
* profession
* email\_id

**Committee\_Members**

* c\_id
* m\_id

**Resources**

* r\_id
* resource name
* price
* provider name
* provider phone\_no
* provider address

**Members**

* m\_id
* name
* phone\_no
* emailId
* position
* branch
* course
* year
* DOB
* DOJ
* c\_id

**Committee**

* c\_id
* name
* head
* funds
* email id
* website
* type

**Guests\_Events**

* g\_id
* e\_id
* charge

**Resource\_Event**

* r\_id
* e\_id
* quantity

**Events**

* e\_id
* name
* website
* date
* year
* budget
* topic
* location
* charge
* type
* c\_id

**CAudience\_Event**

* a\_id
* e\_id

**Sponsor\_Event**

* e\_id
* s\_id

**Sponsors**

* s\_id
* name
* address
* phone number
* email\_id
* website
* type

**College\_Audience**

* a\_id
* name
* email\_id
* phone no
* branch
* year
* course
* money paid
* college\_id
* e\_id

**ExternalAudience**

* ea\_id
* name
* email\_id
* phone no
* college\_name
* money paid
* e\_id

**EAudience\_Event**

* ea\_id
* e\_id

Figure : Schema Diagram

**2.4 CONSTRAINTS**

**What are constraints?**

Constraints are the rules used to limit the type of data that can go into the table, to maintain the accuracy and integrity of the data inside the table. Constraints can be divided into the following two types-

1. Column level constraints: limits only column data
2. Table level constraints: limits only table data

**What are the most used constraints applicable to any table?**

Some of the most common constraints are-

Not null, Unique, Primary key, Foreign key, Check and Default.

**So what are the constraints we have used in our tables?**

* Events : e\_id is the primary key, c\_id is the foreign key, name is not null.
* Committee: c\_id is the primary key, name is not null, head is also not null.
* Members: m\_id is the primary key, c\_id is the foreign key, name is not null, phone no is unique.
* Guests: g\_id is primary key.
* Sponsors: s\_id is the primary key, name is not null, email id and website are not null as well as phone no is not null.
* Resources: r\_id is primary key, name, provider phone no are not null.
* College\_Audience: a\_id is primary key, e\_id is the foreign key, name, email id, phone no, college id are not null.
* External\_Audience: ea\_id is the primary key, e\_id is the foreign key, name, college name, email id and phone no are not null.
* Guests\_Events: g\_id and e\_id are the foreign keys.
* Resource\_Event: r\_id and e\_id are the foreign keys.
* Sponsor\_Event: e\_id and s\_id are foreign keys.
* Committee\_Members: c\_id and m\_id are foreign keys.
* CAudience\_Event: a\_id and e\_id are foreign keys.
* EAudience\_Event: ea\_id and e\_id are foreign keys.

**2.4. NORMALIZATION TECHNIQUES APPLIED ON RELATIONAL MODEL**

**What is normalization?**

**Normalization** is a database design technique that organizes tables in a manner that reduces redundancy and dependency of data. Normalization divides larger tables into smaller tables and links them using relationships. The purpose of Normalization is to eliminate redundant (useless) data and ensure data is stored logically.

Normalization is used to avoid insertion, deletion and update anomaly.

**Three forms of normalization:**

**FIRST NORMAL FORM (1NF)**

For a table to be in the First Normal Form, it should follow the following 4 rules:

1. It should only have single(atomic) valued attributes/columns.
2. Values stored in a column should be of the same domain
3. All the columns in a table should have unique names.
4. And the order in which data is stored, does not matter.

From the ER, we see that the committee members have an attribute- email\_id which is multivalued. This violates 1NF form and so we try to convert it. To do so we can make another table just for email ids. This table will have the primary key of the main table, so a new table : member\_emailID(e\_id,emailID) will be created where e\_id will be the key of the members table and the primary key of the new table will be {e\_id,emailID}

All other tables satisfy 1NF.

**SECOND NORMAL FORM (2NF)**

For a table to be in the Second Normal Form,

1. It should be in the First Normal form.
2. And, it should not have Partial Dependency.

From the relational model we can see that there are no such tables where there is a composite primary key but and partial dependency holds true. Hence we can say that no table has partial dependency and hence all tables are 2NF.

**THIRD NORMAL FORM (3NF)**

A table is said to be in the Third Normal Form when,

1. It is in the Second Normal form.
2. And, it doesn't have Transitive Dependency.

all tables are in 2NF and since there is no composite primary key there is no transitive dependency. Hence, we can say that all tables are 3NF.

**BOYCE AND CODD NORMAL FORM (BCNF)**

Boyce and Codd Normal Form is a higher version of the Third Normal form. This form deals with certain type of anomaly that is not handled by 3NF. A 3NF table which does not have multiple overlapping candidate keys is said to be in BCNF. For a table to be in BCNF, following conditions must be satisfied:

* R must be in 3rd Normal Form
* and, for each functional dependency ( X → Y ), X should be a super Key.

All the tables are in 3NF and there is no table having multiple overlapping candidate keys. So, the tables are in BCNF.

We have tried our best to form the tables in such a way as to avoid redundancy of data and to try to make the data is such a way that it is easy to update, delete and insert data. Thus, our model requires minimal normalization techniques.

**CHAPTER 3: IMPLEMENTATION**

**3.1 FRONT END AND BACK END DETAILS**

To implement the front end and back end of our project, we have used some languages and software. The summary of the same are as follows:

|  |  |  |
| --- | --- | --- |
| **Category** | **Software/Language** | **IDE** |
| Front End | Python | Python IDLE |
| Back End | RDBMS | SQLite3 |
| Back End | Browser | DB Browser |

Table : Front End and Back End Details

The reasons for choosing these IDEs and languages is mentioned in further sections. The details about the language is mentioned in the 3.1 chapter and the details of the IDE are mentioned in the 3.2 chapter.

**3.1.1 FRONT END**

**LANGUAGE USED: PYTHON**

Python is a high level, interpreted and general-purpose dynamic programming language that focuses on code readability. It has fewer steps when compared to Java and C. It was founded in 1991 by developer Guido Van Rossum. It is used in many organizations as it supports multiple programming paradigms. It also performs automatic memory management.

Advantages of using python:

1. Presence of third-party modules
2. Extensive support libraries (NumPy for numerical calculations, Pandas for data analytics etc)
3. Open source and community development
4. Easy to learn
5. User-friendly data structure
6. High-level language
7. Dynamically typed language (No need to mention data type based on value assigned, it takes data type)
8. Object-oriented language
9. Portable and Interactive
10. Portable across Operating systems

Why we chose Python over other languages like c++ and java:

1. **Easy database connectivity**

The main of our project is to implement a Database system and to implement any functionality you need a good front end so it is easy for any user to understand the flow of the code. For this we needed a language which has a strong and easy database connectivity.

Database connectivity using python is much easier as compared to c++ and java.

You can connect to any database by importing the required library and writing a few commands. Whereas for languages like C++ it gets very difficult to find a library for connection and writing the commands.

1. **GUI**

GUI using C++ and Java is very tedious. Python has a very easy to implement GUI. We use the library Tkinter and by using a few lines of codes we can make a nice GUI. With a click of a button we can move on to the next page and this helps in a better flow of the program. We have many functionalities and they are required to be run in a flow, like from the main page we can go to the admin section. Committee section or the external user section. Now to move and back to these pages we needed a GUI which made it easy for us travel through them.

1. **Libraries**

Python has a rich variety of libraries. Python downloads with an extensive library and itcontain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don’t have to write the complete code for that manually. We have used almost 2 libraries in every code of our project- SQLite3 and Tkinter and they have made writing this code very easy.

1. **Easy to read and write**

The syntax of Python is very easy to read and write. It uses simple words and the codes are like the English language. Since our code consists of over 10 python scripts we needed a language which would be easy to write and as 2 different people are doing the project, we needed it to be such that the other person will be easily able to understand the code. Also connecting all these scripts was very easy as all we had to do was write an import statement.

**The details of the tool (IDE used) is mentioned in the next chapter.**

**3.1.2 BACK END**

**LANGUAGE USED: RDBMS- SQL**

RDBMS Stands for "Relational Database Management System." An RDBMS is a [DBMS](https://techterms.com/definition/dbms) designed specifically for [relational databases](https://techterms.com/definition/relational_database). Therefore, RDBMSes are a subset of DBMSes.

A relational database refers to a [database](https://techterms.com/definition/database) that stores data in a structured format, using [rows](https://techterms.com/definition/row) and [columns](https://techterms.com/definition/column). This makes it easy to locate and access specific values within the database. It is "relational" because the values within each [table](https://techterms.com/definition/table) are related to each other. Tables may also be related to other tables. The relational structure makes it possible to run [queries](https://techterms.com/definition/query) across multiple tables at once.

**SQL**

**SQL** is a database computer language designed for the retrieval and management of data in a relational database. **SQL** stands for **Structured Query Language**.

SQL is a computer language for storing, manipulating and retrieving data stored in a relational database. SQL is the standard language for Relational Database System.

## **Applications of SQL**

As mentioned before, SQL is one of the most widely used query language over the databases. The applications of SQL are:

* Allows users to access data in the relational database management systems.
* Allows users to describe the data.
* Allows users to define the data in a database and manipulate that data.
* Allows to embed within other languages using SQL modules, libraries & pre-compilers.
* Allows users to create and drop databases and tables.
* Allows users to create view, stored procedure, functions in a database.
* Allows users to set permissions on tables, procedures and views.

**3.2 TOOLS AND LIBRARIES USED**

**3.2.1 TOOLS AND LIBRARIES FOR FRONT END: PYTHON IDLE**

The**Python IDLE** (**Integrated DeveLopment Environment**) **editor** is a graphical user interface for Python development. This GUI is free and installed automatically during the Python installation. It enables you to edit, run, and debug Python programs in a simple GUI environment.

IDLE is actually a Python program that uses the standard library’s **tkinter** **GUI toolkit** to build its windows. It is portable and can be run on all major platforms, such as Windows, Linux, Mac OS, etc. It supports the following features:

* command history and syntax colorization
* auto-indent and unindent for Python code
* word auto-completion
* support for multiple windows
* integrated debugger

IDLE is very easy to implement and as this is our first time with creating front end, we decided to use IDLE for our project.

In Python IDLE, we use 2 main libraries for our project: Tkinter Library and SQLite3 Library.

**LIBRARY 1: TKINTER**

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Creating a GUI using tkinter is an easy task.  
**To create a tkinter app:**

1. Importing the module – tkinter
2. Create the main window (container)
3. Add any number of widgets to the main window
4. Apply the event Trigger on the widgets.

Import statement: from tkinter import \*

**LIBRARY 2: SQLITE3**

SQLite is a C library that provides a lightweight disk-based database that doesn’t require a separate server process and allows accessing the database using a nonstandard variant of the SQL query language. Some applications can use SQLite for internal data storage. It’s also possible to prototype an application using SQLite and then port the code to a larger database such as PostgreSQL or Oracle.

We can import the sql library using a simple statement: import sqlite3

To connect the program to a specific database we use the statement:

conn = sqlite3.connect('example.db')

where conn is the [Connection](https://docs.python.org/2/library/multiprocessing.html#Connection) object that represents the database.

Once you have a [Connection](https://docs.python.org/2/library/multiprocessing.html#Connection), you can create a [Cursor](https://docs.python.org/2/library/sqlite3.html#sqlite3.Cursor) object and use this to execute commands.

Some important commands used for this library are:

|  |  |
| --- | --- |
| **COMMAND** | **FUNCTION** |
| cursor.execute() | To execute queries in the database |
| cursor.fecthall() | Fetch all the data selected |
| cursor.fetchone() | Fetch single row of data |
| cursor. commit() | Make permanent changes to the database. |
| connection.rollback() | This method rolls back any changes to the database since the last call to commit() |
| connection.close() | This method closes the database connection. |

Table : SQLite3 Library Commands

**3.2.2 TOOLS FOR BACK END: SQLITE3**

SQLite is a very popular database which has been successfully used with on disk file format for desktop applications like version control systems, financial analysis tools, media cataloging and editing suites, CAD packages, record keeping programs etc. **SQLite** is a [relational database management system](https://en.wikipedia.org/wiki/Relational_database_management_system) (RDBMS) contained in a [C](https://en.wikipedia.org/wiki/C_(programming_language)) [library](https://en.wikipedia.org/wiki/Library_(computer_science)). In contrast to many other database management systems, SQLite is not a [client–server](https://en.wikipedia.org/wiki/Client%E2%80%93server) database engine. Rather, it is embedded into the end program. SQLite is [ACID](https://en.wikipedia.org/wiki/ACID)-compliant and implements most of the [SQL](https://en.wikipedia.org/wiki/SQL) standard, generally following [PostgreSQL](https://en.wikipedia.org/wiki/PostgreSQL) syntax. SQLite is a popular choice as [embedded database](https://en.wikipedia.org/wiki/Embedded_database) software for local/client storage in [application software](https://en.wikipedia.org/wiki/Application_software) such as [web browsers](https://en.wikipedia.org/wiki/Web_browser). It is arguably the most widely deployed [database engine](https://en.wikipedia.org/wiki/Database_engine), as it is used today by several widespread browsers, [operating systems](https://en.wikipedia.org/wiki/Operating_system), and [embedded systems](https://en.wikipedia.org/wiki/Embedded_system) (such as mobile phones), among others. SQLite has [bindings](https://en.wikipedia.org/wiki/Language_binding) to many programming languages.

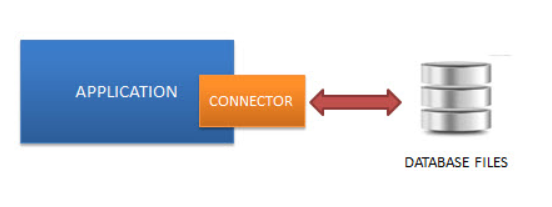


Figure : SQLite3 Software

Normally, an RDBMS such as MySQL, PostgreSQL, etc., requires a separate server process to operate. The applications that want to access the database server use TCP/IP protocol to send and receive requests. This is called client/server architecture. SQLite does not work this way and does not require a server to run. SQLite database is integrated with the application that accesses the database. The applications interact with the SQLite database read and write directly from the database files stored on disk.

There are a lot of advantages to use SQLite as an application file format:

**1) Lightweight-** SQLite is a very light weighted database so, it is easy to use it as an embedded software with devices like televisions, Mobile phones, cameras, home electronic devices, etc.

**2) Better Performance -** Reading and writing operations are very fast for SQLite database. It is almost 35% faster than File system. It only loads the data which is needed, rather than reading the entire file and hold it in memory. If you edit small parts, it only overwrite the parts of the file which was changed.

**3) No Installation Needed-** SQLite is very easy to learn. You don’t need to install and configure it. Just download SQLite libraries in your computer and it is ready for creating the database.

**4) Reliable -** It updates your content continuously so, little or no work is lost in a case of power failure or crash. SQLite is less bugs prone rather than custom written file I/O codes. SQLite queries are smaller than equivalent procedural codes so, chances of bugs are minimal.

**5) Portable -** SQLite is portable across all 32-bit and 64-bit operating systems and big- and little-endian architectures. Multiple processes can be attached with same application file and can read and write without interfering each other. It can be used with all programming languages without any compatibility issue.

**6) Accessible -** SQLite database is accessible through a wide variety of third-party tools. SQLite database's content is more likely to be recoverable if it has been lost. Data lives longer than code.

**7) Reduce Cost and Complexity-** It reduces application cost because content can be accessed and updated using concise SQL queries instead of lengthy and error-prone procedural queries. SQLite can be easily extended in in future releases just by adding new tables and/or columns. It also preserve the backwards compatibility.

**3.2.3. BROWSER USED: DB BROWSER**

It is not possible to read the data from SQLite directly as the format in which the data is stored is not understandable. Hence, we need a browser to help us to read the data stored. For this purpose we use the DB Browser.

**What is SQLite Browser?**

DB Browser for SQLite is a high quality, visual, open-source tool made for creating, designing, and editing database files that are compatible with SQLite. It is for users and developers who want to create, search, design and edit [databases](https://www.edureka.co/blog/what-is-a-database/). SQLite browser uses a general spreadsheet-like interface, and there is no need to learn complicated [SQL commands](https://www.edureka.co/blog/sql-commands). It is a tool that is used by both developers and end-users, and for that reason, it has to remain as simple as possible.

## **Uses of SQLite Browser**

It is a tool that lets us view the data that is stored in an SQLite Database. Depending on the format and type of data in the database it may or may not be readable by a human. This is generally used for debugging or other development tasks where the developer needs to read the data that has been stored but does not have a built-in system to access it through the program.

**Some commands that are available in SQLite browsers for users to:**

* Create and compact database files
* Create, define, modify and delete tables
* Create, define and delete indexes
* Browse, edit, add, search and delete records
* Import and export records as text
* Import and export tables from/to CSV files
* Issue [SQL queries](https://www.edureka.co/blog/insert-query-sql/) and inspect the results
* Examine log of all the SQL commands issued by the application

**3.3.1. FLOW OF SCRIPTS**

See upcoming events

**MAIN PAGE**

Sort events

External User

**EXTERNAL USER MAIN PAGE**

Search events

Committee

Popularity of event

Admin

Guest of event

**Check credentials**

Sponsor of event

**Check Committee Name**

Register for event

**INSERT DATABASE**

Calculate cost of event

Event

Add member

Committee

Committee Member

Remove member

Guest

Add sponsor to event

Audience

Add resource to event

Sponsors

Resources

Add guests to event

Admin

Figure : Flow of Scripts

**3.3.2. SCRIPTS DESCRIPTION**

1. **Main Page**

This is the main page of the project. It connects the project to the three users: external audience, admin and committee.

Add a sample screenshot

1. **Login Page**

This page checks if the admin trying to access is authorized to do so. We check the credentials (username and password) and if they are correct, further access is allowed.

Add a sample screenshot

1. **Add Database Page**

This page gives an option to which table you want to add the data. From this page you are connected to multiple pages which are used to add data to the tables.

1. **Admin/College Audience / External Audience/ Committee/ Events/ Guests/Members/ Resource/ Sponsors**

These scripts are used to add data to the specific tables. A sample page is shown below:

Add a sample screenshot

1. **Audience Main Page**

There are 2 types of audience- College Audience and Non-College Audience. So, this page connects us to those pages which are used for adding database.

Add a sample screenshot

1. **Committee Main Page**

This is used to connect all the functionalities of the committee member. From this page we can go to the individual pages. Before that, we ask for the committee name and then check from the database if the committee exists. If yes, further access is allowed and in the further pages, the details of only that committee is displayed.

Add a sample screenshot

1. **Calculate cost of event**

This page first displays the events of the committee and then inputs the e\_id of the event whose cost is to be calculated. Then it displays the details of the cost management. (Refer page )

1. **Add sponsors to event/add resources to event/add guests to event.**

These pages are used to add sponsors, resources and guests to events.

Add a sample screenshot

1. **External Main Page**

This is the main page of the external user. This page connects the other functionalities of the external user and gives the menu option.

Add a sample screenshot

1. **See upcoming events**

Take inputs of the year and will display the events which will be held after that year.

1. **Sort events**

Gives an option to sort events – using past and upcoming years.

1. **Search Events**

Gives the option of searching an event using two parameters: Charges of event and Committee.

In charges it you can search using free events or a specific price or both.

In committee we can see the events of a specific committee.

1. **See popularity of events**

This page shows the amount of people who attended the event upon taking the name and year of event.

1. **See guests of an event**

Takes input of the name and year and displays the guests performing at the event.

1. **See sponsors of an event**

Shows the events sponsored by a specific sponsor.

**3.4 SCREENSHOTS**

**CHAPTER 5: CONCLUSION AND**

**FUTURE WORK**

We have created a working event management database system as our project this year complete with front end and back end to provide the people what they need. Planning of any task becomes complete with an organised work force. The desired organised data will be sufficed by the database system we have created. It keeps all data under one roof with scope for creating (new ideas that pop into the head), updating (implementing some old ideas with a new outlook), deleting (some ideas might not work every time) and exploring the data stored under various categories and functions. It is absolutely suitable for any life, fast paced or not. It is suitable for someone who wants to get their work done but doesn’t want any unnecessary fuss with it. It is suitable for any student who just wants to organise an event and wants it to be great and successful.

**FUTURE WORK**

We wish to collaborate with the Student Council and other committees of MPSTME and make this a proper working model. We feel that our project has a lot of scope and will be very helpful for our college which has a huge number of events held every year with a very large turnout. The current model is created with our understanding of how event management works and we wish to take their inputs on this so as to cater to specific needs.

We also wish to implement more functionalities in this model and increase the database. Other thing we want to focus on is to improve the GUI of the model and make the front end better.

**@VAISHNAVI**

1. **Add abstract**
2. **Add screenshots**
3. **Add proper er diagram**
4. **Add constraints**
5. **Add database structure pics with 2 lines of the tables( idk if this is required since we are giving description in er itself)**
6. **After you add everything please add page numbers in the table of contents**
7. **Refer all the parts where I have yellow highlights and fill in your explanations.**

**Changes in er:**

**Cardinality**

**MAKE EMAIL ID OF MEMBERS AS MULTIVALUED ATTRIBUTE (imp)**