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

SVKM'S
NMIMS
Deemed-to-be UNIVERSITY

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

LIST OF TABLES

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

Table 1: 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	• 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 2: 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
   a. If money left, add to the committee funds
   b. 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

In this functionality we can see which sponsor sponsors which event. We can see the sponsors using sponsor name and type.

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 3: 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 4: User Table

CHAPTER 2: SYSTEM DESIGN AND CONSTRAINTS

2.1 ER DIAGRAM

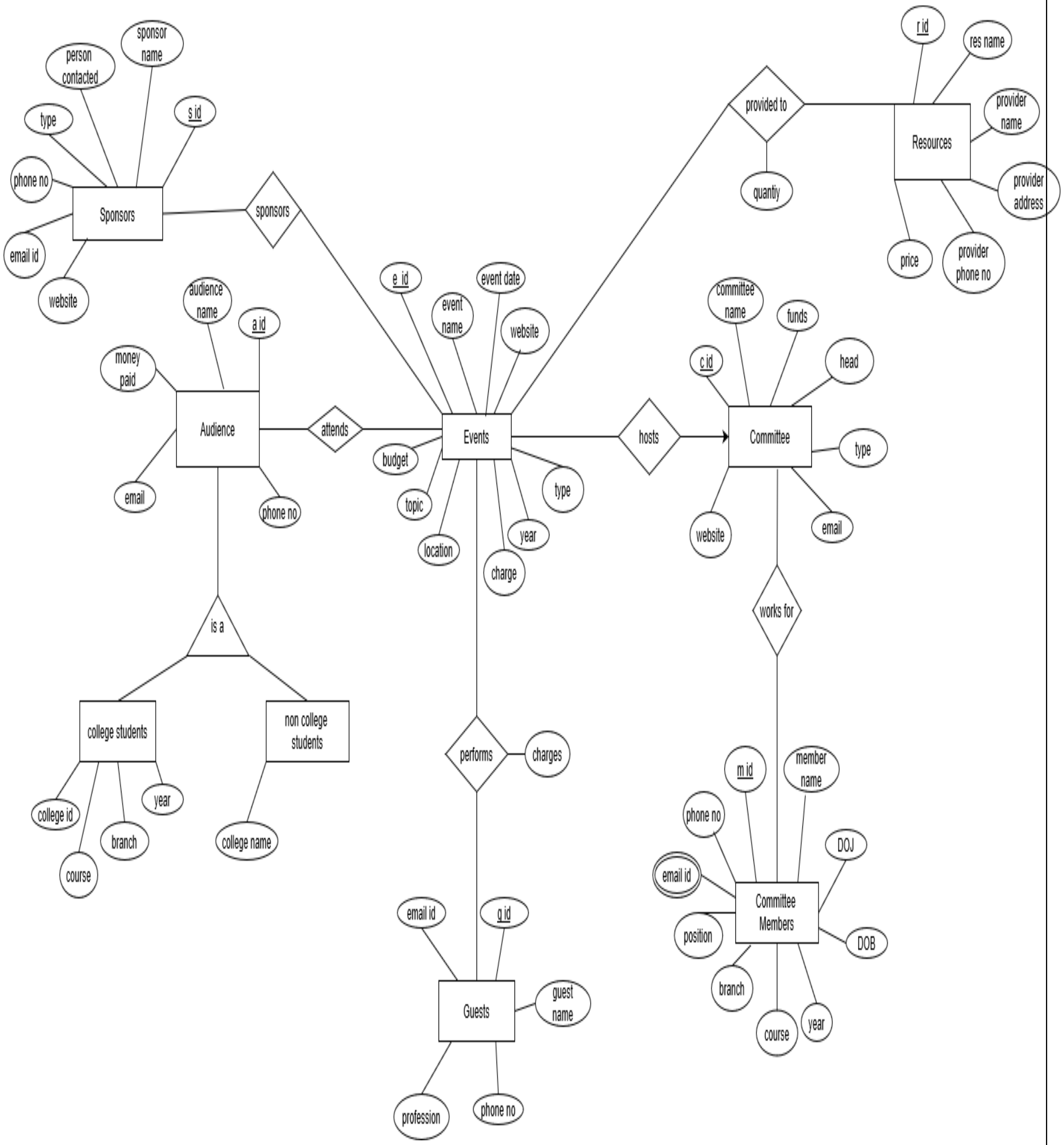


Figure 1: ER Diagram

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

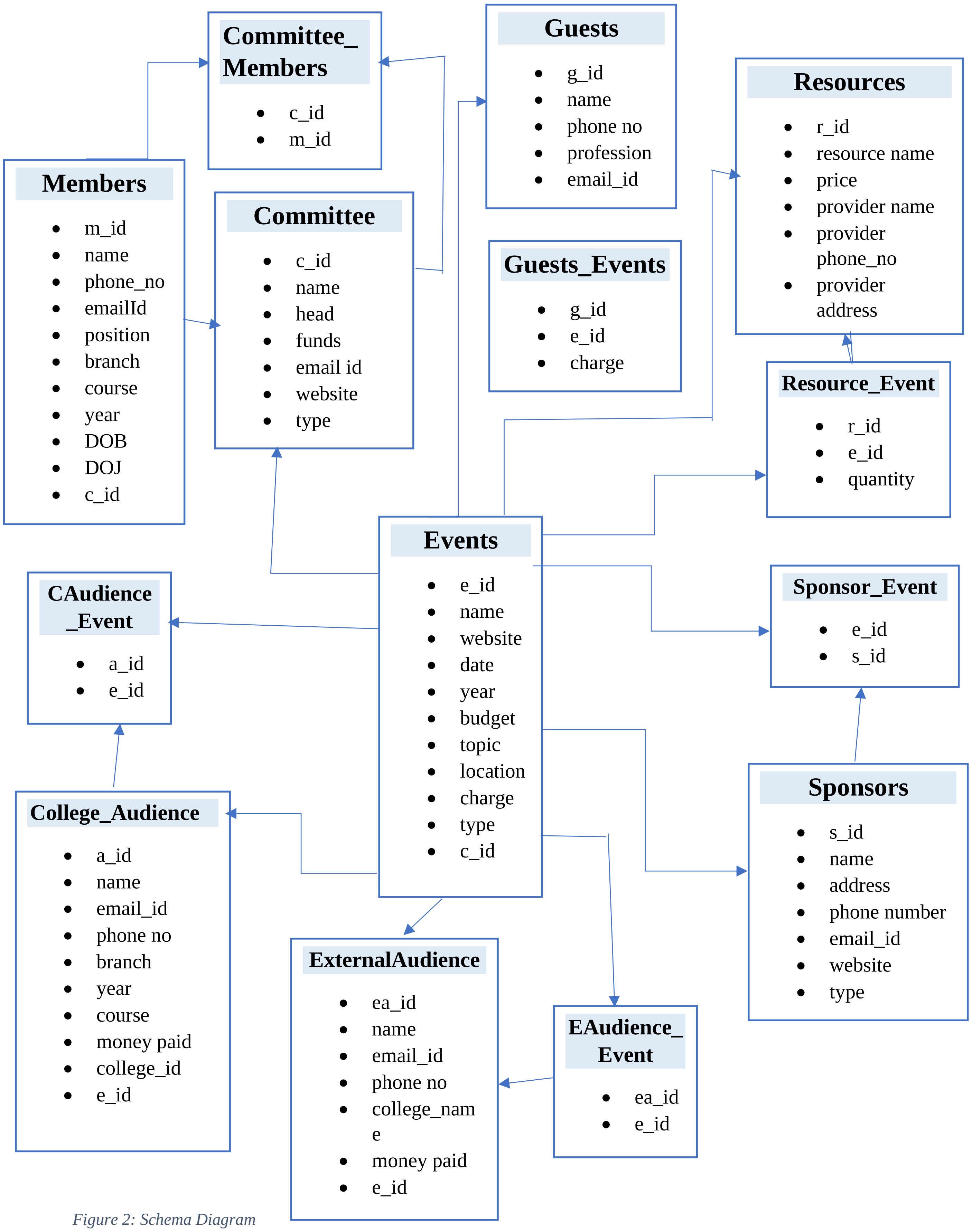


Figure 2: Schema Diagram

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 ($X \rightarrow Y$), 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 tables in 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 5: 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.
2. **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.
3. **Libraries**
   Python has a rich variety of libraries. Python downloads with an extensive library and it contain 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.
4. **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 designed specifically for relational databases. Therefore, RDBMSes are a subset of DBMSes.
A relational database refers to a database that stores data in a structured format, using rows and columns. This makes it easy to locate and access specific values within the database. It is "relational" because the values within each table are related to each other. Tables may also be related to other tables. The relational structure makes it possible to run queries 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 object that represents the database.

Once you have a Connection, you can create a 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 6: 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 (RDBMS) contained in a C library. In contrast to many other database management systems, SQLite is not a client–server database engine. Rather, it is embedded into the end program. SQLite is ACID-compliant and implements most of the SQL standard, generally following PostgreSQL syntax. SQLite is a popular choice as embedded database software for local/client storage in application software such as web browsers. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems (such as mobile phones), among others. SQLite has bindings to many programming languages.

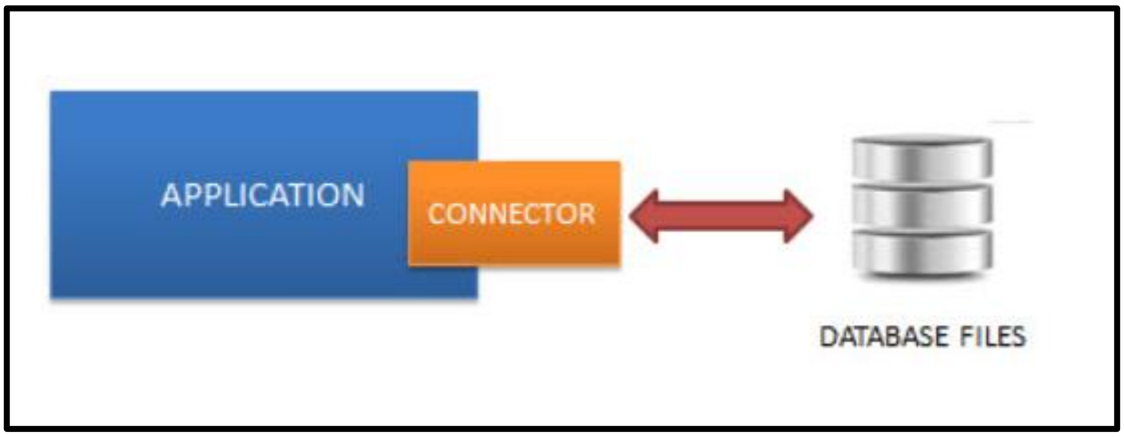


Figure 3: SQLite3 Software

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. SQLite browser uses a general spreadsheet-like interface, and there is no need to learn complicated 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 and inspect the results
- Examine log of all the SQL commands issued by the application

3.3.1. FLOW OF SCRIPTS

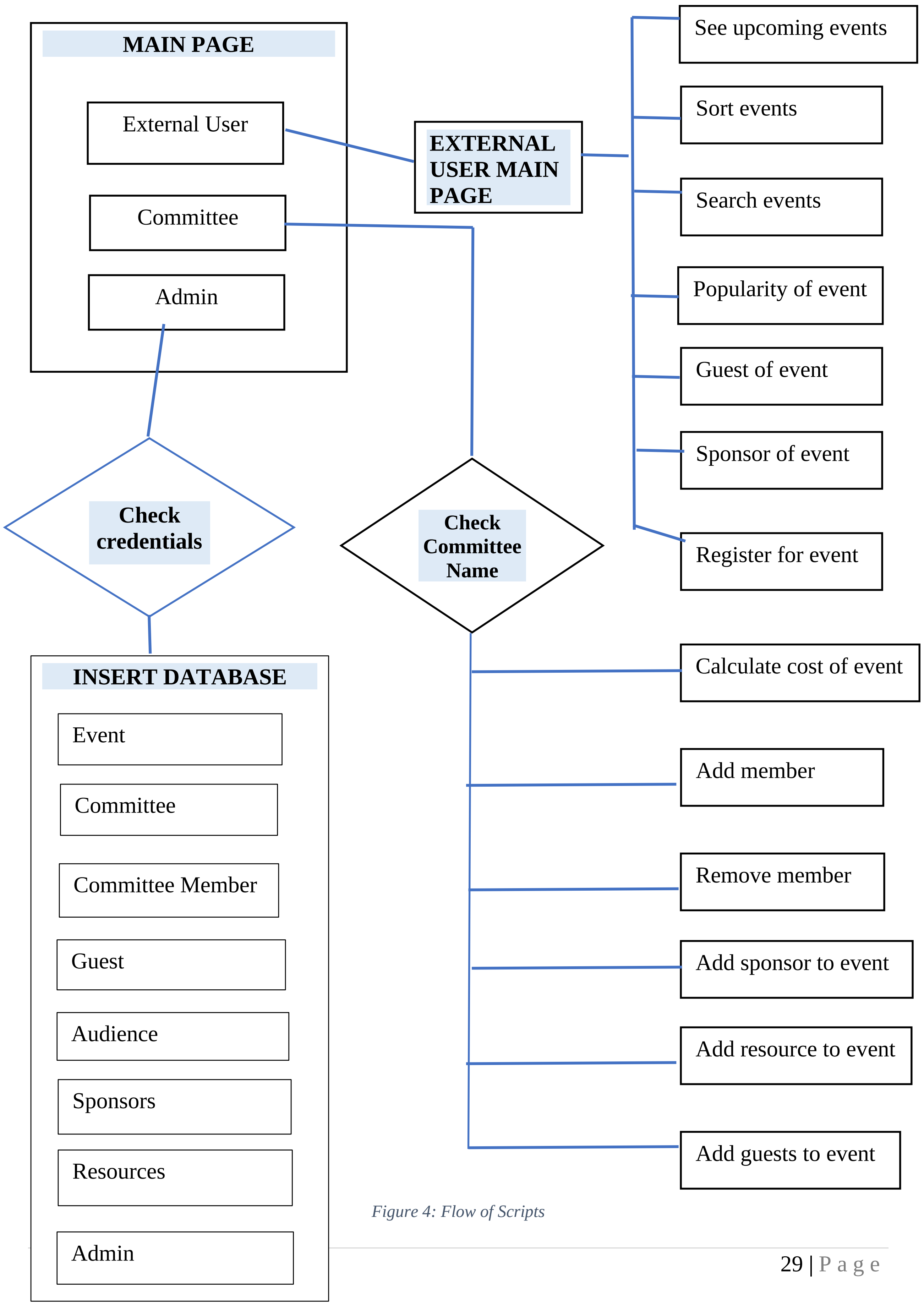
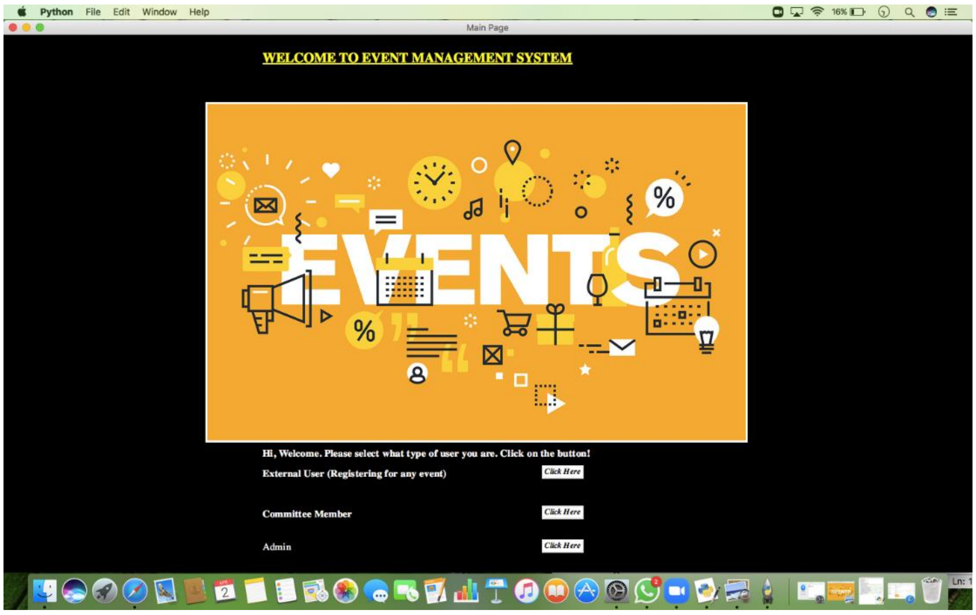


Figure 4: Flow of Scripts

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.



WELCOME TO EVENT MANAGEMENT SYSTEM

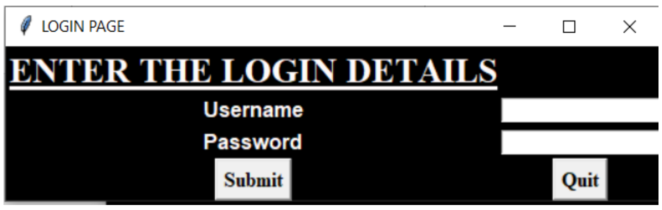
Hi, Welcome. Please select what type of user you are. Click on the button!

External User (Registering for any event)

Committee Member

Admin

2. **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.

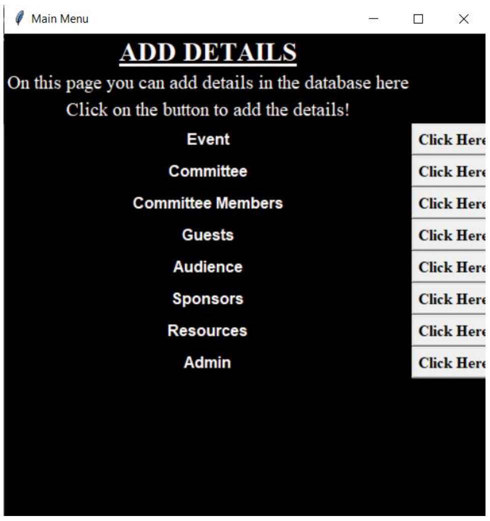


ENTER THE LOGIN DETAILS

Username

Password

3. **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.


Main Menu

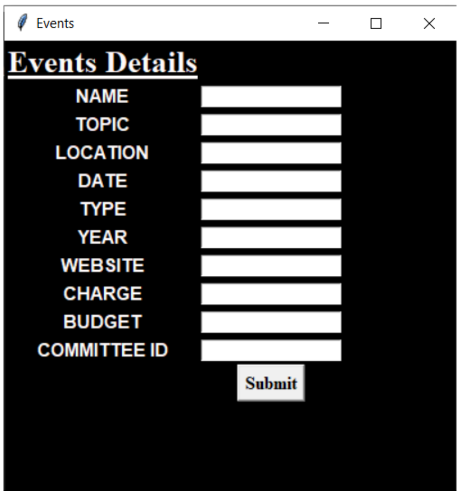
ADD DETAILS

On this page you can add details in the database here

Click on the button to add the details!

Event	Click Here
Committee	Click Here
Committee Members	Click Here
Guests	Click Here
Audience	Click Here
Sponsors	Click Here
Resources	Click Here
Admin	Click Here

4. **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:

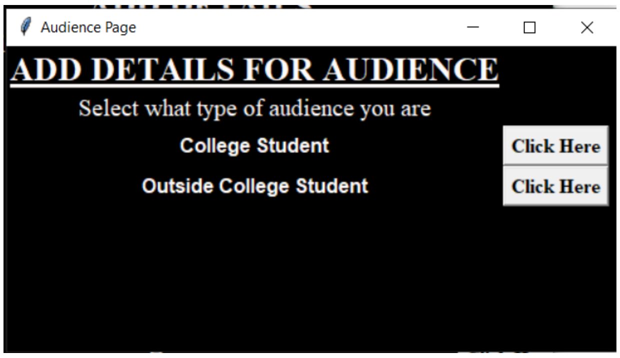

Events

Events Details

NAME
TOPIC
LOCATION
DATE
TYPE
YEAR
WEBSITE
CHARGE
BUDGET
COMMITTEE ID

Submit

5. **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.

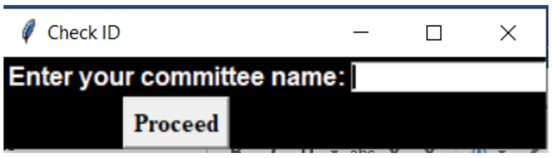

Audience Page

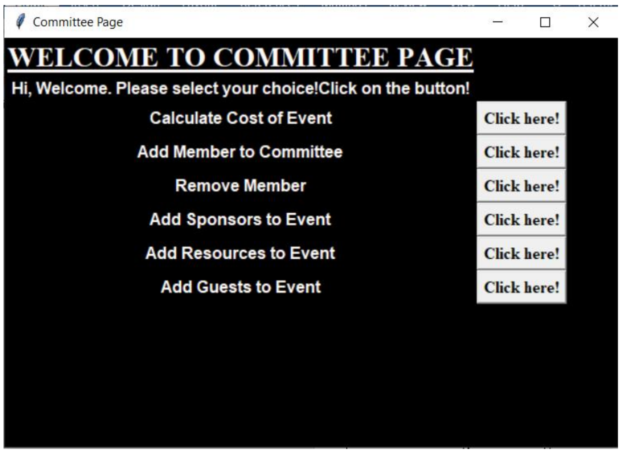
ADD DETAILS FOR AUDIENCE

Select what type of audience you are

College Student	Click Here
Outside College Student	Click Here

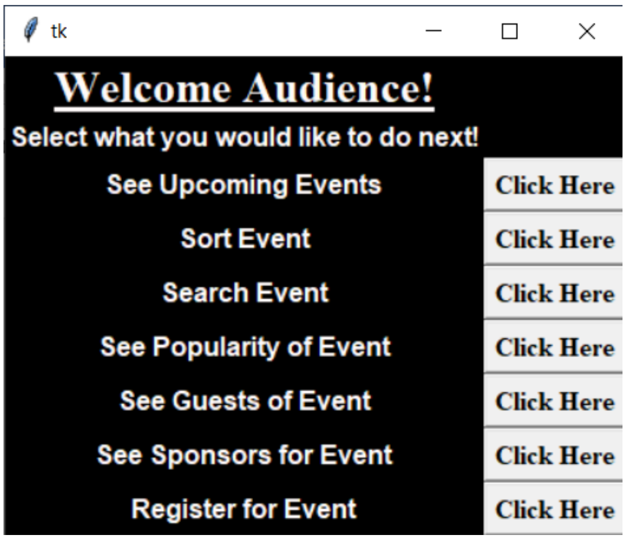
6. **Committee Main Page**
   This is used to connect all the functionalities of the committee member. From this page we can go to the individual pages. Before that, we ask for the committee name and then check from the database if the committee exists. If yes, further access is allowed and in the further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





7. **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)
8. **Add sponsors to event/add resources to event/add guests to event.**
   These pages are used to add sponsors, resources and guests to events.

9. **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.


tk

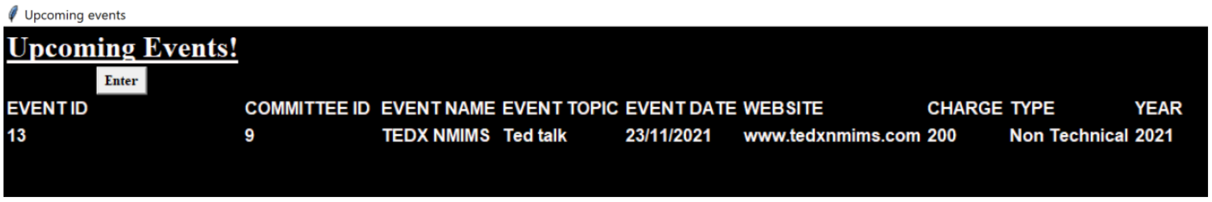
Welcome Audience!

Select what you would like to do next!

See Upcoming Events	Click Here
Sort Event	Click Here
Search Event	Click Here
See Popularity of Event	Click Here
See Guests of Event	Click Here
See Sponsors for Event	Click Here
Register for Event	Click Here

10.See upcoming events

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


Upcoming events

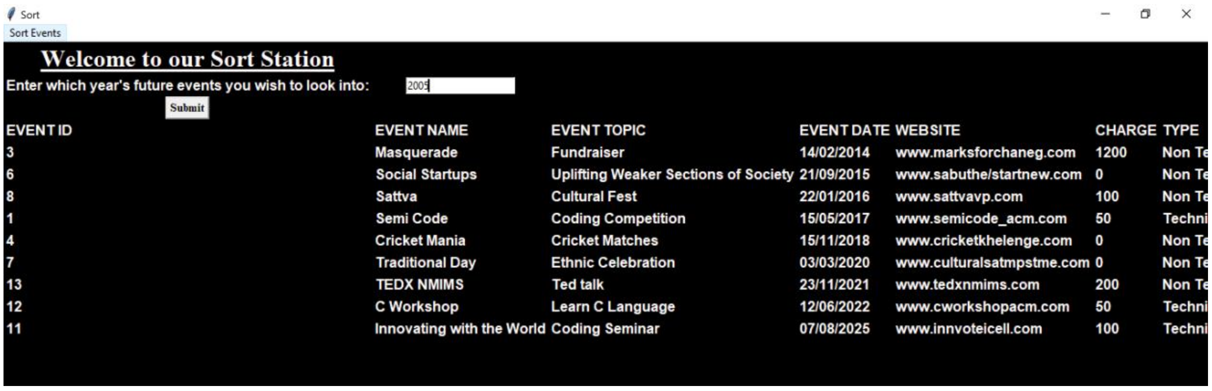
Upcoming Events!

Enter

EVENT ID	COMMITTEE ID	EVENT NAME	EVENT TOPIC	EVENT DATE	WEBSITE	CHARGE	TYPE	YEAR
13	9	TEDX NMIMS	Ted talk	23/11/2021	www.tedxnmims.com	200	Non Technical	2021

11.Sort events

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


Sort

Sort Events

Welcome to our Sort Station

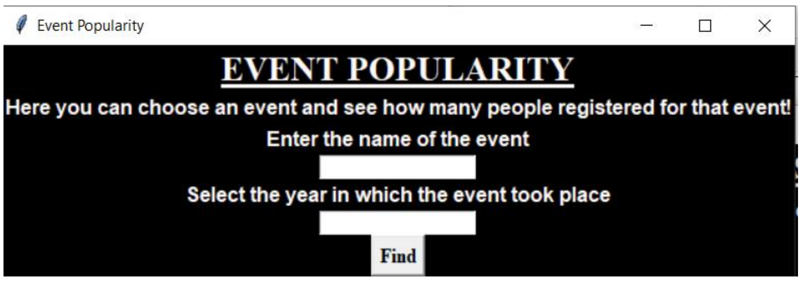
Enter which year's future events you wish to look into: 2005

Submit

EVENT ID	EVENT NAME	EVENT TOPIC	EVENT DATE	WEBSITE	CHARGE	TYPE
3	Masquerade	Fundraiser	14/02/2014	www.marksforchaneg.com	1200	Non Te
6	Social Startups	Uplifting Weaker Sections of Society	21/09/2015	www.sabuthe/startnew.com	0	Non Te
8	Sattva	Cultural Fest	22/01/2016	www.sattvavp.com	100	Non Te
1	Semi Code	Coding Competition	15/05/2017	www.semicode_acm.com	50	Techni
4	Cricket Mania	Cricket Matches	15/11/2018	www.cricketkhelenge.com	0	Non Te
7	Traditional Day	Ethnic Celebration	03/03/2020	www.culturalsatmpstme.com	0	Non Te
13	TEDX NMIMS	Ted talk	23/11/2021	www.tedxnmims.com	200	Non Te
12	C Workshop	Learn C Language	12/06/2022	www.cworkshopacm.com	50	Techni
11	Innovating with the World	Coding Seminar	07/08/2025	www.innvoteicell.com	100	Techni

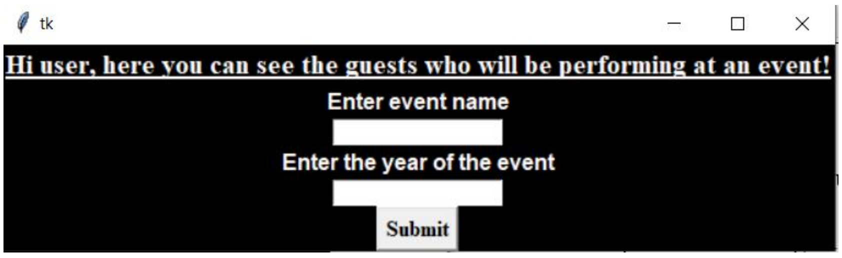
12.Search Events

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


Event Popularity
EVENT POPULARITY
Here you can choose an event and see how many people registered for that event!
Enter the name of the event
Select the year in which the event took place
Find

13.See guests of an event

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


tk
Hi user, here you can see the guests who will be performing at an event!
Enter event name
Enter the year of the event
Submit

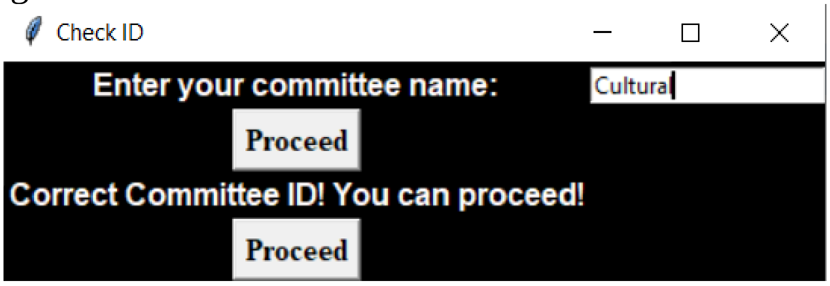
14.See sponsors of an event

Shows the events sponsored by a specific sponsors.

3.4 SCREENSHOTS

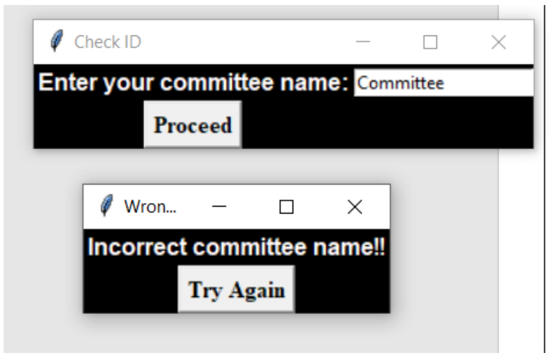
Checking committee credentials

Here we first check if the committee is present in the database
Upon entering correct committee name:



```
Check ID
Enter your committee name: Cultural
Proceed
Correct Committee ID! You can proceed!
Proceed
```

Upon entering incorrect committee name

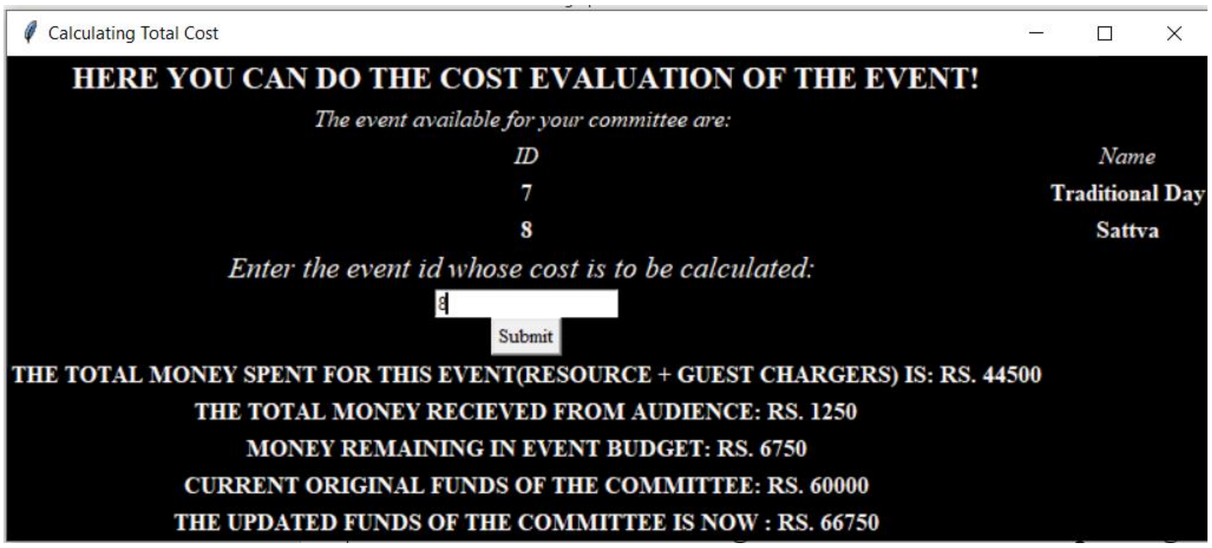


```
Check ID
Enter your committee name: Committee
Proceed

Wron...
Incorrect committee name!!
Try Again
```

Calculating total cost of event and updating

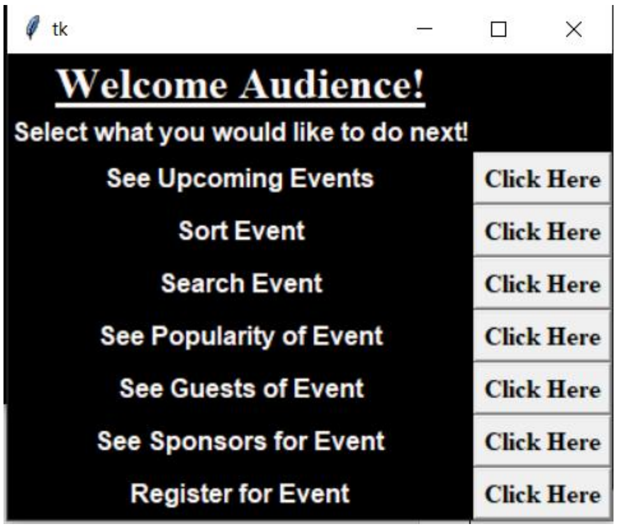
Here we first display the id and name of the events of the committee, then we analyse the costing.



```
Calculating Total Cost
HERE YOU CAN DO THE COST EVALUATION OF THE EVENT!
The event available for your committee are:
ID                                    Name
7                                     Traditional Day
8                                     Sattva
Enter the event id whose cost is to be calculated:
8
Submit
THE TOTAL MONEY SPENT FOR THIS EVENT(RESOURCE + GUEST CHARGERS) IS: RS. 44500
THE TOTAL MONEY RECIEVED FROM AUDIENCE: RS. 1250
MONEY REMAINING IN EVENT BUDGET: RS. 6750
CURRENT ORIGINAL FUNDS OF THE COMMITTEE: RS. 60000
THE UPDATED FUNDS OF THE COMMITTEE IS NOW : RS. 66750
```

Registering audience

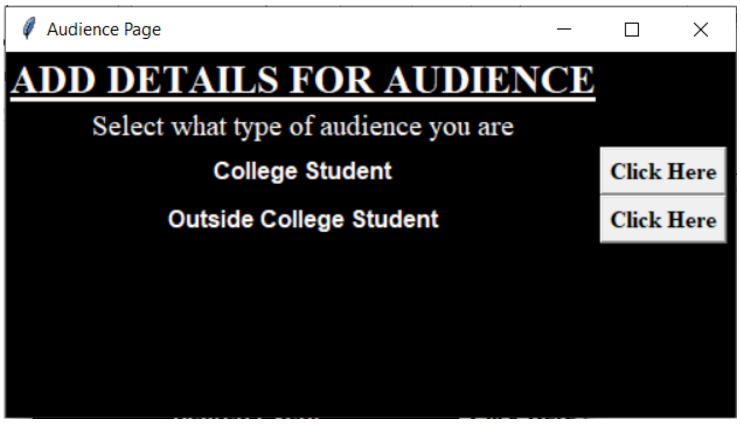
Steps for registering for an event


tk

Welcome Audience!

Select what you would like to do next!

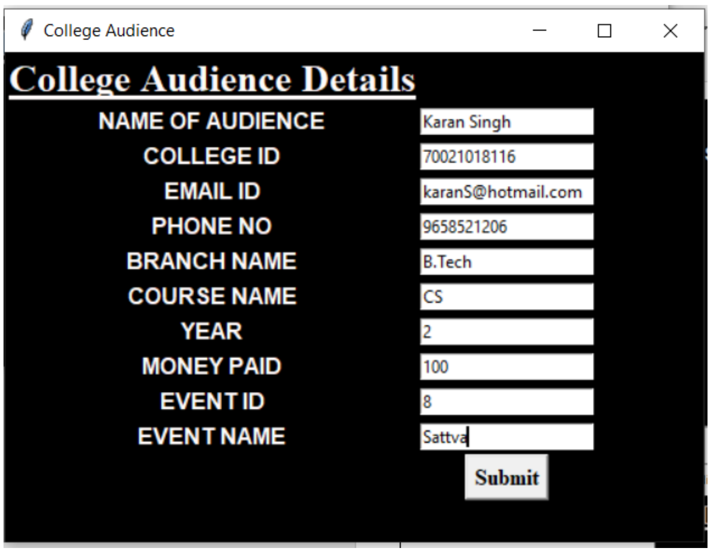
See Upcoming Events	Click Here
Sort Event	Click Here
Search Event	Click Here
See Popularity of Event	Click Here
See Guests of Event	Click Here
See Sponsors for Event	Click Here
Register for Event	Click Here


Audience Page

ADD DETAILS FOR AUDIENCE

Select what type of audience you are

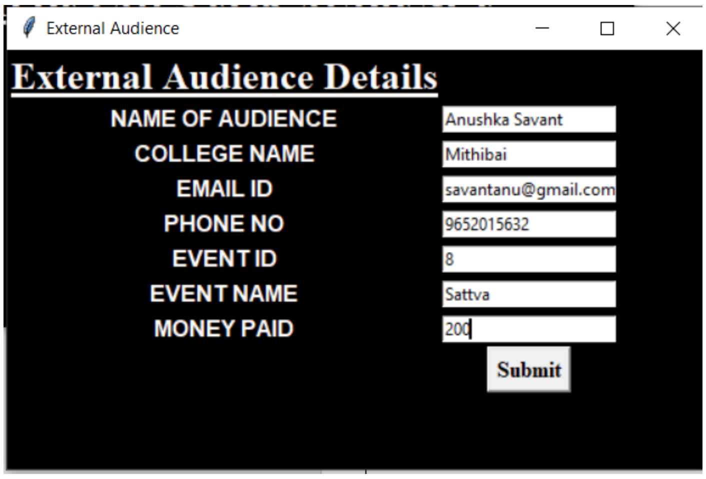
College Student	Click Here
Outside College Student	Click Here


College Audience

College Audience Details

NAME OF AUDIENCE	Karan Singh
COLLEGE ID	70021018116
EMAIL ID	karanS@hotmail.com
PHONE NO	9658521206
BRANCH NAME	B.Tech
COURSE NAME	CS
YEAR	2
MONEY PAID	100
EVENT ID	8
EVENT NAME	Sattva

Submit

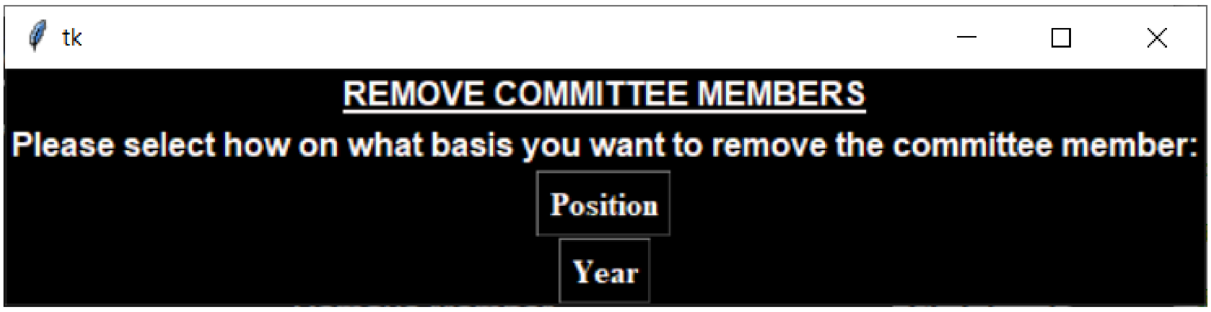

External Audience

External Audience Details

NAME OF AUDIENCE	Anushka Savant
COLLEGE NAME	Mithibai
EMAIL ID	savantanu@gmail.com
PHONE NO	9652015632
EVENT ID	8
EVENT NAME	Sattva
MONEY PAID	200

Submit

Remove members

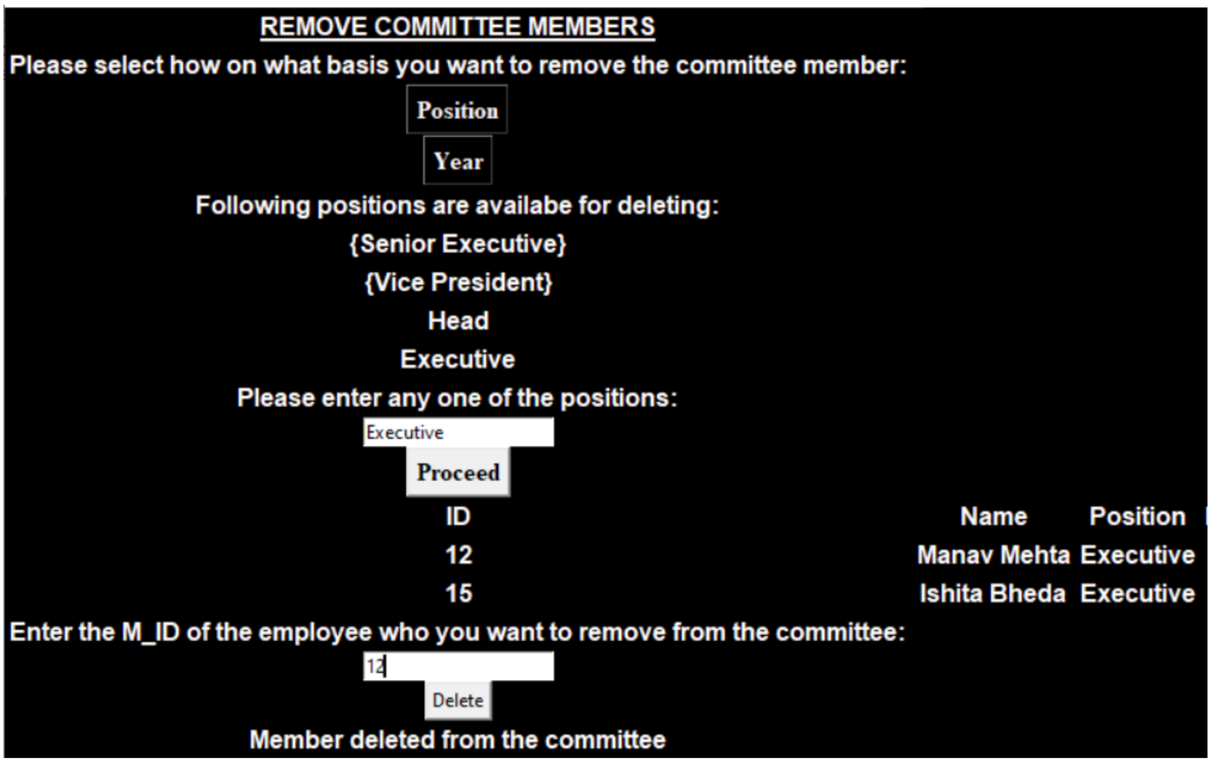

tk

REMOVE COMMITTEE MEMBERS

Please select how on what basis you want to remove the committee member:

Position

Year


REMOVE COMMITTEE MEMBERS

Please select how on what basis you want to remove the committee member:

Position

Year

Following positions are availabe for deleting:

{Senior Executive}

{Vice President}

Head

Executive

Please enter any one of the positions:

Executive

Proceed

ID	Name	Position
12	Manav Mehta	Executive
15	Ishita Bheda	Executive

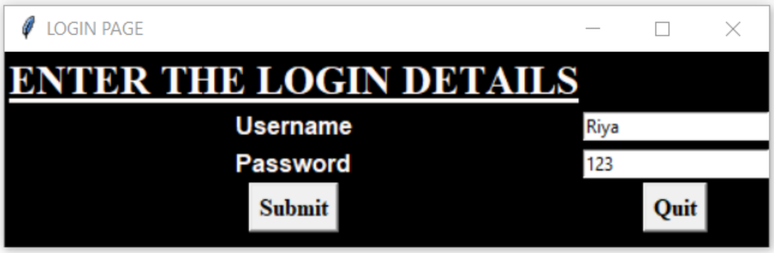
Enter the M_ID of the employee who you want to remove from the committee:

12

Delete

Member deleted from the committee

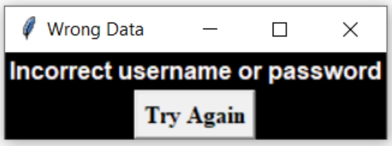
Checking login for admin


LOGIN PAGE

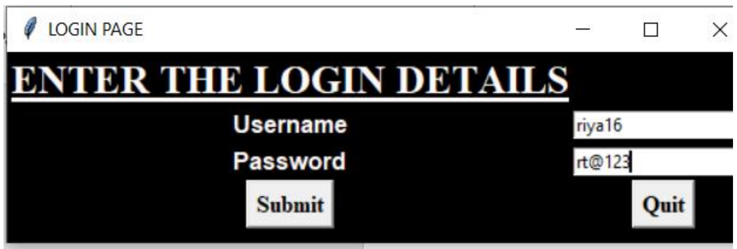
ENTER THE LOGIN DETAILS

Username: Riya

Password: 123


Wrong Data

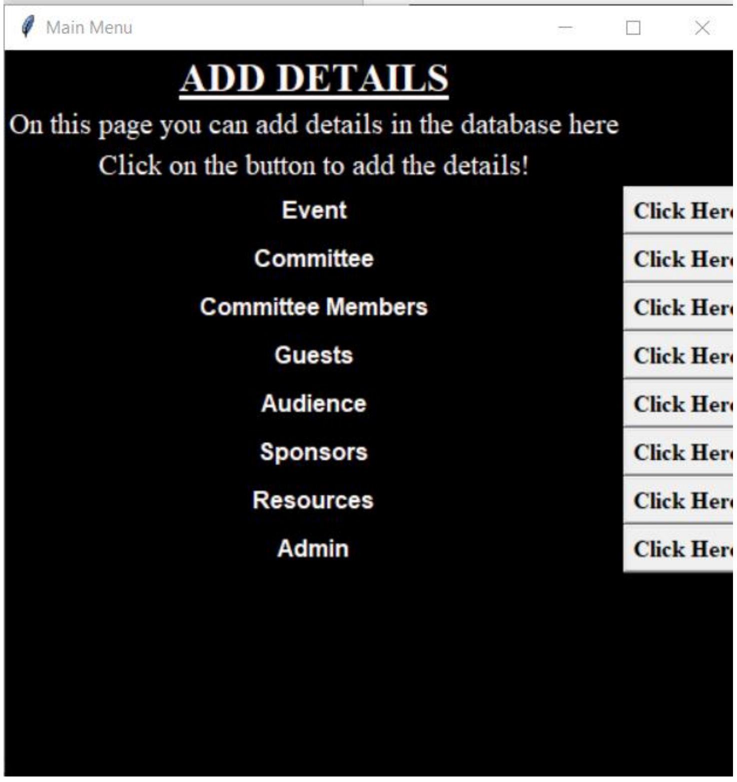
Incorrect username or password


LOGIN PAGE

ENTER THE LOGIN DETAILS

Username: riya16

Password: rt@123


Main Menu

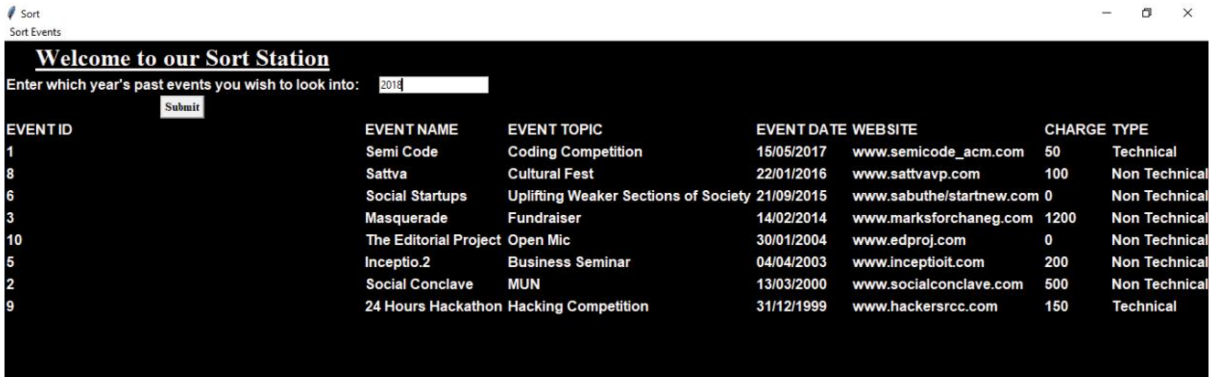
ADD DETAILS

On this page you can add details in the database here

Click on the button to add the details!

Event	Click Her
Committee	Click Her
Committee Members	Click Her
Guests	Click Her
Audience	Click Her
Sponsors	Click Her
Resources	Click Her
Admin	Click Her

Sort event



Welcome to our Sort Station

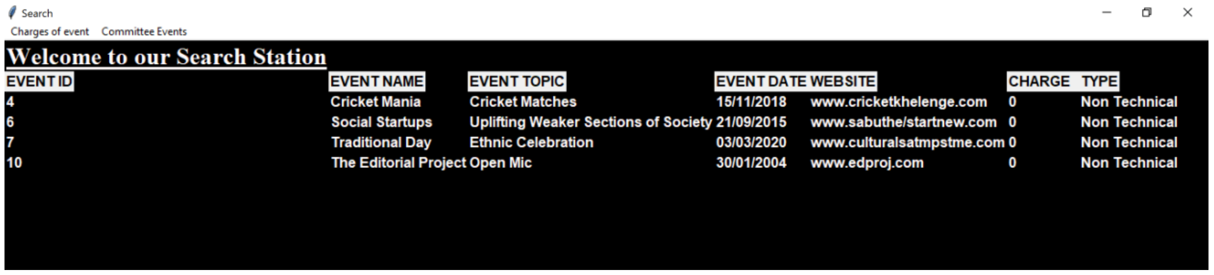
Enter which year's past events you wish to look into: 2018

Submit

EVENT ID	EVENT NAME	EVENT TOPIC	EVENT DATE	WEBSITE	CHARGE	TYPE
1	Semi Code	Coding Competition	15/05/2017	www.semicode_acm.com	50	Technical
8	Sattva	Cultural Fest	22/01/2016	www.sattvavp.com	100	Non Technical
6	Social Startups	Uplifting Weaker Sections of Society	21/09/2015	www.sabuthe/startnew.com	0	Non Technical
3	Masquerade	Fundraiser	14/02/2014	www.marksforchaneg.com	1200	Non Technical
10	The Editorial Project	Open Mic	30/01/2004	www.edproj.com	0	Non Technical
5	Inceptio.2	Business Seminar	04/04/2003	www.inceptioit.com	200	Non Technical
2	Social Conclave	MUN	13/03/2000	www.socialconclave.com	500	Non Technical
9	24 Hours Hackathon	Hacking Competition	31/12/1999	www.hackersrcc.com	150	Technical

Search event

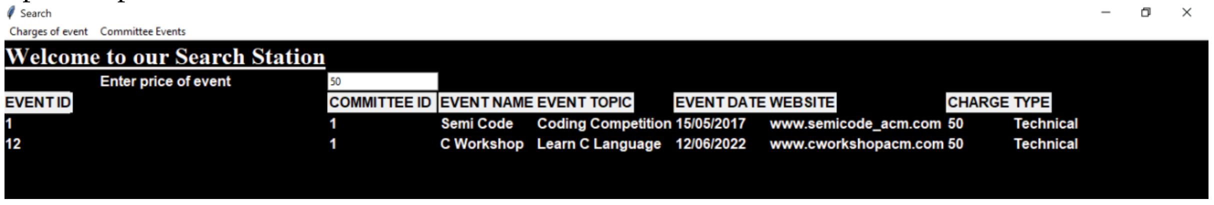
Free events:



Welcome to our Search Station

EVENT ID	EVENT NAME	EVENT TOPIC	EVENT DATE	WEBSITE	CHARGE	TYPE
4	Cricket Mania	Cricket Matches	15/11/2018	www.cricketkhelenge.com	0	Non Technical
6	Social Startups	Uplifting Weaker Sections of Society	21/09/2015	www.sabuthe/startnew.com	0	Non Technical
7	Traditional Day	Ethnic Celebration	03/03/2020	www.culturalsatmpstme.com	0	Non Technical
10	The Editorial Project	Open Mic	30/01/2004	www.edproj.com	0	Non Technical

Specific price:

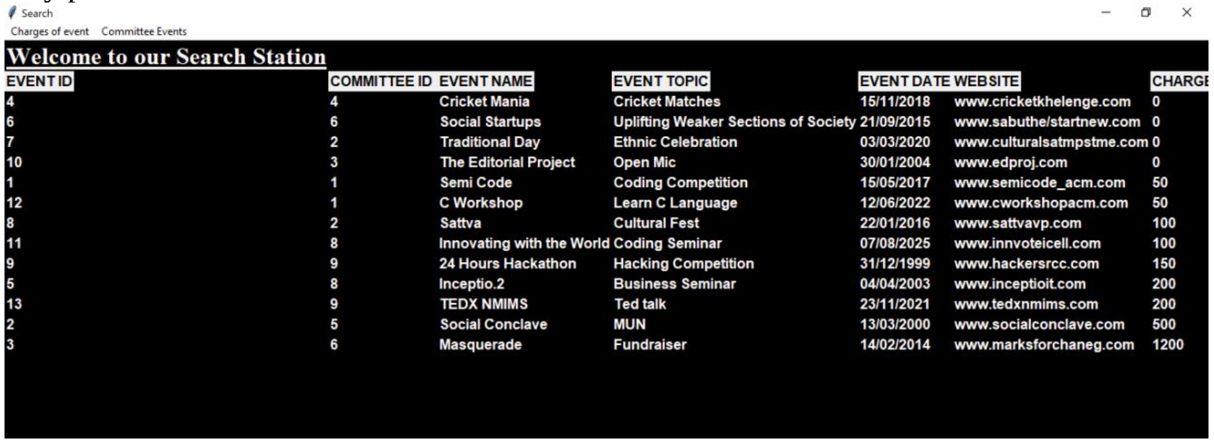


Welcome to our Search Station

Enter price of event 50

EVENT ID	COMMITTEE ID	EVENT NAME	EVENT TOPIC	EVENT DATE	WEBSITE	CHARGE	TYPE
1	1	Semi Code	Coding Competition	15/05/2017	www.semicode_acm.com	50	Technical
12	1	C Workshop	Learn C Language	12/06/2022	www.cworkshopacm.com	50	Technical

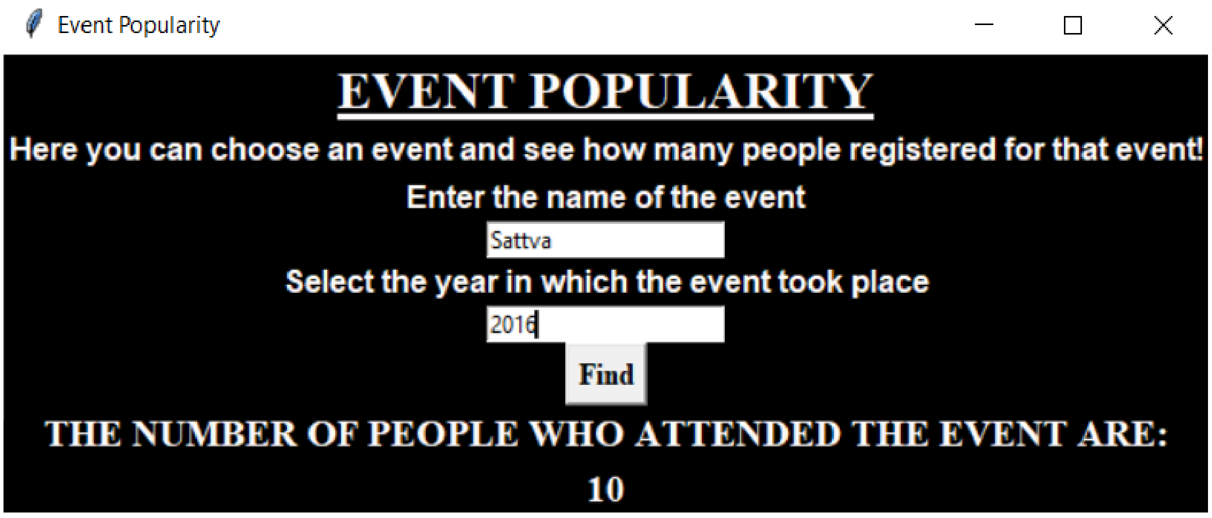
Any price:



Welcome to our Search Station

EVENT ID	COMMITTEE ID	EVENT NAME	EVENT TOPIC	EVENT DATE	WEBSITE	CHARGE
4	4	Cricket Mania	Cricket Matches	15/11/2018	www.cricketkhelenge.com	0
6	6	Social Startups	Uplifting Weaker Sections of Society	21/09/2015	www.sabuthe/startnew.com	0
7	2	Traditional Day	Ethnic Celebration	03/03/2020	www.culturalsatmpstme.com	0
10	3	The Editorial Project	Open Mic	30/01/2004	www.edproj.com	0
1	1	Semi Code	Coding Competition	15/05/2017	www.semicode_acm.com	50
12	1	C Workshop	Learn C Language	12/06/2022	www.cworkshopacm.com	50
8	2	Sattva	Cultural Fest	22/01/2016	www.sattvavp.com	100
11	8	Innovating with the World	Coding Seminar	07/08/2025	www.innvoteicell.com	100
9	9	24 Hours Hackathon	Hacking Competition	31/12/1999	www.hackersrcc.com	150
5	8	Inceptio.2	Business Seminar	04/04/2003	www.inceptioit.com	200
13	9	TEDX NMIMS	Ted talk	23/11/2021	www.tedxnmims.com	200
2	5	Social Conclave	MUN	13/03/2000	www.socialconclave.com	500
3	6	Masquerade	Fundraiser	14/02/2014	www.marksforchaneg.com	1200

See popularity of events


Event Popularity

EVENT POPULARITY

Here you can choose an event and see how many people registered for that event!

Enter the name of the event

Sattva

Select the year in which the event took place

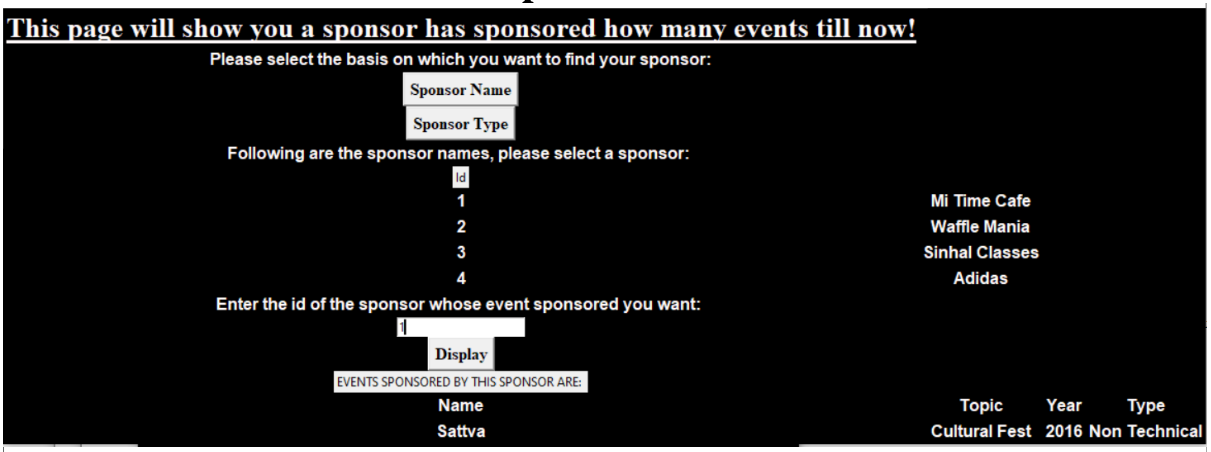
2016

Find

THE NUMBER OF PEOPLE WHO ATTENDED THE EVENT ARE:

10

See sponsor details


This page will show you a sponsor has sponsored how many events till now!

Please select the basis on which you want to find your sponsor:

Sponsor Name

Sponsor Type

Following are the sponsor names, please select a sponsor:

Id	
1	Mi Time Cafe
2	Waffle Mania
3	Sinhal Classes
4	Adidas

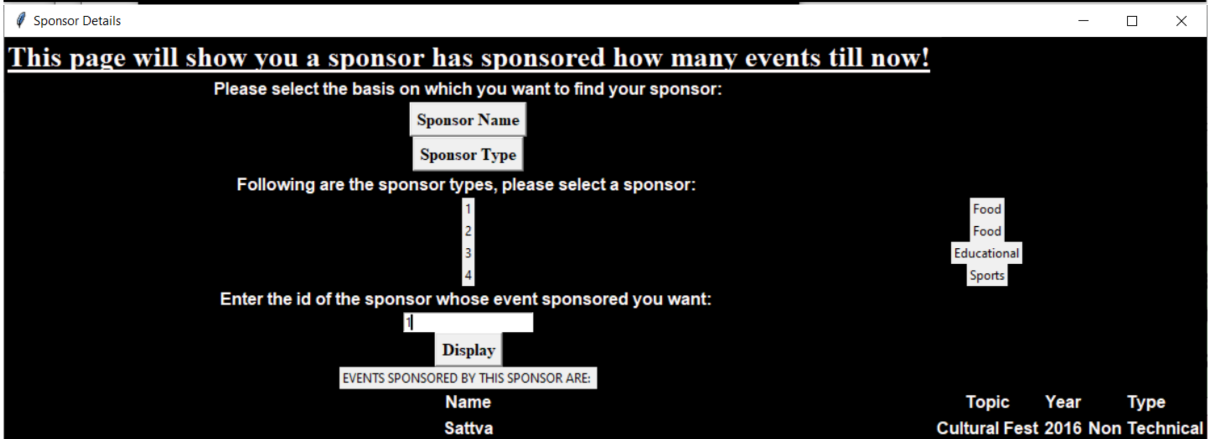
Enter the id of the sponsor whose event sponsored you want:

1

Display

EVENTS SPONSORED BY THIS SPONSOR ARE:

Name	Topic	Year	Type
Sattva	Cultural Fest	2016	Non Technical


Sponsor Details

This page will show you a sponsor has sponsored how many events till now!

Please select the basis on which you want to find your sponsor:

Sponsor Name

Sponsor Type

Following are the sponsor types, please select a sponsor:

1	Food
2	Food
3	Educational
4	Sports

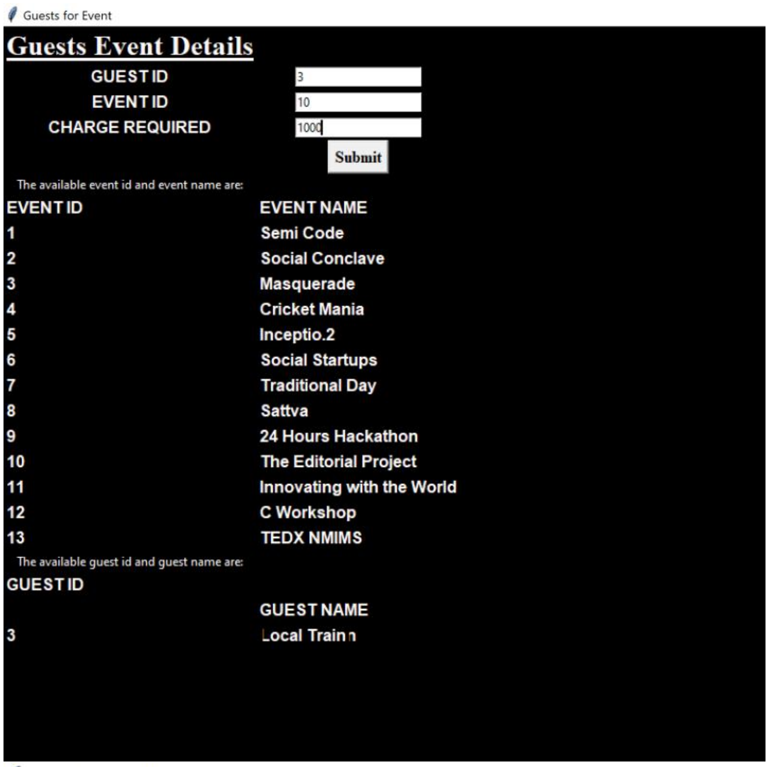
Enter the id of the sponsor whose event sponsored you want:

1

Display

EVENTS SPONSORED BY THIS SPONSOR ARE:

Name	Topic	Year	Type
Sattva	Cultural Fest	2016	Non Technical



Guests for Event

Guests Event Details

GUEST ID	3
EVENT ID	10
CHARGE REQUIRED	1000

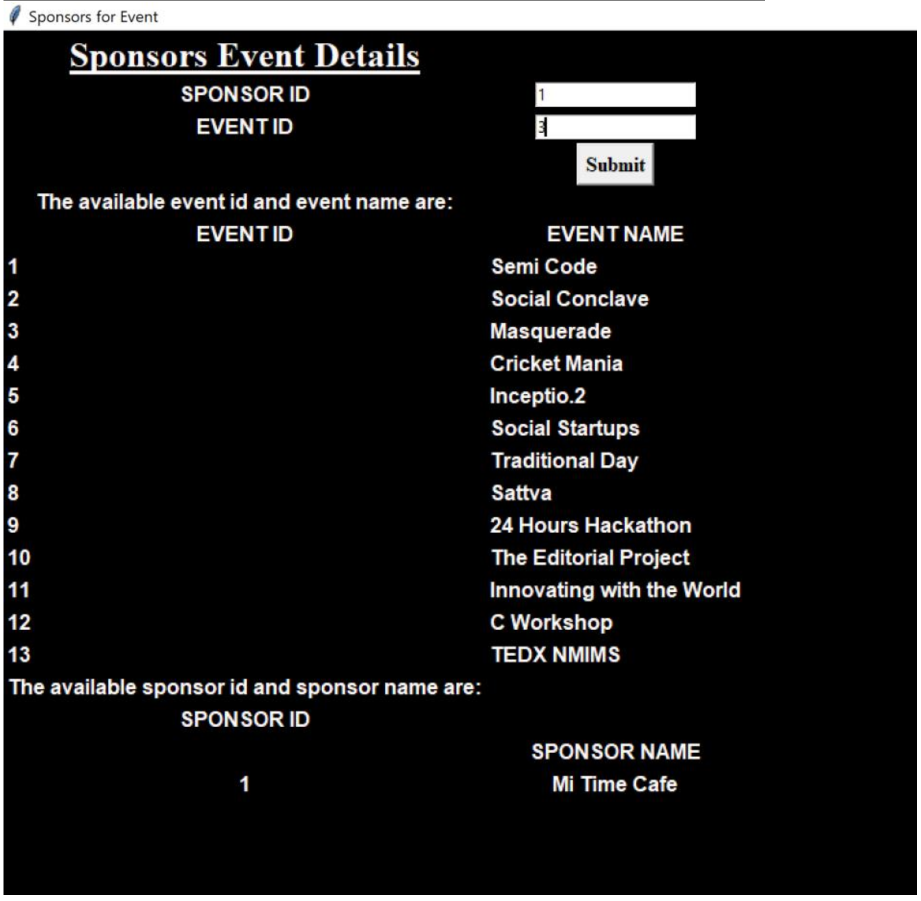
Submit

The available event id and event name are:

EVENT ID	EVENT NAME
1	Semi Code
2	Social Conclave
3	Masquerade
4	Cricket Mania
5	Inceptio.2
6	Social Startups
7	Traditional Day
8	Sattva
9	24 Hours Hackathon
10	The Editorial Project
11	Innovating with the World
12	C Workshop
13	TEDX NMIMS

The available guest id and guest name are:

GUEST ID	GUEST NAME
3	Local Train



Sponsors for Event

Sponsors Event Details

SPONSOR ID	1
EVENT ID	3

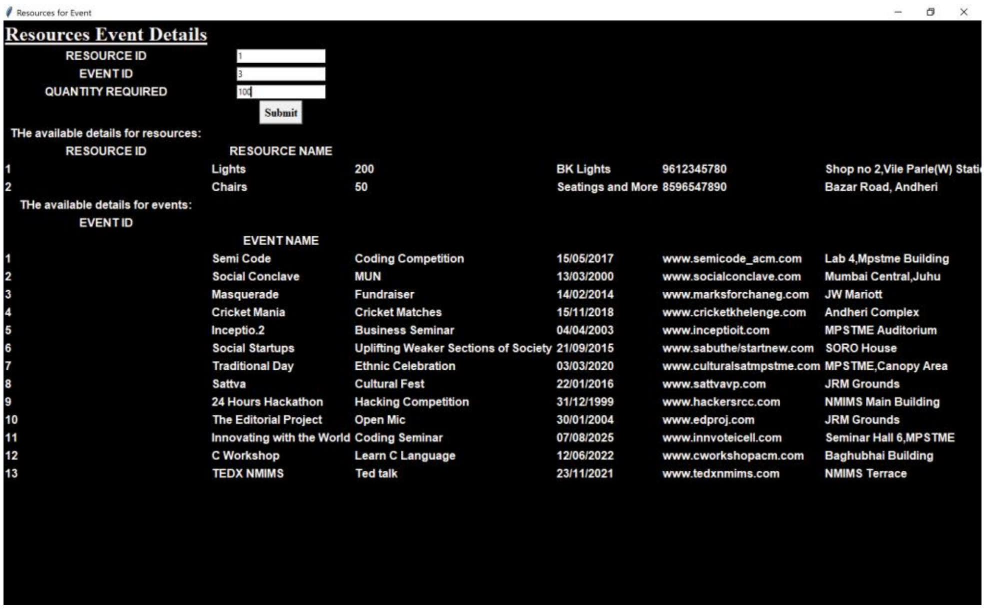
Submit

The available event id and event name are:

EVENT ID	EVENT NAME
1	Semi Code
2	Social Conclave
3	Masquerade
4	Cricket Mania
5	Inceptio.2
6	Social Startups
7	Traditional Day
8	Sattva
9	24 Hours Hackathon
10	The Editorial Project
11	Innovating with the World
12	C Workshop
13	TEDX NMIMS

The available sponsor id and sponsor name are:

SPONSOR ID	SPONSOR NAME
1	Mi Time Cafe



Resources for Event

Resources Event Details

RESOURCE ID 1
EVENT ID 3
QUANTITY REQUIRED 100

Submit

THe available details for resources:

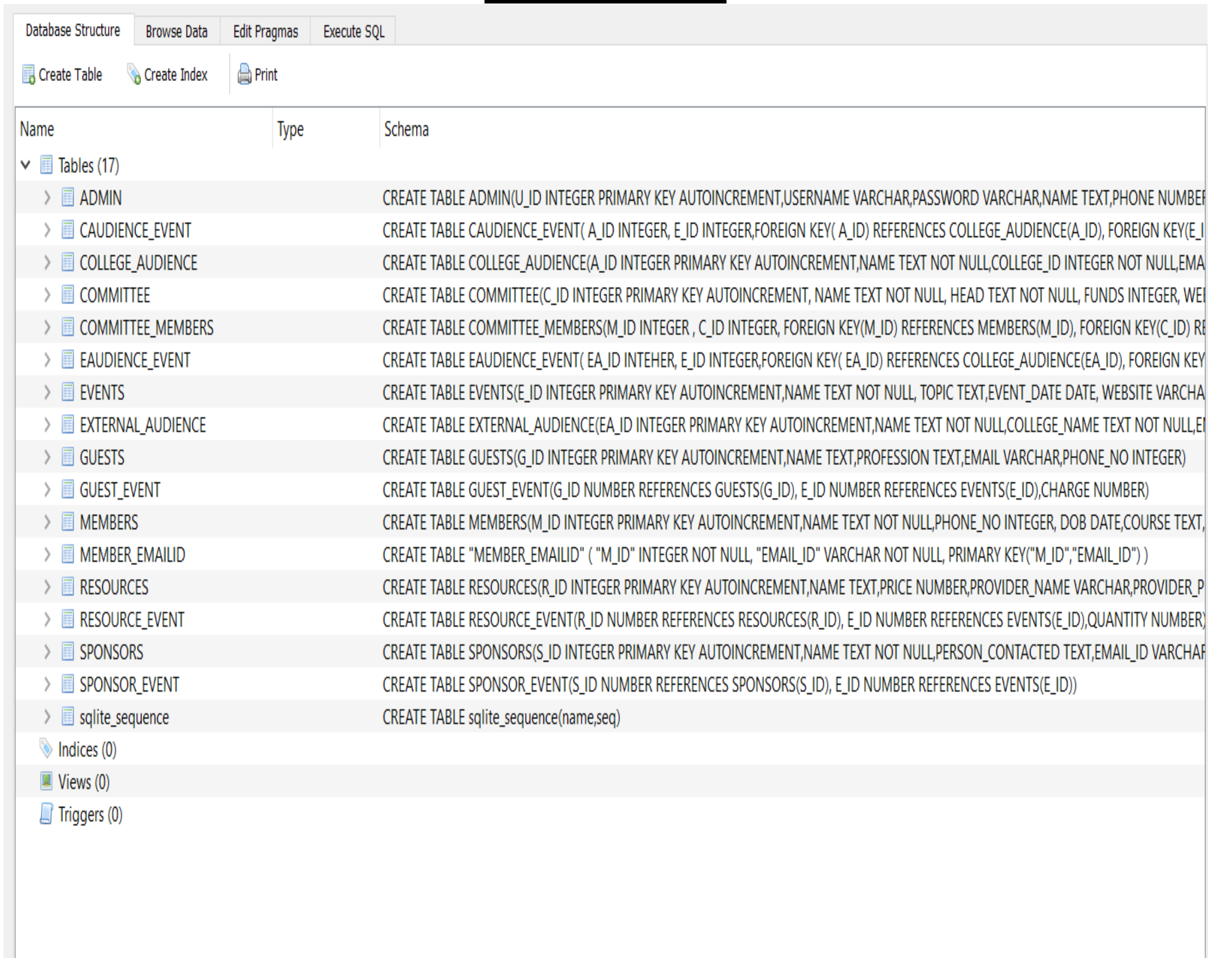
RESOURCE ID	RESOURCE NAME				
1	Lights	200	BK Lights	9612345780	Shop no 2,Vile Parle(W) Stati
2	Chairs	50	Seatings and More	8596547890	Bazar Road, Andheri

THe available details for events:

EVENT ID	EVENT NAME				
1	Semi Code	Coding Competition	15/05/2017	www.semicode_acm.com	Lab 4,Mpstme Building
2	Social Conclave	MUN	13/03/2000	www.socialconclave.com	Mumbai Central,Juhu
3	Masquerade	Fundraiser	14/02/2014	www.marksforchaneg.com	JW Mariott
4	Cricket Mania	Cricket Matches	15/11/2018	www.cricketkhelenge.com	Andheri Complex
5	Inceptio.2	Business Seminar	04/04/2003	www.inceptioit.com	MPSTME Auditorium
6	Social Startups	Uplifting Weaker Sections of Society	21/09/2015	www.sabuthe/startnew.com	SORO House
7	Traditional Day	Ethnic Celebration	03/03/2020	www.culturalsatmpstme.com	MPSTME,Canopy Area
8	Sattva	Cultural Fest	22/01/2016	www.sattvavp.com	JRM Grounds
9	24 Hours Hackathon	Hacking Competition	31/12/1999	www.hackersrcc.com	NMIMS Main Building
10	The Editorial Project	Open Mic	30/01/2004	www.edproj.com	JRM Grounds
11	Innovating with the World	Coding Seminar	07/08/2025	www.innvoteicell.com	Seminar Hall 6,MPSTME
12	C Workshop	Learn C Language	12/06/2022	www.cworkshopacm.com	Baghubhai Building
13	TEDX NMIMS	Ted talk	23/11/2021	www.tedxnmims.com	NMIMS Terrace

3.4 DATABASE STRUCTURE

Main structure

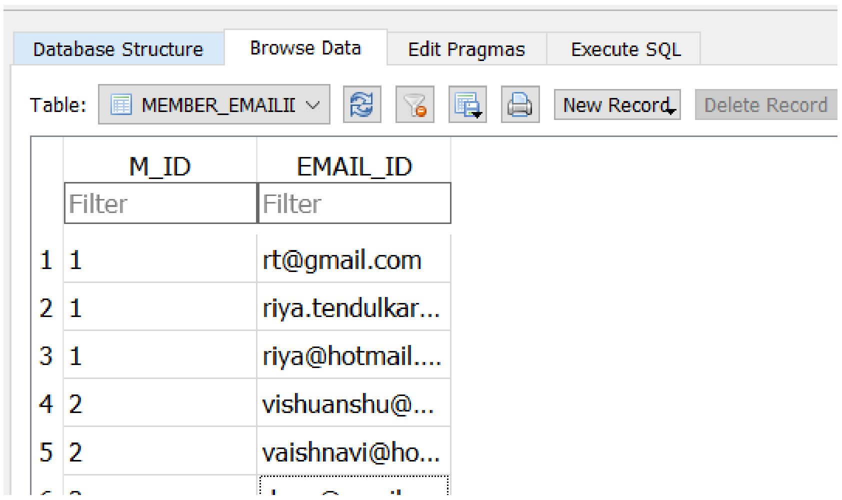


Database Structure | Browse Data | Edit Pragmas | Execute SQL

Create Table | Create Index | Print

Name	Type	Schema
Tables (17)		
ADMIN		CREATE TABLE ADMIN(U_ID INTEGER PRIMARY KEY AUTOINCREMENT,USERNAME VARCHAR,PASSWORD VARCHAR,NAME TEXT,PHONE NUMBE
CAUDIENCE_EVENT		CREATE TABLE CAUDIENCE_EVENT(A_ID INTEGER, E_ID INTEGER,FOREIGN KEY(A_ID) REFERENCES COLLEGE_AUDIENCE(A_ID), FOREIGN KEY(E_
COLLEGE_AUDIENCE		CREATE TABLE COLLEGE_AUDIENCE(A_ID INTEGER PRIMARY KEY AUTOINCREMENT,NAME TEXT NOT NULL,COLLEGE_ID INTEGER NOT NULL,EMA
COMMITTEE		CREATE TABLE COMMITTEE(C_ID INTEGER PRIMARY KEY AUTOINCREMENT, NAME TEXT NOT NULL, HEAD TEXT NOT NULL, FUNDS INTEGER, WE
COMMITTEE_MEMBERS		CREATE TABLE COMMITTEE_MEMBERS(M_ID INTEGER , C_ID INTEGER, FOREIGN KEY(M_ID) REFERENCES MEMBERS(M_ID), FOREIGN KEY(C_ID) R
EAUDIENCE_EVENT		CREATE TABLE EAUDIENCE_EVENT(EA_ID INTEHER, E_ID INTEGER,FOREIGN KEY(EA_ID) REFERENCES COLLEGE_AUDIENCE(EA_ID), FOREIGN KEY
EVENTS		CREATE TABLE EVENTS(E_ID INTEGER PRIMARY KEY AUTOINCREMENT,NAME TEXT NOT NULL, TOPIC TEXT,EVENT_DATE DATE, WEBSITE VARCHA
EXTERNAL_AUDIENCE		CREATE TABLE EXTERNAL_AUDIENCE(EA_ID INTEGER PRIMARY KEY AUTOINCREMENT,NAME TEXT NOT NULL,COLLEGE_NAME TEXT NOT NULL,E
GUESTS		CREATE TABLE GUESTS(G_ID INTEGER PRIMARY KEY AUTOINCREMENT,NAME TEXT,PROFESSION TEXT,EMAIL VARCHAR,PHONE_NO INTEGER)
GUEST_EVENT		CREATE TABLE GUEST_EVENT(G_ID NUMBER REFERENCES GUESTS(G_ID), E_ID NUMBER REFERENCES EVENTS(E_ID),CHARGE NUMBER)
MEMBERS		CREATE TABLE MEMBERS(M_ID INTEGER PRIMARY KEY AUTOINCREMENT,NAME TEXT NOT NULL,PHONE_NO INTEGER, DOB DATE,COURSE TEXT,
MEMBER_EMAILID		CREATE TABLE "MEMBER_EMAILID" ("M_ID" INTEGER NOT NULL, "EMAIL_ID" VARCHAR NOT NULL, PRIMARY KEY("M_ID","EMAIL_ID"))
RESOURCES		CREATE TABLE RESOURCES(R_ID INTEGER PRIMARY KEY AUTOINCREMENT,NAME TEXT,PRICE NUMBER,PROVIDER_NAME VARCHAR,PROVIDER_P
RESOURCE_EVENT		CREATE TABLE RESOURCE_EVENT(R_ID NUMBER REFERENCES RESOURCES(R_ID), E_ID NUMBER REFERENCES EVENTS(E_ID),QUANTITY NUMBER)
SPONSORS		CREATE TABLE SPONSORS(S_ID INTEGER PRIMARY KEY AUTOINCREMENT,NAME TEXT NOT NULL,PERSON_CONTACTED TEXT,EMAIL_ID VARCHA
SPONSOR_EVENT		CREATE TABLE SPONSOR_EVENT(S_ID NUMBER REFERENCES SPONSORS(S_ID), E_ID NUMBER REFERENCES EVENTS(E_ID))
sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)
Indices (0)		
Views (0)		
Triggers (0)		

MEMBER_EMAILID

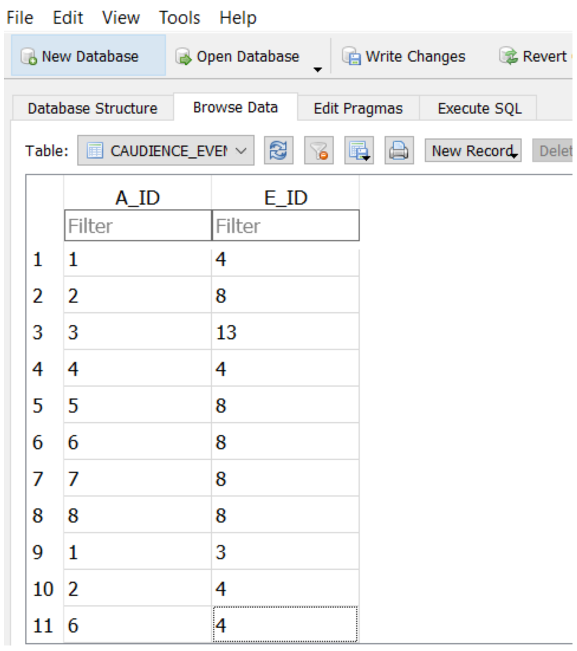


Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: MEMBER_EMAILII | New Record | Delete Record

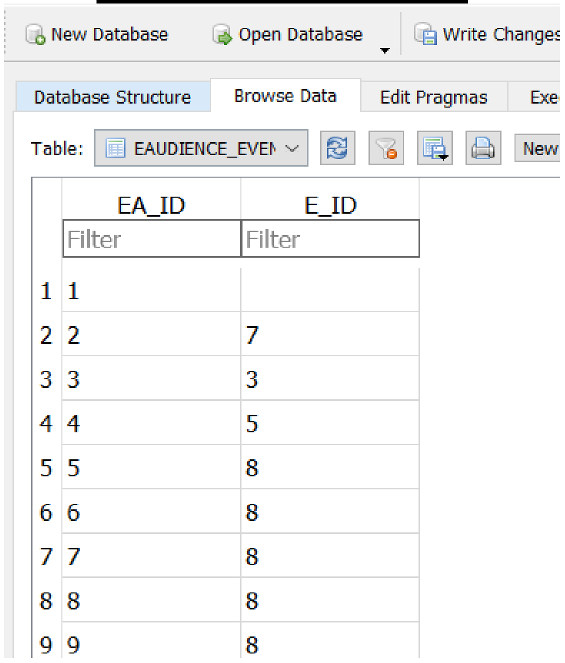
	M_ID	EMAIL_ID
	Filter	Filter
1	1	rt@gmail.com
2	1	riya.tendulkar...
3	1	riya@hotmail....
4	2	vishuanshu@...
5	2	vaishnavi@ho...

CAUDIENCE_EVENTS



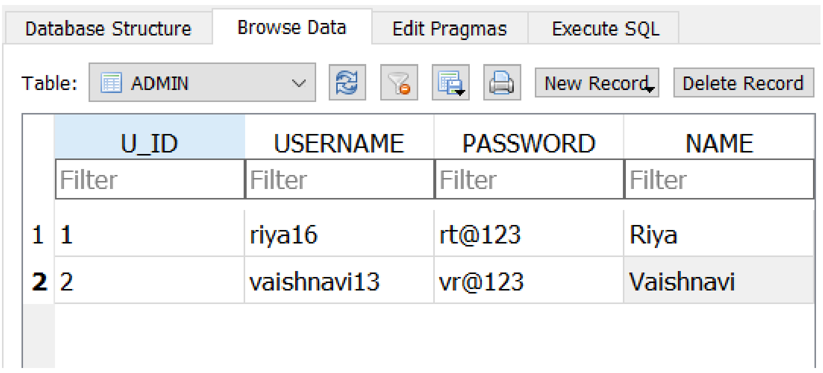
	A_ID	E_ID
1	1	4
2	2	8
3	3	13
4	4	4
5	5	8
6	6	8
7	7	8
8	8	8
9	1	3
10	2	4
11	6	4

EAUDIENCE_EVENT



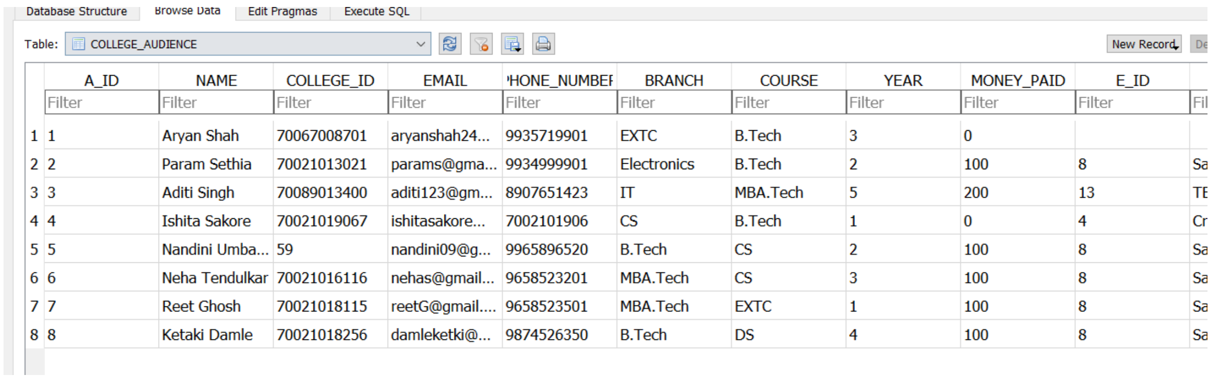
	EA_ID	E_ID
1	1	
2	2	7
3	3	3
4	4	5
5	5	8
6	6	8
7	7	8
8	8	8
9	9	8

ADMIN



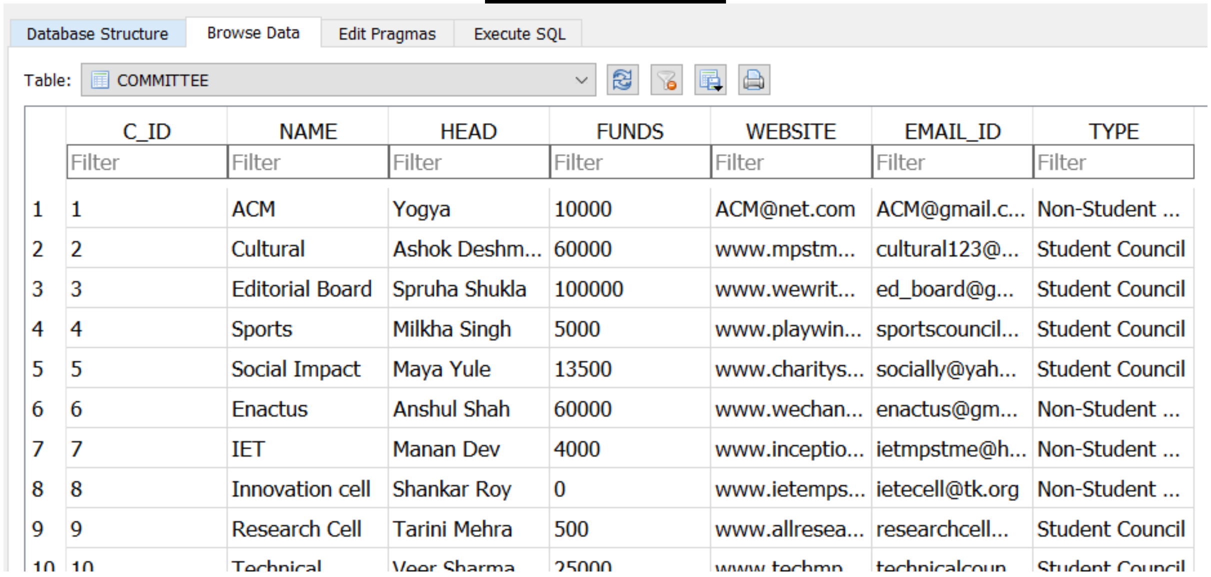
	U_ID	USERNAME	PASSWORD	NAME
	Filter	Filter	Filter	Filter
1	1	riya16	rt@123	Riya
2	2	vaishnavi13	vr@123	Vaishnavi

COLLEGE_AUDIENCE



	A_ID	NAME	COLLEGE_ID	EMAIL	'HONE_NUMBEI	BRANCH	COURSE	YEAR	MONEY_PAID	E_ID	
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Fil
1	1	Aryan Shah	70067008701	aryanshah24...	9935719901	EXTC	B.Tech	3	0		
2	2	Param Sethia	70021013021	params@gma...	9934999901	Electronics	B.Tech	2	100	8	Sa
3	3	Aditi Singh	70089013400	aditi123@gm...	8907651423	IT	MBA.Tech	5	200	13	TE
4	4	Ishita Sakore	70021019067	ishitasakore...	7002101906	CS	B.Tech	1	0	4	Cr
5	5	Nandini Umba...	59	nandini09@g...	9965896520	B.Tech	CS	2	100	8	Sa
6	6	Neha Tendulkar	70021016116	nehas@gmail...	9658523201	MBA.Tech	CS	3	100	8	Sa
7	7	Reet Ghosh	70021018115	reetG@gmail....	9658523501	MBA.Tech	EXTC	1	100	8	Sa
8	8	Ketaki Damle	70021018256	damleketki@...	9874526350	B.Tech	DS	4	100	8	Sa

COMMITTEE



	C_ID	NAME	HEAD	FUNDS	WEBSITE	EMAIL_ID	TYPE
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	ACM	Yogya	10000	ACM@net.com	ACM@gmail.c...	Non-Student ...
2	2	Cultural	Ashok Deshm...	60000	www.mpstm...	cultural123@...	Student Council
3	3	Editorial Board	Spruha Shukla	100000	www.wewrit...	ed_board@g...	Student Council
4	4	Sports	Milkha Singh	5000	www.playwin...	sportscouncil...	Student Council
5	5	Social Impact	Maya Yule	13500	www.charitys...	socially@yah...	Student Council
6	6	Enactus	Anshul Shah	60000	www.wechan...	enactus@gm...	Non-Student ...
7	7	IET	Manan Dev	4000	www.inceptio...	ietmpstme@h...	Non-Student ...
8	8	Innovation cell	Shankar Roy	0	www.ietemps...	ietecell@tk.org	Non-Student ...
9	9	Research Cell	Tarini Mehra	500	www.allresea...	researchcell...	Student Council
10	10	Technical	Veer Sharma	25000	www.techmn	technicalcoun	Student Council

EAUDIENCE_EVENTS

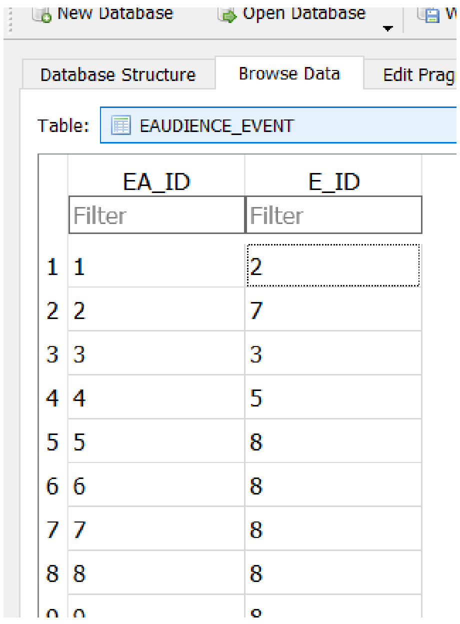


Table: EAUDIENCE_EVENT

	EA_ID	E_ID
1	1	2
2	2	7
3	3	3
4	4	5
5	5	8
6	6	8
7	7	8
8	8	8
9	9	8

EVENTS

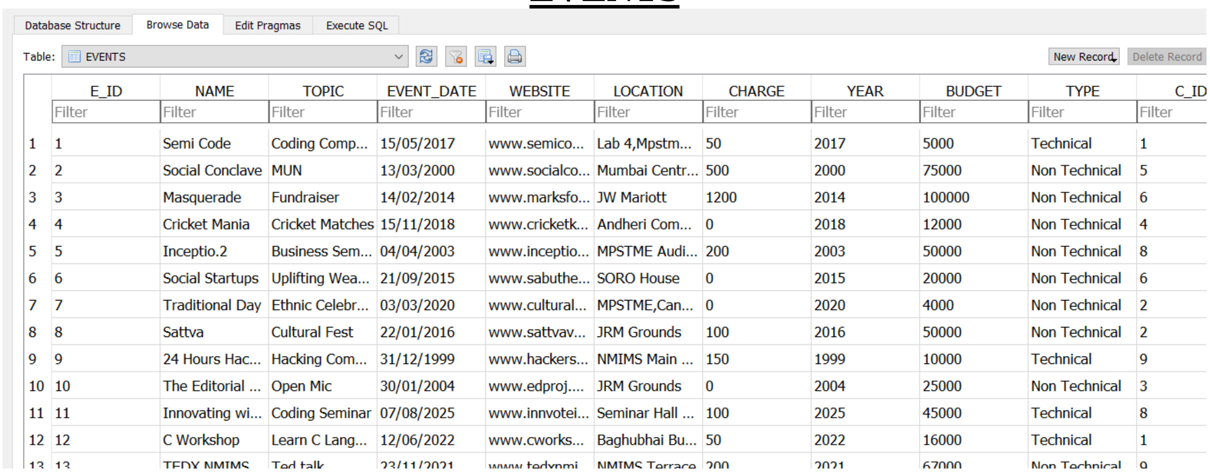


Table: EVENTS

	E_ID	NAME	TOPIC	EVENT_DATE	WEBSITE	LOCATION	CHARGE	YEAR	BUDGET	TYPE	C_ID
1	1	Semi Code	Coding Comp...	15/05/2017	www.semico...	Lab 4,Mpstm...	50	2017	5000	Technical	1
2	2	Social Conclave	MUN	13/03/2000	www.socialco...	Mumbai Centr...	500	2000	75000	Non Technical	5
3	3	Masquerade	Fundraiser	14/02/2014	www.marksfo...	JW Mariott	1200	2014	100000	Non Technical	6
4	4	Cricket Mania	Cricket Matches	15/11/2018	www.cricketk...	Andheri Com...	0	2018	12000	Non Technical	4
5	5	Inceptio.2	Business Sem...	04/04/2003	www.inceptio...	MPSTME Audi...	200	2003	50000	Non Technical	8
6	6	Social Startups	Uplifting Wea...	21/09/2015	www.sabuthe...	SORO House	0	2015	20000	Non Technical	6
7	7	Traditional Day	Ethnic Celebr...	03/03/2020	www.cultural...	MPSTME,Can...	0	2020	4000	Non Technical	2
8	8	Sattva	Cultural Fest	22/01/2016	www.sattvav...	JRM Grounds	100	2016	50000	Non Technical	2
9	9	24 Hours Hac...	Hacking Com...	31/12/1999	www.hackers...	NMIMS Main ...	150	1999	10000	Technical	9
10	10	The Editorial ...	Open Mic	30/01/2004	www.edproj....	JRM Grounds	0	2004	25000	Non Technical	3
11	11	Innovating wi...	Coding Seminar	07/08/2025	www.innvotei...	Seminar Hall ...	100	2025	45000	Technical	8
12	12	C Workshop	Learn C Lang...	12/06/2022	www.cworks...	Baghubhai Bu...	50	2022	16000	Technical	1
13	13	TEDX NMIMS	Ted talk	23/11/2021	www.tedxnmi...	NMIMS Terrace	200	2021	67000	Non Technical	9

EXTERNAL_EVENTS

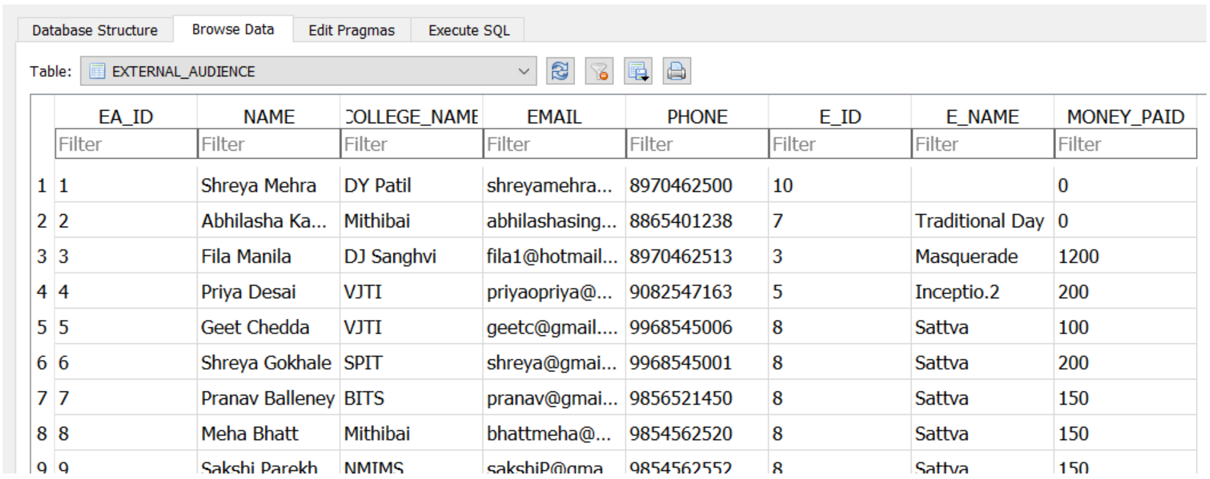


Table: EXTERNAL_AUDIENCE

	EA_ID	NAME	COLLEGE_NAME	EMAIL	PHONE	E_ID	E_NAME	MONEY_PAID
1	1	Shreya Mehra	DY Patil	shreyamehra...	8970462500	10		0
2	2	Abhilasha Ka...	Mithibai	abhilashasing...	8865401238	7	Traditional Day	0
3	3	Fila Manila	DJ Sanghvi	fila1@hotmail...	8970462513	3	Masquerade	1200
4	4	Priya Desai	VJTI	priyaopriya@...	9082547163	5	Inceptio.2	200
5	5	Geet Chedda	VJTI	geetc@gmail....	9968545006	8	Sattva	100
6	6	Shreya Gokhale	SPIT	shreya@gmai...	9968545001	8	Sattva	200
7	7	Pranav Balleney	BITS	pranav@gmai...	9856521450	8	Sattva	150
8	8	Meha Bhatt	Mithibai	bhattmeha@...	9854562520	8	Sattva	150
9	9	Sakshi Parekh	NMIMS	sakshiP@gma	9854562552	8	Sattva	150

MEMBERS

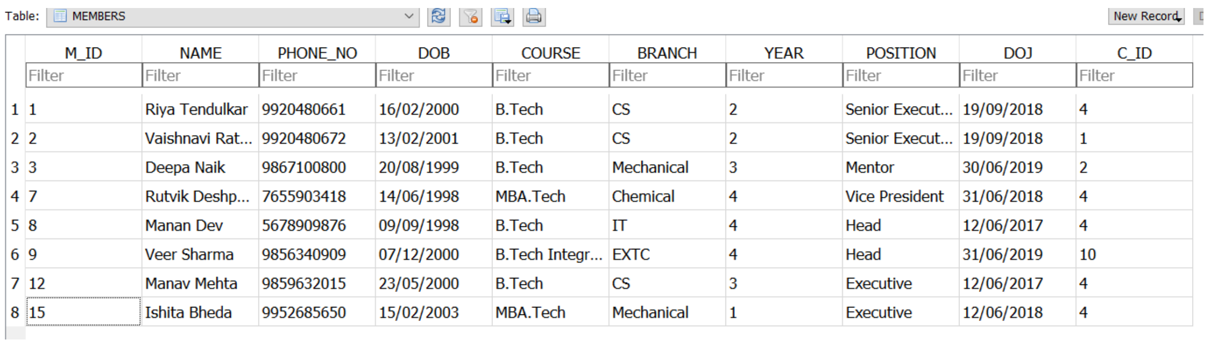


Table: MEMBERS

New Record

	M_ID	NAME	PHONE_NO	DOB	COURSE	BRANCH	YEAR	POSITION	DOJ	C_ID
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Riya Tendulkar	9920480661	16/02/2000	B.Tech	CS	2	Senior Execut...	19/09/2018	4
2	2	Vaishnavi Rat...	9920480672	13/02/2001	B.Tech	CS	2	Senior Execut...	19/09/2018	1
3	3	Deepa Naik	9867100800	20/08/1999	B.Tech	Mechanical	3	Mentor	30/06/2019	2
4	7	Rutvik Deshp...	7655903418	14/06/1998	MBA.Tech	Chemical	4	Vice President	31/06/2018	4
5	8	Manan Dev	5678909876	09/09/1998	B.Tech	IT	4	Head	12/06/2017	4
6	9	Veer Sharma	9856340909	07/12/2000	B.Tech Integr...	EXTC	4	Head	31/06/2019	10
7	12	Manav Mehta	9859632015	23/05/2000	B.Tech	CS	3	Executive	12/06/2017	4
8	15	Ishita Bheda	9952685650	15/02/2003	MBA.Tech	Mechanical	1	Executive	12/06/2018	4

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.

CHAPTER 5: APPENDIX

Source code

MAIN PAGE

```
from tkinter import *
from Login import *
from CommitteeMainPage import *
from ExternalMainPage import *
def MainPageGUI():
    mainPage=Tk()
    mainPage.title("Main Page")
    mainPage.geometry("640x350")
    mainPage.configure(bg="black")
    l1=Label(mainPage,text="WELCOME TO EVENT MANAGEMENT
SYSTEM",font="Times 20 bold underline",bg="black",foreground="snow")
    l1.grid(row=0,column=0)
    l2=Label(mainPage,text="Hi, Welcome. Please select what type of user you are. Click on
the button!", font="Times 15 bold",bg="black",foreground="snow")
    l2.grid(row=1,column=0)
    LExternalUser=Label(mainPage,text="External User (Registering for any
event)",font="Times 15",bg="black",foreground="snow")
    LExternalUser.grid(row=2,column=0)
    ExternalUser=Button(mainPage,text="Click Here",font="Times
10",activebackground="red",command=ExternalMainPageGUI)
    ExternalUser.grid(row=3,column=0)
    LCommitteeMember=Label(mainPage,text="Committee Member",font="Times
15",bg="black",foreground="snow")
    LCommitteeMember.grid(row=4,column=0)
    CommitteeMember=Button(mainPage,text="Click Here",font="Times
10",activebackground="red",command=CommitteeMainPageGUI)
    CommitteeMember.grid(row=5,column=0)
    LAdmin=Label(mainPage,text="Admin",font="Times
15",bg="black",foreground="snow")
    LAdmin.grid(row=6,column=0)
    Admin=Button(mainPage,text="Click Here",font="Times
10",activebackground="red",command=LoginGUI)
    Admin.grid(row=7,column=0)
MainPageGUI()
```

AUDIENCE

```
from tkinter import *
from DatabaseAddPage import *
from College_Audience import *
from External_Audience import *
def AudienceGUI():
    t=Tk()
    t.geometry("500x250")
    t.title("Audience Page")
    t.configure(bg="black")
    l1=Label(t,text="ADD DETAILS FOR AUDIENCE",font="Times 20 bold
underline",bg="black",foreground="snow")
    l1.grid(row=0,column=0)
    l2=Label(t,text="Select what type of audience you are",font="Times
15",bg="black",foreground="snow")
    l2.grid(row=2,column=0)

    #Setting Label
    Lcollege=Label(t,text="College
Student",bg="black",foreground="snow",font="Baskerville 12 bold")
    Lexternal=Label(t,text="Outside College
Student",bg="black",foreground="snow",font="Baskerville 12 bold")

    #Adding Label
    Lcollege.grid(row=3,column=0)
    Lexternal.grid(row=4,column=0)

    #Creating Buttons
    college=Button(t,text="Click Here",font="Times 12
bold",activebackground="red",command=College_AudienceGUI)
    external=Button(t,text="Click Here",font="Times 12
bold",activebackground="red",command=External_AudienceGUI)

    #Adding Pages
    college.grid(row=3,column=1)
    external.grid(row=4,column=1)
college audience
from tkinter import *
import sqlite3
connection = sqlite3.connect("Event_Management.db")
conn= connection.cursor()


def creatingTableCollege_Audience():
    global
a_id,a_name,college_id,a_email,a_phone,e_id,e_name,a_branch,a_course,a_year,money_pai
d
    #Creating table if it doesn't exits
    conn.execute("CREATE TABLE IF NOT EXISTS COLLEGE_AUDIENCE(A_ID
INTEGER PRIMARY KEY AUTOINCREMENT,NAME TEXT NOT
NULL,COLLEGE_ID INTEGER NOT NULL,EMAIL VARCHAR NOT
NULL,PHONE_NUMBER NOT NULL,BRANCH TEXT,COURSE TEXT,YEAR
NUMBER,MONEY_PAID NUMBER,E_ID INTEGER REFERENCES
EVENTS(E_ID),E_NAME TEXT REFERENCES EVENTS(E_NAME))")
    connection.commit()
    #Inserting values into the table
    conn.execute("INSERT INTO
COLLEGE_AUDIENCE(NAME,COLLEGE_ID,EMAIL,PHONE_NUMBER,BRANCH,C
OURSE,YEAR,MONEY_PAID,E_ID,E_NAME)
VALUES(?,?,?,?,?,?,?,?,?,?)",(a_name.get(),college_id.get(),a_email.get(),a_phone.get(),a_br
anch.get(),a_course.get(),a_year.get(),money_paid.get(),e_id.get(),e_name.get()))
   conn.execute("INSERT INTO CAUDIENCE_EVENT(A_ID,E_ID)
VALUES(?,?)",(conn.lastrowid,e_id.get()))
   connection.commit()
   #Permanantly making changes to the database
   conn.execute("SELECT * FROM COLLEGE_AUDIENCE")
   #Displaying the data for the user's convenience
   for row in conn.fetchall():
      print(row)

def College_AudienceGUI():
   global
a_id,a_name,college_id,a_email,a_phone,e_id,e_name,a_branch,a_course,a_year,money_pai
d
   #Setting window properties
   t=Tk()
   t.title("College Audience")
   t.geometry("500x350")
   t.configure(bg="black")
   l1=Label(t,text="College Audience Details",font="Times 20 bold
underline",bg="black",foreground="snow")
   l1.grid(row=0,column=0)
   #Defining Labels
   Lname=Label(t,text="NAME OF
AUDIENCE",bg="black",foreground="snow",font="Baskerville 12 bold")
   Lcollege_id=Label(t,text="COLLEGE
ID",bg="black",foreground="snow",font="Baskerville 12 bold")
   Lemail=Label(t,text="EMAIL ID",bg="black",foreground="snow",font="Baskerville 12
bold")
   Lphone=Label(t,text="PHONE NO",bg="black",foreground="snow",font="Baskerville 12
bold")
   Lbranch=Label(t,text="BRANCH
NAME",bg="black",foreground="snow",font="Baskerville 12 bold")
   Lcourse=Label(t,text="COURSE
NAME",bg="black",foreground="snow",font="Baskerville 12 bold")
   Lyear=Label(t,text="YEAR",bg="black",foreground="snow",font="Baskerville 12 bold")
   Lmoney=Label(t,text="MONEY PAID",bg="black",foreground="snow",font="Baskerville
12 bold")
   Le_id=Label(t,text="EVENT ID",bg="black",foreground="snow",font="Baskerville 12
bold")
   Le_name=Label(t,text="EVENT
NAME",bg="black",foreground="snow",font="Baskerville 12 bold")


   #Defining Entry Boxes
   a_name=Entry(t)
   college_id=Entry(t)
   a_email=Entry(t)
   a_phone=Entry(t)
```

```
    a_branch=Entry(t)
    a_course=Entry(t)
    a_year=Entry(t)
    money_paid=Entry(t)
    e_id=Entry(t)
    e_name=Entry(t)

    #Adding Labels to window

    Lname.grid(row=2,column=0)
    Lcollege_id.grid(row=3,column=0)
    Lemail.grid(row=4,column=0)
    Lphone.grid(row=5,column=0)
    Lbranch.grid(row=8,column=0)
    Lcourse.grid(row=9,column=0)
    Lyear.grid(row=10,column=0)
    Lmoney.grid(row=11,column=0)
    Le_id.grid(row=12,column=0)
    Le_name.grid(row=13,column=0)

    #Adding Entry box to window

    a_name.grid(row=2,column=1)
    college_id.grid(row=3,column=1)
    a_email.grid(row=4,column=1)
    a_phone.grid(row=5,column=1)
    a_branch.grid(row=8,column=1)
    a_course.grid(row=9,column=1)
    a_year.grid(row=10,column=1)
    money_paid.grid(row=11,column=1)
    e_id.grid(row=12,column=1)
    e_name.grid(row=13,column=1)

    #Recieving and passing all the values
    bSubmit=Button(t,text="Submit",command=creatingTableCollege_Audience,font="Times
12 bold",activebackground="red")
    bSubmit.grid(row=14,column=1)
    connection.commit()


committee
from tkinter import *
import sqlite3
connection = sqlite3.connect("Event_Management.db")
conn= connection.cursor()



def creatingTableCommittee():
    global comm_name,comm_email,comm_website,headed_by,Type,money_available
    #Creating table if it is not present
    conn.execute("CREATE TABLE IF NOT EXISTS COMMITTEE(C_ID INTEGER
PRIMARY KEY AUTOINCREMENT, NAME TEXT NOT NULL, HEAD TEXT NOT
NULL, FUNDS INTEGER, WEBSITE VARCHAR, EMAIL_ID VARCHAR, TYPE
TEXT)")
    #Inserting values to table
    conn.execute("INSERT INTO
COMMITTEE(NAME,HEAD,FUNDS,WEBSITE,EMAIL_ID,TYPE)
VALUES(?,?,?,?,?,?)",(comm_name.get(),headed_by.get(),money_available.get(),comm_we
bsite.get(),comm_email.get(),Type.get()))
    #Permanently saving the data to the database
    connection.commit()
    #Displaying the data on shell for convenince of user
    conn.execute("SELECT * FROM COMMITTEE")
    for row in conn.fetchall():
        print(row)

def CommitteeGUI():
    global comm_name,comm_email,comm_website,headed_by,Type,money_available
    #Setting properties of the window
    t=Tk()
    t.title("Committee")
    t.configure(bg="black")
    t.geometry("400x300")
    l1=Label(t,text="Committee Details",font="Times 20 bold
underline",bg="black",foreground="snow")
    l1.grid(row=0,column=0)

    #Defining LAbels
    Lcomm_name=Label(t,text="NAME",bg="black",foreground="snow",font="Baskerville
12 bold")
    Lemail=Label(t,text="EMAIL ID",bg="black",foreground="snow",font="Baskerville 12
bold")
    Lweb=Label(t,text="WEBSITE",bg="black",foreground="snow",font="Baskerville 12
bold")
    Lhead=Label(t,text="HEADED BY",bg="black",foreground="snow",font="Baskerville 12
bold")
    Ltype=Label(t,text="TYPE",bg="black",foreground="snow",font="Baskerville 12 bold")
    Lmoney=Label(t,text="FUNDS",bg="black",foreground="snow",font="Baskerville 12
bold")

    #Defining Entry Boxes

    comm_name=Entry(t)
    comm_email=Entry(t)
    comm_website=Entry(t)
    headed_by=Entry(t)
    Type=Entry(t)
    money_available=Entry(t)

    #Adding Labels to window
    Lcomm_name.grid(row=2,column=0)
    Lemail.grid(row=3,column=0)
    Lweb.grid(row=4,column=0)
    Lhead.grid(row=5,column=0)
    Ltype.grid(row=6,column=0)
    Lmoney.grid(row=7,column=0)

    #Adding Entry box to window

    comm_name.grid(row=2,column=1)
    comm_email.grid(row=3,column=1)
    comm_website.grid(row=4,column=1)
    headed_by.grid(row=5,column=1)
    Type.grid(row=6,column=1)
    money_available.grid(row=7,column=1)


    #Recieving and passing all the values
    bSubmit=Button(t,text="Submit",command=creatingTableCommittee,font="Times 12
bold",activebackground="red")
    bSubmit.grid(row=11,column=1)
    connection.commit()
committee main page
from tkinter import *
from Total_Cost import *
from Members import *
from Remove_Member import *
from Resource_Event import *
from Guest_Event import *
from Sponsor_Event import *
import sqlite3
connection = sqlite3.connect("Event_Management.db")
cursor = connection.cursor()
def DisplayMenu():
    global y
    y.destroy()
    t=Tk()
    t.title("Committee Page")
    t.configure(bg="black")
    t.geometry("600x400")
    l1=Label(t,text="WELCOME TO COMMITTEE PAGE",font="Times 20 bold
underline",bg="black",foreground="snow")
    l1.grid(row=0,column=0)
    l2=Label(t,text="Hi, Welcome. Please select your choice!Click on the
button!",bg="black",foreground="snow",font="Baskerville 12 bold")
    l2.grid(row=1,column=0)

    #Creating  Labels
    Lcost=Label(t,text="Calculate Cost of
Event",bg="black",foreground="snow",font="Baskerville 12 bold")
    Ladd=Label(t,text="Add Member to
Committee",bg="black",foreground="snow",font="Baskerville 12 bold")
    Lremove=Label(t,text="Remove
Member",bg="black",foreground="snow",font="Baskerville 12 bold")
    Lspon=Label(t,text="Add Sponsors to
Event",bg="black",foreground="snow",font="Baskerville 12 bold")
    Lresource=Label(t,text="Add Resources to
Event",bg="black",foreground="snow",font="Baskerville 12 bold")
    Lguest=Label(t,text="Add Guests to
Event",bg="black",foreground="snow",font="Baskerville 12 bold")


    #Adding Labels
    Lcost.grid(row=2,column=0)
    Ladd.grid(row=3,column=0)
    Lremove.grid(row=4,column=0)
    Lspon.grid(row=5,column=0)
    Lresource.grid(row=6,column=0)
    Lguest.grid(row=7,column=0)


    #Creating Buttons
    cost=Button(t,text="Click here!",font="Times 12
bold",activebackground="red",command=lambda: destructTotalCost(t))
    add=Button(t,text="Click here!",font="Times 12
bold",activebackground="red",command=MembersGUI)
    remove=Button(t,text="Click here!",font="Times 12
bold",activebackground="red",command=RemoveMemberGUI)
    spon=Button(t,text="Click here!",font="Times 12
bold",activebackground="red",command=Sponsor_EventGUI)
    resource=Button(t,text="Click here!",font="Times 12
bold",activebackground="red",command=Resource_EventGUI)
    guest=Button(t,text="Click here!",font="Times 12
bold",activebackground="red",command=Guest_EventGUI)


    #Adding Buttons
    cost.grid(row=2,column=1)
    add.grid(row=3,column=1)
    remove.grid(row=4,column=1)
    spon.grid(row=5,column=1)
    resource.grid(row=6,column=1)
    guest.grid(row=7,column=1)

def Pass():
    global e
    find_comm=("SELECT C_ID FROM COMMITTEE WHERE NAME=?")
    cursor.execute(find_comm,[(e.get())])
    if cursor.fetchone() is not None:
        cursor.execute(find_comm,[(e.get())])
        store=cursor.fetchone()
        getId(store)
        getID(store)
        l=Label(y,text="Correct Committee ID! You can
proceed!",bg="black",foreground="snow",font="Baskerville 12 bold")
        l.grid(row=2,column=0)
        b2=Button(y,text="Proceed",command=DisplayMenu,font="Times 12
bold",activebackground="red")
        b2.grid(row=3,column=0)
    else:
        incorrect=Tk()
        incorrect.configure(bg="black")
        incorrect.title("Wrong Data")
        l1=Label(incorrect,text="Incorrect committee
name!!",bg="black",foreground="snow",font="Baskerville 12 bold")
        l1.pack()
        desButton=Button(incorrect,text="Try Again",font="Times 12
bold",activebackground="red",command=incorrect.destroy)
        desButton.pack()

def CommitteeMainPageGUI():
    global e,y
    y=Tk()
    y.title("Check ID")
    y.configure(bg="black")
    l=Label(y,text="Enter your committee
name:",bg="black",foreground="snow",font="Baskerville 12 bold")
    e=Entry(y)
    l.grid(row=0,column=0)
    e.grid(row=0,column=1)
    b1=Button(y,text="Proceed",command=Pass,font="Times 12
bold",activebackground="red")
    b1.grid(row=1,column=0)


database add page
from tkinter import *
from Admin import *
from Events import *
from Committee import *
from Members import *
from Guests import *
from Audience import *
from Sponsors import *
from Resource import *
class DatabaseAddPage:
    def DatabaseAddPageGUI():
        t=Tk()
```

```
	t.title("Main Menu")
	t.geometry("500x500")
	t.configure(bg="black")
	l1=Label(t,text="ADD DETAILS",font="Times 20 bold
underline",bg="black",foreground="snow")
	l1.grid(row=0,column=0)
	l2=Label(t,text="On this page you can add details in the database here",font="Times
15",bg="black",foreground="snow")
	l2.grid(row=1,column=0)
	l3=Label(t,text="Click on the button to add the details!",font="Times
15",bg="black",foreground="snow")
	l3.grid(row=2,column=0)

	#Creating LAbels labels
	Levent=Label(t,text="Event",bg="black",foreground="snow",font="Baskerville 12
bold")

Lcommittee=Label(t,text="Committee",bg="black",foreground="snow",font="Baskerville 12
bold")
	Lmembers=Label(t,text="Committee
Members",bg="black",foreground="snow",font="Baskerville 12 bold")
	Lguest=Label(t,text="Guests",bg="black",foreground="snow",font="Baskerville 12
bold")
	Laudience=Label(t,text="Audience",bg="black",foreground="snow",font="Baskerville
12 bold")
	Lsponsors=Label(t,text="Sponsors",bg="black",foreground="snow",font="Baskerville
12 bold")
	Lresources=Label(t,text="Resources",bg="black",foreground="snow",font="Baskerville
12 bold")
	Ladmin=Label(t,text="Admin",bg="black",foreground="snow",font="Baskerville 12
bold")

	#Adding LAbels
	Levent.grid(row=3,column=0)
	Lcommittee.grid(row=4,column=0)
	Lmembers.grid(row=5,column=0)
	Lguest.grid(row=6,column=0)
	Laudience.grid(row=7,column=0)
	Lsponsors.grid(row=8,column=0)
	Lresources.grid(row=9,column=0)
	Ladmin.grid(row=10,column=0)

	#Creating Buttons
	event=Button(t,text="Click Here",font="Times 12
bold",activebackground="red",command=EventsGUI)
	committee=Button(t,text="Click Here",font="Times 12
bold",activebackground="red",command=CommitteeGUI)
	members=Button(t,text="Click Here",font="Times 12
bold",activebackground="red",command=MembersGUI)
```

```
        guest=Button(t,text="Click Here",font="Times 12
bold",activebackground="red",command=GuestsGUI)
        audience=Button(t,text="Click Here",font="Times 12
bold",activebackground="red",command=AudienceGUI)
        sponsors=Button(t,text="Click Here",font="Times 12
bold",activebackground="red",command=SponsorsGUI)
        resources=Button(t,text="Click Here",font="Times 12
bold",activebackground="red",command=ResourceGUI)
        admin=Button(t,text="Click Here",font="Times 12
bold",activebackground="red",command=AdminGUI)

        #Adding Buttons
        event.grid(row=3,column=1)
        committee.grid(row=4,column=1)
        members.grid(row=5,column=1)
        guest.grid(row=6,column=1)
        audience.grid(row=7,column=1)
        sponsors.grid(row=8,column=1)
        resources.grid(row=9,column=1)
        admin.grid(row=10,column=1)
        t.mainloop()


display guests
from tkinter import *
import sqlite3
connection = sqlite3.connect("Event_Management.db")
cursor = connection.cursor()
def disp():
    global e1,e2,t
    f1=("select name,profession from guests where g_id in (select g_id from GUEST_EVENT
where e_id=(SELECT E_ID from events where name=? and year=?))")
    cursor.execute(f1,[(e1.get()),(e2.get())])
    for row in cursor.fetchall():
        l=Label(t,text=row)
        l.pack()
def DisplayGuestsGUI():
    global e1,e2,t
    t=Tk()
    t.configure(bg="black")
    l1=Label(t,text="Hi user, here you can see the guests who will be performing at an
event!",font="Times 15 bold underline",bg="black",foreground="snow")
    l1.pack()
    l2=Label(t,text="Enter event name",bg="black",foreground="snow",font="Baskerville 12
bold")
    l2.pack()
    e1=Entry(t)
    e1.pack()
    l3=Label(t,text="Enter the year of the
event",bg="black",foreground="snow",font="Baskerville 12 bold")
```

```
    l3.pack()
    e2=Entry(t)
    e2.pack()
    b1=Button(t,text="Submit",command=disp,font="Times 12 
bold",activebackground="red")
    b1.pack()
```

events

```
from tkinter import *
import sqlite3
connection = sqlite3.connect("Event_Management.db")
conn= connection.cursor()


def creatingTableEvents():
    global name,topic,location,date,Type,year,website,money,budget,CID
    #Creating table if it is not present
    conn.execute("CREATE TABLE IF NOT EXISTS EVENTS(E_ID INTEGER PRIMARY 
KEY AUTOINCREMENT,NAME TEXT NOT NULL, TOPIC TEXT,EVENT_DATE 
DATE, WEBSITE VARCHAR, LOCATION TEXT,CHARGE INTEGER,YEAR 
INTEGER, BUDGET INTEGER,TYPE TEXT,C_ID INTEGER,FOREIGN KEY (C_ID) 
REFERENCES COMMITTEE(C_ID))")
    #Inserting values to table
    conn.execute("INSERT INTO 
EVENTS(NAME,TOPIC,EVENT_DATE,LOCATION,WEBSITE,CHARGE,YEAR,BUDG
ET,TYPE,C_ID) 
VALUES(?,?,?,?,?,?,?,?,?,?)",(name.get(),topic.get(),date.get(),location.get(),website.get(),m
oney.get(),year.get(),budget.get(),Type.get(),CID.get()))
    #Permanently saving the data to the database
    connection.commit()
    #Displaying the data on shell for convenince of user
    conn.execute("SELECT * FROM EVENTS")
    for row in conn.fetchall():
        print(row)


def EventsGUI():
    global name,topic,location,date,Type,year,website,money,budget,CID

    #Setting the window properties
    t=Tk()
    t.title("Events")
    t.geometry("400x400")
    t.configure(bg="black")
    l1=Label(t,text="Events Details",font="Times 20 bold 
underline",bg="black",foreground="snow")
    l1.grid(row=0,column=0)

    #Defining LAbels
    Lname=Label(t,text="NAME",bg="black",foreground="snow",font="Baskerville 12 bold")
    Ltopic=Label(t,text="TOPIC",bg="black",foreground="snow",font="Baskerville 12 bold")
    Llocation=Label(t,text="LOCATION",bg="black",foreground="snow",font="Baskerville 12 bold")
    Ldate=Label(t,text="DATE",bg="black",foreground="snow",font="Baskerville 12 bold")
    Ltype=Label(t,text="TYPE",bg="black",foreground="snow",font="Baskerville 12 bold")
    Lyear=Label(t,text="YEAR",bg="black",foreground="snow",font="Baskerville 12 bold")
    Lwebsite=Label(t,text="WEBSITE",bg="black",foreground="snow",font="Baskerville 12 bold")
    Lmoney=Label(t,text="CHARGE",bg="black",foreground="snow",font="Baskerville 12 bold")
    Lbudget=Label(t,text="BUDGET",bg="black",foreground="snow",font="Baskerville 12 bold")
    LCID=Label(t,text="COMMITTEE ID",bg="black",foreground="snow",font="Baskerville 12 bold")


    #Defining Entry Boxes
    name=Entry(t)
    topic=Entry(t)
    location=Entry(t)
    date=Entry(t)
    Type=Entry(t)
    year=Entry(t)
    website=Entry(t)
    money=Entry(t)
    budget=Entry(t)
    CID=Entry(t)

    #Adding Labels to window
    Lname.grid(row=1,column=0)
    Ltopic.grid(row=2,column=0)
    Llocation.grid(row=3,column=0)
    Ldate.grid(row=4,column=0)
    Ltype.grid(row=6,column=0)
    Lyear.grid(row=7,column=0)
    Lwebsite.grid(row=8,column=0)
    Lmoney.grid(row=9,column=0)
    Lbudget.grid(row=10,column=0)
    LCID.grid(row=11,column=0)


    #Adding Entry box to window
    name.grid(row=1,column=1)
    topic.grid(row=2,column=1)
    location.grid(row=3,column=1)
    date.grid(row=4,column=1)
    Type.grid(row=6,column=1)
    year.grid(row=7,column=1)
    website.grid(row=8,column=1)
    money.grid(row=9,column=1)
    budget.grid(row=10,column=1)
    CID.grid(row=11,column=1)


    #Recieving and passing all the values
    bSubmit=Button(t,text="Submit",font="Times 12
bold",activebackground="red",command=creatingTableEvents)
    bSubmit.grid(row=13,column=1)


external audience
from tkinter import *
import sqlite3
connection = sqlite3.connect("Event_Management.db")
conn= connection.cursor()


def creatingTableExternal_Audience():
    global ea_id,ea_name,college_name,ea_email,ea_phone,e_id,e_name,money_paid
    #Creating table if it doesn't exits
    conn.execute("CREATE TABLE IF NOT EXISTS EXTERNAL_AUDIENCE(EA_ID
INTEGER PRIMARY KEY AUTOINCREMENT,NAME TEXT NOT
NULL,COLLEGE_NAME TEXT NOT NULL,EMAIL VARCHAR NOT NULL,PHONE
NUMBER NOT NULL,E_ID INTEGER REFERENCES EVENTS(E_ID),E_NAME TEXT
REFERENCES EVENTS(E_NAME),MONEY_PAID NUMBER)")
    connection.commit()
    #Inserting values into the table
    conn.execute("INSERT INTO
EXTERNAL_AUDIENCE(NAME,COLLEGE_NAME,EMAIL,PHONE,E_ID,E_NAME,M
ONEY_PAID)
VALUES(?,?,?,?,?,?,?)",(ea_name.get(),college_name.get(),ea_email.get(),ea_phone.get(),e_i
d.get(),e_name.get(),money_paid.get()))
    conn.execute("INSERT INTO EAUDIENCE_EVENT(EA_ID,E_ID)
VALUES(?,?)",(conn.lastrowid,e_id.get()))
    connection.commit()
    #Permanantly making changes to the database
    conn.execute("SELECT * FROM EXTERNAL_AUDIENCE")
    #Displaying the data for the user's convenience
    for row in conn.fetchall():
        print(row)

def External_AudienceGUI():
    global ea_id,ea_name,college_name,ea_email,ea_phone,e_id,e_name,money_paid
    #Setting window properties
    t=Tk()
    t.title("External Audience")
    t.geometry("500x300")
    t.configure(bg="black")
    l1=Label(t,text="External Audience Details",font="Times 20 bold
underline",bg="black",foreground="snow")
    l1.grid(row=0,column=0)
    #Defining Labels
    Lname=Label(t,text="NAME OF
AUDIENCE",bg="black",foreground="snow",font="Baskerville 12 bold")
    Lcollege_name=Label(t,text="COLLEGE
NAME",bg="black",foreground="snow",font="Baskerville 12 bold")
    Lemail=Label(t,text="EMAIL ID",bg="black",foreground="snow",font="Baskerville 12
bold")
    Lphone=Label(t,text="PHONE NO",bg="black",foreground="snow",font="Baskerville 12
bold")
    Le_id=Label(t,text="EVENT ID",bg="black",foreground="snow",font="Baskerville 12
bold")
    Le_name=Label(t,text="EVENT
NAME",bg="black",foreground="snow",font="Baskerville 12 bold")
    Lmoney=Label(t,text="MONEY PAID",bg="black",foreground="snow",font="Baskerville
12 bold")


    #Defining Entry Boxes
    ea_name=Entry(t)
    college_name=Entry(t)
    ea_email=Entry(t)
    ea_phone=Entry(t)
    e_id=Entry(t)
    e_name=Entry(t)
    money_paid=Entry(t)

    #Adding Labels to window
    Lname.grid(row=2,column=0)
    Lcollege_name.grid(row=3,column=0)
    Lemail.grid(row=4,column=0)
    Lphone.grid(row=5,column=0)
    Le_id.grid(row=6,column=0)
    Le_name.grid(row=7,column=0)
    Lmoney.grid(row=8,column=0)

    #Adding Entry box to window
    ea_name.grid(row=2,column=1)
    college_name.grid(row=3,column=1)
    ea_email.grid(row=4,column=1)
    ea_phone.grid(row=5,column=1)
    e_id.grid(row=6,column=1)
    e_name.grid(row=7,column=1)
    money_paid.grid(row=8,column=1)

    #Recieving and passing all the values
```

```
bSubmit=Button(t,text="Submit",command=creatingTableExternal_Audience,font="Times
12 bold",activebackground="red")
    bSubmit.grid(row=10,column=1)
    connection.commit()


external main page
from tkinter import *
from Audience import *
from Popularity_Page import *
from Search import *
from Sort import *
from Upcoming import *
from Display_Guests import *
from Sponsor_Count import *
def ExternalMainPageGUI():
    t=Tk()
    t.configure(bg="black")
    l1=Label(t,text="Welcome Audience!",font="Times 20 bold
underline",bg="black",foreground="snow")
    l1.grid(row=0,column=0)
    l2=Label(t,text="Select what you would like to do
next!",bg="black",foreground="snow",font="Baskerville 12 bold")
    l2.grid(row=1,column=0)

    #Creating Label
    LseeEvent=Label(t,text="See Upcoming
Events",bg="black",foreground="snow",font="Baskerville 12 bold")
    Lsort=Label(t,text="Sort Event",bg="black",foreground="snow",font="Baskerville 12
bold")
    Lsearch=Label(t,text="Search Event",bg="black",foreground="snow",font="Baskerville
12 bold")
    Lpop=Label(t,text="See Popularity of
Event",bg="black",foreground="snow",font="Baskerville 12 bold")
    Lguest=Label(t,text="See Guests of
Event",bg="black",foreground="snow",font="Baskerville 12 bold")
    Lspon=Label(t,text="See Sponsors for
Event",bg="black",foreground="snow",font="Baskerville 12 bold")
    Lregister=Label(t,text="Register for
Event",bg="black",foreground="snow",font="Baskerville 12 bold")

    #Adding Events
    LseeEvent.grid(row=4,column=0)
    Lsort.grid(row=5,column=0)
    Lsearch.grid(row=6,column=0)
    Lpop.grid(row=7,column=0)
    Lguest.grid(row=8,column=0)
    Lspon.grid(row=9,column=0)
    Lregister.grid(row=10,column=0)
    #Adding Buttons
    seeEvent=Button(t,text="Click Here",command=upcomingGUI,font="Times 12
bold",activebackground="red")
    sort=Button(t,text="Click Here",command=sortGUI,font="Times 12
bold",activebackground="red")
    search=Button(t,text="Click Here",command=searchGUI,font="Times 12
bold",activebackground="red")
    pop=Button(t,text="Click Here",command=PopularityGUI,font="Times 12
bold",activebackground="red")
    guest=Button(t,text="Click Here",command=DisplayGuestsGUI,font="Times 12
bold",activebackground="red")
    spon=Button(t,text="Click Here",command=SponsorCountGUI,font="Times 12
bold",activebackground="red")
    register=Button(t,text="Click Here",command=AudienceGUI,font="Times 12
bold",activebackground="red")

    #Adding Buttons
    seeEvent.grid(row=4,column=1)
    sort.grid(row=5,column=1)
    search.grid(row=6,column=1)
    pop.grid(row=7,column=1)
    guest.grid(row=8,column=1)
    spon.grid(row=9,column=1)
    register.grid(row=10,column=1)
    t.mainloop()
guest event
from tkinter import *
import sqlite3
connection = sqlite3.connect("Event_Management.db")
conn= connection.cursor()


def creatingTableGuest_Event():
    global g_id,e_id,g_charge
    conn.execute("CREATE TABLE IF NOT EXISTS GUEST_EVENT(G_ID NUMBER
REFERENCES GUESTS(G_ID), E_ID NUMBER REFERENCES
EVENTS(E_ID),CHARGE NUMBER)")
    connection.commit()
    conn.execute("INSERT INTO GUEST_EVENT
VALUES(?,?,?)",(g_id.get(),e_id.get(),g_charge.get()))
    connection.commit()
    conn.execute("SELECT * FROM GUEST_EVENT")
    for row in conn.fetchall():
        print(row)


def Guest_EventGUI():
    global g_id,e_id,g_charge
    t=Tk()
    t.title("Guests for Event")
    t.geometry("400x250")
    t.configure(bg="black")
    l1=Label(t,text="Guests Event Details",font="Times 20 bold
underline",bg="black",foreground="snow")
    l1.grid(row=0,column=0)

    #Defining Labels
    Lg_id=Label(t,text="GUEST ID",bg="black",foreground="snow",font="Baskerville 12
bold")
    Le_id=Label(t,text="EVENT ID",bg="black",foreground="snow",font="Baskerville 12
bold")
    Lg_charge=Label(t,text="CHARGE
REQUIRED",bg="black",foreground="snow",font="Baskerville 12 bold")


    #Defining Entry Boxes
    g_id=Entry(t)
    e_id=Entry(t)
    g_charge=Entry(t)

    #Adding Labels to window
    Lg_id.grid(row=1,column=0)
    Le_id.grid(row=2,column=0)
    Lg_charge.grid(row=3,column=0)

    #Adding Entry box to window
    g_id.grid(row=1,column=1)
    e_id.grid(row=2,column=1)
    g_charge.grid(row=3,column=1)

    #Recieving and passing all the values
    bSubmit=Button(t,text="Submit",command=creatingTableGuest_Event,font="Times 12
bold",activebackground="red")
    bSubmit.grid(row=4,column=1)
    connection.commit()

guests
from tkinter import *
import sqlite3
connection = sqlite3.connect("Event_Management.db")
conn= connection.cursor()


def creatingTableGuests():
    global g_id,g_name,g_profession,event_no,event_name,g_email,g_phone
    #Creating table if it is not present
    conn.execute("CREATE TABLE IF NOT EXISTS GUESTS(G_ID INTEGER PRIMARY
KEY AUTOINCREMENT,NAME TEXT,PROFESSION TEXT,EMAIL
VARCHAR,PHONE_NO INTEGER)")
```

```
    #Inserting values to table
    conn.execute("INSERT INTO GUESTS(NAME,PROFESSION,EMAIL,PHONE_NO)
VALUES(?,?,?,?)",(g_name.get(),g_profession.get(),g_email.get(),g_phone.get()))
    #Permanently saving the data to the database
    conn.execute("SELECT * FROM GUESTS")
    #Displaying the data on shell for convenince of user
    for row in conn.fetchall():
        print(row)

def GuestsGUI():
    global g_id,g_name,g_profession,g_email,g_phone
    #Setting window dimensions
    t=Tk()
    t.title("Guests")
    t.geometry("400x250")
    t.configure(bg="black")
    l1=Label(t,text="Guests Details",font="Times 20 bold
underline",bg="black",foreground="snow")
    l1.grid(row=0,column=0)
    #Defining Labels
    Lg_name=Label(t,text="NAME",bg="black",foreground="snow",font="Baskerville 12
bold")

Lg_profession=Label(t,text="PROFESSION",bg="black",foreground="snow",font="Baskerv
ille 12 bold")
    Lg_email=Label(t,text="EMAIL ID",bg="black",foreground="snow",font="Baskerville 12
bold")
    Lg_phone=Label(t,text="PHONE NO",bg="black",foreground="snow",font="Baskerville
12 bold")


    #Defining Entry Boxes
    g_name=Entry(t)
    g_profession=Entry(t)
    g_email=Entry(t)
    g_phone=Entry(t)

    #Adding Labels to window
    Lg_name.grid(row=2,column=0)
    Lg_profession.grid(row=3,column=0)
    Lg_email.grid(row=6,column=0)
    Lg_phone.grid(row=7,column=0)


    #Adding Entry box to window
    g_name.grid(row=2,column=1)
    g_profession.grid(row=3,column=1)
    g_email.grid(row=6,column=1)
    g_phone.grid(row=7,column=1)
    #Recieving and passing all the values
    bSubmit=Button(t,text="Submit",command=creatingTableGuests,font="Times 12
bold",activebackground="red")
    bSubmit.grid(row=11,column=1)
    connection.commit()

from tkinter import *
from DatabaseAddPage import *
import sqlite3
connection=sqlite3.connect("Event_Management.DB")
cursor=connection.cursor()
def close():
    t.destroy()
#check if input matches info existing in database
def check():
    global u,p
    find_user=("SELECT * FROM ADMIN WHERE USERNAME =? AND PASSWORD=
?")
    cursor.execute(find_user,[(u.get()),(p.get())])
    if cursor.fetchone() is not None:
        DatabaseAddPage.DatabaseAddPageGUI()
    else:
        incorrect=Tk()
        incorrect.configure(bg="black")
        incorrect.title("Wrong Data")
        l1=Label(incorrect,text="Incorrect username or
password",bg="black",foreground="snow",font="Baskerville 12 bold")
        l1.pack()
        desButton=Button(incorrect,text="Try Again",font="Times 12
bold",activebackground="red",command=incorrect.destroy)
        desButton.pack()

login

def LoginGUI():
    global u,p,t
    t=Tk()
    t.configure(bg="black")
    t.title("LOGIN PAGE")

    #string variables used
    u=StringVar()
    p=StringVar()

    l1=Label(t,text="ENTER THE LOGIN DETAILS",font="Times 20 bold
underline",bg="black",foreground="snow")
    l1.grid(row=0,column=0)

    #creating labels
    l2=Label(t,text="Username",bg="black",foreground="snow",font="Baskerville 12 bold")
    l2.grid(row=1,column=0)
    L3=Label(t,text="Password",bg="black",foreground="snow",font="Baskerville 12 bold")
    L3.grid(row=2,column=0)


    #taking inputs
    u=Entry(t)
    u.grid(row=1,column=1)
    p=Entry(t)
    p.grid(row=2,column=1)

    #submit button
    s=Button(t,text="Submit",command=check,font="Times 12
bold",activebackground="red")
    s.grid(row=3,column=0)
    q=Button(t,text="Quit",command=close,font="Times 12 bold",activebackground="red")
    q.grid(row=3,column=1)
    t.mainloop()

members
from tkinter import *
import sqlite3
connection = sqlite3.connect("Event_Management.db")
conn= connection.cursor()

def creatingTableCommitteeMembers():
    global name,emailID,phoneNo,birthdate,course,branch,year,position,doj,cid
    #Creating table if it is not present
    conn.execute("CREATE TABLE IF NOT EXISTS MEMBERS(M_ID INTEGER
PRIMARY KEY AUTOINCREMENT,NAME TEXT NOT NULL,PHONE_NO INTEGER,
DOB DATE,COURSE TEXT,BRANCH TEXT,YEAR INTEGER,POSITION TEXT,DOJ
DATE,C_ID INTEGER,FOREIGN KEY (C_ID)  REFERENCES COMMITTEE(C_ID))")
    #Inserting values to table
    conn.execute("INSERT INTO
MEMBERS(NAME,PHONE_NO,DOB,COURSE,BRANCH,YEAR,POSITION,DOJ,C_ID)
VALUES(?,?,?,?,?,?,?,?,?)",(name.get(),phoneNo.get(),birthdate.get(),course.get(),branch.get
(),year.get(),position.get(),doj.get(),cid.get()))
    conn.execute("INSERT INTO COMMITTEE_MEMBERS(M_ID,C_ID) VALUES
(?,?)",(conn.lastrowid,cid.get()))
    #Permanently saving the data to the database
    connection.commit()
    #Displaying the data on shell for convenince of user
    conn.execute("SELECT * FROM MEMBERS")
    for row in conn.fetchall():
        print(row)

def MembersGUI():
    global name,emailID,phoneNo,birthdate,course,branch,year,position,doj,cid
    #Setting window dimensions
    t=Tk()
    t.title("Committee Members")
    t.geometry("500x370")
    t.configure(bg="black")
    l1=Label(t,text="Committee Members Details",font="Times 20 bold
underline",bg="black",foreground="snow")
    l1.grid(row=0,column=0)

    #Defining LAbels
    Lname=Label(t,text="NAME",bg="black",foreground="snow",font="Baskerville 12
bold")
    LemailID=Label(t,text="EMAIL ID",bg="black",foreground="snow",font="Baskerville 12
bold")
    LphoneNo=Label(t,text="PHONE
NUMBER",bg="black",foreground="snow",font="Baskerville 12 bold")

Lbirthdate=Label(t,text="BIRTHDATE",bg="black",foreground="snow",font="Baskerville
12 bold")
    Lcourse=Label(t,text="COURSE",bg="black",foreground="snow",font="Baskerville 12
bold")
    Lbranch=Label(t,text="BRANCH",bg="black",foreground="snow",font="Baskerville 12
bold")
    Lyear=Label(t,text="YEAR",bg="black",foreground="snow",font="Baskerville 12 bold")
    Lposition=Label(t,text="POSITION",bg="black",foreground="snow",font="Baskerville 12
bold")
    Ldoj=Label(t,text="DATE OF
JOINING",bg="black",foreground="snow",font="Baskerville 12 bold")
    Lcid=Label(t,text="COMMITTEE ID",bg="black",foreground="snow",font="Baskerville
12 bold")

    #Defining Entry Boxes
    name=Entry(t)
    emailID=Entry(t)
    phoneNo=Entry(t)
    birthdate=Entry(t)
    course=Entry(t)
    branch=Entry(t)
    year=Entry(t)
    position=Entry(t)
    doj=Entry(t)
    cid=Entry(t)

    #Adding Labels to window
    Lname.grid(row=1,column=0)
    LemailID.grid(row=2,column=0)
    LphoneNo.grid(row=3,column=0)
    Lbirthdate.grid(row=5,column=0)
    Lcourse.grid(row=6,column=0)
    Lbranch.grid(row=7,column=0)
    Lyear.grid(row=8,column=0)
    Lposition.grid(row=9,column=0)
    Ldoj.grid(row=10,column=0)
    Lcid.grid(row=11,column=0)

    #Adding Entry box to window
    name.grid(row=1,column=1)
    emailID.grid(row=2,column=1)
    phoneNo.grid(row=3,column=1)
    birthdate.grid(row=5,column=1)
    course.grid(row=6,column=1)
    branch.grid(row=7,column=1)
    year.grid(row=8,column=1)
    position.grid(row=9,column=1)
    doj.grid(row=10,column=1)
    cid.grid(row=11,column=1)

    #Recieving and passing all the values

bSubmit=Button(t,text="Submit",command=creatingTableCommitteeMembers,font="Times
12 bold",activebackground="red")
    bSubmit.grid(row=12,column=1)

popularity page
from tkinter import *
import sqlite3
connection = sqlite3.connect("Event_Management.db")
cursor = connection.cursor()

def run():
    global t,ent,ent2
    count=0
    find=("SELECT COUNT(A_ID) FROM COLLEGE_AUDIENCE GROUP BY
COLLEGE_AUDIENCE.E_ID HAVING (COLLEGE_AUDIENCE.E_ID=(SELECT E_ID
FROM EVENTS WHERE NAME=? AND YEAR=?))")
    cursor.execute(find,[(ent.get()),(ent2.get())])
    val1=cursor.fetchone()
    val2=val1[0]
    count=count+val2
    find1=("SELECT COUNT(EA_ID) FROM EXTERNAL_AUDIENCE GROUP BY
EXTERNAL_AUDIENCE.E_ID HAVING (EXTERNAL_AUDIENCE.E_ID=(SELECT
E_ID FROM EVENTS WHERE NAME=? AND YEAR=?))")
    cursor.execute(find1,[(ent.get()),(ent2.get())])
    val3=cursor.fetchone()
    val4=val3[0]
    count=count+val4
    l5=Label(t,text="THE NUMBER OF PEOPLE WHO ATTENDED THE EVENT
ARE:",font="Times 15 bold",bg="black",foreground="snow")
    l5.pack()
    l6=Label(t,text=count,font="Times 15 bold",bg="black",foreground="snow")
    l6.pack()
```

```
def PopularityGUI():
    global t,ent,ent2
    t=Tk()
    t.configure(bg="black")
    t.title("Event Popularity")
    l1=Label(t,text="EVENT POPULARITY",font="Times 20 bold
underline",bg="black",foreground="snow")
    l1.pack()
    l2=Label(t,text="Here you can choose an event and see how many people registered for
that event!",bg="black",foreground="snow",font="Baskerville 12 bold")
    l2.pack()
    l3=Label(t,text="Enter the name of the
event",bg="black",foreground="snow",font="Baskerville 12 bold")
    l3.pack()
    ent=Entry(t)
    ent.pack()
    l4=Label(t,text="Select the year in which the event took
place",bg="black",foreground="snow",font="Baskerville 12 bold")
    l4.pack()
    ent2=Entry(t)
    ent2.pack()
    b1=Button(t,text="Find",command=run,font="Times 12 bold",activebackground="red")
    b1.pack()

remove member
from tkinter import *
import sqlite3
connection = sqlite3.connect("Event_Management.db")
cursor = connection.cursor()

def getId(val):
    global Id
    rem=val
    Id=val[0]

def Del():
    global e2,t,rowCount
    rem1=("DELETE FROM COMMITTEE_MEMBERS WHERE M_ID=?")
    cursor.execute(rem1,[(e2.get())])
    remov=("DELETE FROM MEMBERS WHERE M_ID=?")
    cursor.execute(remov,[(e2.get())])
    connection.commit()
    l8=Label(t,text="Member deleted from the
committee",bg="black",foreground="snow",font="Baskerville 12
bold").grid(row=rowCount,column=0)
    rowCount+=1
def disp():
    global e1,e2,Id,t,rowCount
    lid=Label(t,text="ID",bg="black",foreground="snow",font="Baskerville 12
bold").grid(row=rowCount,column=0)
```

```
    lname=Label(t,text="Name",bg="black",foreground="snow",font="Baskerville 12 
bold").grid(row=rowCount,column=1)
    lid=Label(t,text="Position",bg="black",foreground="snow",font="Baskerville 12 
bold").grid(row=rowCount,column=2)
    lid=Label(t,text="Date of joining",bg="black",foreground="snow",font="Baskerville 12 
bold").grid(row=rowCount,column=3)
    rowCount+=1
    sel=("SELECT M_ID,NAME,POSITION,DOJ FROM MEMBERS WHERE 
POSITION=? AND C_ID=?")
    cursor.execute(sel,[e1.get(),Id])
    print(e1.get())
    for row in cursor.fetchall():
        for i in range(0,3):
            l6=Label(t,text=row[i],bg="black",foreground="snow",font="Baskerville 12 
bold").grid(row=rowCount,column=i)
        rowCount+=1
        print(row)
    l7=Label(t,text="Enter the M_ID of the employee who you want to remove from the 
committee:",bg="black",foreground="snow",font="Baskerville 12 
bold").grid(row=rowCount,column=0)
    rowCount+=1
    e2=Entry(t)
    e2.grid(row=rowCount,column=0)
    rowCount+=1

    b6=Button(t,text="Delete",command=Del).grid(row=rowCount,column=0)
    rowCount+=1

def delYear():
    global e3,t,rowCount
    rem1=("DELETE FROM COMMITTEE_MEMBERS WHERE M_ID=?")
    cursor.execute(rem1,[(e3.get())])
    rem=("DELETE FROM MEMBERS WHERE M_ID=?")
    cursor.execute(rem,[(e3.get())])
    connection.commit()
    l8=Label(t,text="Member deleted from the 
committee",bg="black",foreground="snow",font="Baskerville 12 
bold").grid(row=rowCount,column=0)
    rowCount+=1

def dispYear():
    global vari,e3,Id,t,rowCount
    rowCount+=1
    lid=Label(t,text="ID",bg="black",foreground="snow",font="Baskerville 12 
bold").grid(row=rowCount,column=0)
    lname=Label(t,text="Name",bg="black",foreground="snow",font="Baskerville 12 
bold").grid(row=rowCount,column=1)
    lid=Label(t,text="Position",bg="black",foreground="snow",font="Baskerville 12 
bold").grid(row=rowCount,column=2)
```

```
    lid=Label(t,text="Date of joining",bg="black",foreground="snow",font="Baskerville 12
bold").grid(row=rowCount,column=3)
    rowCount+=1
    d=("SELECT M_ID,NAME,POSITION,DOJ FROM MEMBERS WHERE YEAR=?
AND C_ID=?")
    cursor.execute(d,[vari.get()],Id)
    for row in cursor.fetchall():
        for i in range(0,4):
            l9=Label(t,text=row[i],bg="black",foreground="snow",font="Baskerville 12
bold").grid(row=rowCount,column=i)
        rowCount+=1
    l7=Label(t,text="Enter the M_ID of the employee who you want to remove from the
committee:",bg="black",foreground="snow",font="Baskerville 12
bold").grid(row=rowCount,column=0)
    l7.pack()
    e3=Entry(t)
    e3.grid(row=rowCount,column=0)
    rowCount+=1
    b13=Button(t,text="Delete",command=delYear,font="Times 12
bold",activebackground="red").grid(row=rowCount,column=0)
    rowCount+=1

def deletePosition():
    global Id,t,e1,rowCount
    cnt=0
    l3=Label(t,text="Following positions are availabe for
deleting:",bg="black",foreground="snow",font="Baskerville 12
bold").grid(row=rowCount,column=0)
    rowCount+=1
    find=("SELECT DISTINCT(POSITION) FROM MEMBERS WHERE C_ID=?")
    cursor.execute(find,[(Id)])
    for row in cursor.fetchall():
        l4=Label(t,text=row,bg="black",foreground="snow",font="Baskerville 12
bold").grid(row=rowCount,column=0)
        rowCount+=1
    l5=Label(t,text="Please enter any one of the
positions:",bg="black",foreground="snow",font="Baskerville 12
bold").grid(row=rowCount,column=0)
    rowCount+=1
    e1=Entry(t)
    e1.grid(row=rowCount,column=0)
    rowCount+=1
    b5=Button(t,text="Proceed",command=disp,font="Times 12
bold",activebackground="red").grid(row=rowCount,column=0)
    rowCount+=1

    vari=IntVar()


def deleteYear():
    global vari,t,rowCount
    vari=IntVar()
    l9=Label(t,text="Enter the year from which you want to remove the employee
from:",bg="black",foreground="snow",font="Baskerville 12
bold").grid(row=rowCount,column=0)
    rowCount+=1
    b7=Radiobutton(t,text="1",variable=vari,value=1,font="Times 12
bold",bg="black",foreground="snow").grid(row=rowCount,column=0)
    rowCount+=1
    b8=Radiobutton(t,text="2",variable=vari,value=2,font="Times 12
bold",bg="black",foreground="snow").grid(row=rowCount,column=0)
    rowCount+=1
    b9=Radiobutton(t,text="3",variable=vari,value=3,font="Times 12
bold",bg="black",foreground="snow").grid(row=rowCount,column=0)
    rowCount+=1
    b10=Radiobutton(t,text="4",variable=vari,value=4,font="Times 12
bold",bg="black",foreground="snow").grid(row=rowCount,column=0)
    rowCount+=1
    b11=Radiobutton(t,text="5",variable=vari,value=5,font="Times 12
bold",bg="black",foreground="snow").grid(row=rowCount,column=0)
    rowCount+=1
    b12=Button(t,text="Submit",command=dispYear,font="Times 12
bold",activebackground="red").grid(row=rowCount,column=0)
    rowCount+=1


def RemoveMemberGUI():
    global t,var,rowCount
    rowCount=5
    t=Tk()
    t.configure(bg="black")
    l1=Label(t,text="REMOVE COMMITTEE
MEMBERS",bg="black",foreground="snow",font="Baskerville 12 bold underline")
    l1.grid(row=0,column=0)
    l2=Label(t,text="Please select how on what basis you want to remove the committee
member:",bg="black",foreground="snow",font="Baskerville 12 bold")
    l2.grid(row=1,column=0)
    var=IntVar()


    # Adding File Menu and commands

    b1=Button(t,text="Position",command=deletePosition,font="Times 12
bold",activebackground="red",bg="black",foreground="snow")
    b1.grid(row=2,column=0)
    b2=Button(t,text="Year",command=deleteYear,font="Times 12
bold",activebackground="red",bg="black",foreground="snow")
    b2.grid(row=3,column=0)
```

```
resource
from tkinter import *
import sqlite3
connection = sqlite3.connect("Event_Management.db")
conn= connection.cursor()


def creatingTableResources():
   global r_name,r_price,provider_name,provider_phone,provider_add
   #Creating table if it doesn't exits
   conn.execute("CREATE TABLE IF NOT EXISTS RESOURCES(R_ID INTEGER
PRIMARY KEY AUTOINCREMENT,NAME TEXT,PRICE
NUMBER,PROVIDER_NAME VARCHAR,PROVIDER_PHONE_NO
INTEGER,PROVIDER_ADDRESS VARCHAR)")
   connection.commit()
   #Inserting values into the table
   conn.execute("INSERT INTO
RESOURCES(NAME,PRICE,PROVIDER_NAME,PROVIDER_PHONE_NO,PROVIDER
_ADDRESS)
VALUES(?,?,?,?,?)",(r_name.get(),r_price.get(),provider_name.get(),provider_phone.get(),pr
ovider_add.get()))
   connection.commit()
   #Permanantly making changes to the database
   conn.execute("SELECT * FROM RESOURCES")
   #Displaying the data for the user's convenience
   for row in conn.fetchall():
      print(row)

def ResourceGUI():
   global r_name,r_price,provider_name,provider_phone,provider_add
   #Setting window properties
   t=Tk()
   t.title("Resources")
   t.geometry("400x300")
   t.configure(bg="black")
   l1=Label(t,text="Resources Details",font="Times 20 bold
underline",bg="black",foreground="snow")
   l1.grid(row=0,column=0)
   #Defining Labels
   Lname=Label(t,text="NAME OF
RESOURCE",bg="black",foreground="snow",font="Baskerville 12 bold")
   Lprice=Label(t,text="PRICE",bg="black",foreground="snow",font="Baskerville 12 bold")
   Lpname=Label(t,text="PROVIDER
NAME",bg="black",foreground="snow",font="Baskerville 12 bold")
   Lpphno=Label(t,text="PROVIDER PHONE
NO",bg="black",foreground="snow",font="Baskerville 12 bold")
   Lpadd=Label(t,text="PROVIDER
ADDRESS",bg="black",foreground="snow",font="Baskerville 12 bold")

   #Defining Entry Boxes
```

```
    r_name=Entry(t)
    r_price=Entry(t)
    provider_name=Entry(t)
    provider_phone=Entry(t)
    provider_add=Entry(t)

    #Adding Labels to window
    Lname.grid(row=1,column=0)
    Lprice.grid(row=3,column=0)
    Lpname.grid(row=5,column=0)
    Lpphno.grid(row=6,column=0)
    Lpadd.grid(row=7,column=0)


    #Adding Entry box to window
    r_name.grid(row=1,column=1)
    r_price.grid(row=3,column=1)
    provider_name.grid(row=5,column=1)
    provider_phone.grid(row=6,column=1)
    provider_add.grid(row=7,column=1)


    #Recieving and passing all the values
    bSubmit=Button(t,text="Submit",command=creatingTableResources,font="Times 12
bold",activebackground="red")
    bSubmit.grid(row=11,column=1)
    connection.commit()


resource event
from tkinter import *
import sqlite3
connection = sqlite3.connect("Event_Management.db")
conn= connection.cursor()


def creatingTableResource_Event():
    conn.execute("CREATE TABLE IF NOT EXISTS RESOURCE_EVENT(R_ID
NUMBER REFERENCES RESOURCES(R_ID), E_ID NUMBER REFERENCES
EVENTS(E_ID),QUANTITY NUMBER)")
    connection.commit()
    conn.execute("INSERT INTO RESOURCE_EVENT
VALUES(?,?,?)",(r_id.get(),e_id.get(),quantity.get()))
    connection.commit()
    conn.execute("SELECT * FROM RESOURCE_EVENT")
    for row in conn.fetchall():
        print(row)


def Resource_EventGUI():
    global r_id,e_id,quantity
    t=Tk()
    t.title("Resources for Event")
    t.geometry("500x200")
    t.configure(bg="black")
    l1=Label(t,text="Resources Event Details",font="Times 20 bold
underline",bg="black",foreground="snow")
    l1.grid(row=0,column=0)

    #Defining Labels
    Lr_id=Label(t,text="RESOURCE ID",bg="black",foreground="snow",font="Baskerville
12 bold")
    Le_id=Label(t,text="EVENT ID",bg="black",foreground="snow",font="Baskerville 12
bold")
    Lquantity=Label(t,text="QUANTITY
REQUIRED",bg="black",foreground="snow",font="Baskerville 12 bold")


    #Defining Entry Boxes
    r_id=Entry(t)
    e_id=Entry(t)
    quantity=Entry(t)

    #Adding Labels to window
    Lr_id.grid(row=1,column=0)
    Le_id.grid(row=2,column=0)
    Lquantity.grid(row=3,column=0)

    #Adding Entry box to window
    r_id.grid(row=1,column=1)
    e_id.grid(row=2,column=1)
    quantity.grid(row=3,column=1)

    #Recieving and passing all the values
    bSubmit=Button(t,text="Submit",command=creatingTableResource_Event,font="Times
12 bold",activebackground="red")
    bSubmit.grid(row=4,column=1)
    connection.commit()


from tkinter import *
from tkinter.ttk import *
from time import strftime

import sqlite3
connection = sqlite3.connect("Event_Management.db")
cursor= connection.cursor()


#free of charge events
```

```
def free():
    global t
    cursor.execute("CREATE TEMP VIEW SEARCH AS SELECT
E_ID,NAME,TOPIC,EVENT_DATE,WEBSITE,CHARGE,TYPE FROM EVENTS")
    find=("SELECT * FROM SEARCH WHERE CHARGE=0")
    cursor.execute(find)
    a=2
    for row in cursor.fetchall():
        l4=Label(t,text=row,background="black",foreground="snow",font="Baskerville 12
bold")
        l4.grid(row=a,column=0,sticky=W)
        a+=1

#price mentioned
def charge_price():
    global t
    global e1
    cursor.execute("CREATE TEMP VIEW SEARCH AS SELECT
E_ID,C_ID,NAME,TOPIC,EVENT_DATE,WEBSITE,CHARGE,TYPE FROM EVENTS")
    find=("SELECT * FROM SEARCH WHERE CHARGE=?")
    cursor.execute(find,[(e1.get())])
    c=4
    for row in cursor.fetchall():
        l3=Label(t,text=row,background="black",foreground="snow",font="Baskerville 12
bold")
        l3.grid(row=c,column=0,sticky=W)
        c+=1

#entering price desired
def enter():
    global t
    l2=Label(t,text="Enter price of
event",background="black",foreground="snow",font="Baskerville 12 bold")
    l2.grid(row=2,column=0)
    global e1
    e1=Entry(t)
    e1.grid(row=2,column=1)
    b=Button(t,text="Submit",command=charge_price)
    b.grid(row=3,column=0,sticky=W)

#any price
def any():
    global t
    cursor.execute("CREATE TEMP VIEW SEARCH AS SELECT
E_ID,C_ID,NAME,TOPIC,EVENT_DATE,WEBSITE,CHARGE,TYPE FROM EVENTS")
    find=("SELECT * FROM SEARCH ORDER BY CHARGE ASC")
    cursor.execute(find)
    y=2
    for row in cursor.fetchall():
        l5=Label(t,text=row,background="black",foreground="snow",font="Baskerville 12 
bold")
        l5.grid(row=y,column=0,sticky=W)
        y+=1

def com_wise():
    global t
    global e2
    cursor.execute("CREATE TEMP VIEW SEARCH AS SELECT 
E_ID,C_ID,NAME,TOPIC,EVENT_DATE,WEBSITE,CHARGE,TYPE FROM EVENTS")
    find=("SELECT * FROM SEARCH WHERE C_ID=?")
    cursor.execute(find,[(e2.get())])
    b=4
    for row in cursor.fetchall():
        l7=Label(t,text=row,background="black",foreground="snow",font="Baskerville 12 
bold")
        l7.grid(row=b,column=0,sticky=W)
        b+=1

#committeee wise search for events
def enter2():
    global t
    l6=Label(t,text="Enter Committee id to be searched",font="Times 15 
bold",background="black",foreground="snow")
    l6.grid(row=2,column=0)
    global e2
    e2=Entry(t)
    e2.grid(row=2,column=1)
    b=Button(t,text="Submit",command=com_wise)
    b.grid(row=3,column=0)
    com_wise()

def searchGUI():
    global t
    t=Tk()
    t.title("Search")
    t.configure(bg="black")
    t.geometry("400x100")
    l1=Label(t,text="Welcome to our Search Station",font="Times 20 bold 
underline",background="black",foreground="snow")
    l1.grid(row=0,column=0)

    menubar=Menu(t)
    #adding price range
    price=Menu(menubar,tearoff=0)
    menubar.add_cascade(label='Charges of event',menu=price)
    price.add_command(label='Free of cost',command=free)
    price.add_separator()
    price.add_command(label='Specific Price',command=enter)
    price.add_separator()
    price.add_command(label='Any price',command=any)

    #adding committee wise search
    com=Menu(menubar,tearoff=0)
    menubar.add_cascade(label='Committee Events',menu=com)
    com.add_command(label='Search Committee',command=enter2)
    com.add_separator()

    t.config(menu= menubar)
    t.mainloop()
sort
from tkinter import *
import sqlite3
connection = sqlite3.connect("Event_Management.db")
cursor= connection.cursor()

def enter1():
    global t
    global e1
    lab=Label(t,text="Enter which year's past events you wish to look
into:",bg="black",foreground="snow",font="Baskerville 12 bold")
    lab.grid(row=1,column=0)
    e1=Entry(t)
    e1.grid(row=1,column=1)
    b=Button(t,text="Submit",font="Times 10 bold",activebackground="red",command=past)
    b.grid(row=2,column=0)

def enter2():
    global t
    global e1
    lab=Label(t,text="Enter which year's future events you wish to look
into:",bg="black",foreground="snow",font="Baskerville 12 bold")
    lab.grid(row=1,column=0)
    e1=Entry(t)
    e1.grid(row=1,column=1)
    b=Button(t,text="Submit",font="Times 10
bold",activebackground="red",command=future)
    b.grid(row=2,column=0)

#past events
def past():
    global t
    global e1
    cursor.execute("CREATE TEMP VIEW SORT AS SELECT
E_ID,NAME,TOPIC,EVENT_DATE,WEBSITE,CHARGE,TYPE,YEAR FROM
EVENTS")
    find=("SELECT * FROM SORT WHERE YEAR < ? ORDER BY YEAR DESC")
    cursor.execute(find,[(e1.get())])
    a=3
    for row in cursor.fetchall():
        l5=Label(t,text=row,bg="black",foreground="snow",font="Baskerville 12 bold")
        l5.grid(row=a,column=0,sticky=W)
        a+=1

#upcoming events
def future():
    global t
    global e1
    cursor.execute("CREATE TEMP VIEW SORT AS SELECT
E_ID,NAME,TOPIC,EVENT_DATE,WEBSITE,CHARGE,TYPE,YEAR FROM
EVENTS")
    find=("SELECT * FROM SORT WHERE YEAR > ? ORDER BY YEAR ASC")
    cursor.execute(find,[(e1.get())])
    a=3
    for row in cursor.fetchall():
        l6=Label(t,text=row,bg="black",foreground="snow",font="Baskerville 12 bold")
        l6.grid(row=a,column=0,sticky=W)
        a+=1
def sortGUI():
    global t
    t=Tk()
    t.title("Sort")
    t.geometry("500x100")
    t.configure(bg="black")
    l1=Label(t,text="Welcome to our Sort Station",font="Times 20 bold
underline",bg="black",foreground="snow")
    l1.grid(row=0,column=0)

    menubar=Menu(t)
    #adding event of which committee
    sort_event=Menu(menubar)
    menubar.add_cascade(label='Sort Events',menu=sort_event)
    sort_event.add_command(label='Past Events',command=enter1)
    sort_event.add_separator()
    sort_event.add_command(label='Future Events',command=enter2)
    t.config(menu=menubar)
    t.mainloop()

sponsor count
from tkinter import *
import sqlite3
connection = sqlite3.connect("Event_Management.db")
cursor = connection.cursor()
def dispEvent(ID):
    global rowCount,e1
    a1=Label(t,text="EVENTS SPONSORED BY THIS SPONSOR ARE:
").grid(row=rowCount,column=0)
    rowCount+=1
    a2=Label(t,text="Name",bg="black",foreground="snow",font="Baskerville 12
bold").grid(row=rowCount,column=0)
```

```
    a3=Label(t,text="Topic",bg="black",foreground="snow",font="Baskerville 12
bold").grid(row=rowCount,column=1)
    a4=Label(t,text="Year",bg="black",foreground="snow",font="Baskerville 12
bold").grid(row=rowCount,column=2)
    a5=Label(t,text="Type",bg="black",foreground="snow",font="Baskerville 12
bold").grid(row=rowCount,column=3)
    rowCount+=1
    find=("SELECT NAME,TOPIC,YEAR,TYPE FROM EVENTS WHERE E_ID=(SELECT
E_ID FROM SPONSOR_EVENT WHERE S_ID=?)")
    cursor.execute(find,[ID])
    for row in cursor.fetchall():
        for i in range(0,4):
            l5=Label(t,text=row[i],bg="black",foreground="snow",font="Baskerville 12
bold").grid(row=rowCount,column=i)
        rowCount=+1
def name():
    global t,rowCount,e1
    l3=Label(t,text="Following are the sponsor names, please select a sponsor:
",bg="black",foreground="snow",font="Baskerville 12
bold").grid(row=rowCount,column=0)

    rowCount+=1
    lid=Label(t,text="Id").grid(row=rowCount,column=0)
    rowCount+=1
    lid=Label(t,text="Name").grid(row=rowCount,column=1)
    cursor.execute("SELECT S_ID,NAME FROM SPONSORS WHERE S_ID IN (SELECT
S_ID FROM SPONSOR_EVENT)")
    for row in cursor.fetchall():
        for i in range(0,2):
            l4=Label(t,text=row[i],bg="black",foreground="snow",font="Baskerville 12
bold").grid(row=rowCount,column=i)
        rowCount+=1
    l5=Label(t,text="Enter the id of the sponsor whose event sponsored you want:
",bg="black",foreground="snow",font="Baskerville 12
bold").grid(row=rowCount,column=0)
    rowCount+=1
    e1=Entry(t)
    e1.grid(row=rowCount,column=0)
    rowCount+=1
    b1=Button(t,text="Display",command=lambda: dispEvent(e1.get()),font="Times 12
bold",activebackground="red").grid(row=rowCount,column=0)
    rowCount+=1

def type():
    global t,rowCount
    l3=Label(t,text="Following are the sponsor types, please select a sponsor:
",bg="black",foreground="snow",font="Baskerville 12
bold").grid(row=rowCount,column=0)
    rowCount+=1
    cursor.execute("SELECT S_ID,TYPE FROM SPONSORS WHERE S_ID IN (SELECT
S_ID FROM SPONSOR_EVENT)")
    for row in cursor.fetchall():
        for i in range(0,2):
            l4=Label(t,text=row[i]bg="black",foreground="snow",font="Baskerville 12
bold").grid(row=rowCount,column=i)
        rowCount+=1

    l5=Label(t,text="Enter the id of the sponsor whose event sponsored you want:
",bg="black",foreground="snow",font="Baskerville 12
bold").grid(row=rowCount,column=0)
    rowCount+=1
    e1=Entry(t)
    e1.grid(row=rowCount,column=0)
    rowCount+=1
    b1=Button(t,text="Display",command=lambda:dispEvent(e1.get()),font="Times 12
bold",activebackground="red").grid(row=rowCount,column=0)
    rowCount+=1



def SponsorCountGUI():
    global t,rowCount
    rowCount=0
    t=Tk()
    t.title("Sponsor Details")
    t.configure(bg="black")
    l1=Label(t,text="This page will show you a sponsor has sponsored how many events till
now!",font="Times 20 bold
underline",bg="black",foreground="snow").grid(row=rowCount,column=0)
    rowCount+=1
    l2=Label(t,text="Please select the basis on which you want to find your
sponsor:",bg="black",foreground="snow",font="Baskerville 12
bold").grid(row=rowCount,column=0)
    rowCount+=1
    b1=Button(t,text="Sponsor Name",command=name,font="Times 12
bold",activebackground="red").grid(row=rowCount,column=0)
    rowCount+=1
    b2=Button(t,text="Sponsor Type",command=type,font="Times 12
bold",activebackground="red").grid(row=rowCount,column=0)
    rowCount+=1

sponsor event
from tkinter import *
import sqlite3
connection = sqlite3.connect("Event_Management.db")
conn= connection.cursor()


def creatingTableSponsor_Event():
    print("in")
    conn.execute("CREATE TABLE IF NOT EXISTS SPONSOR_EVENT(S_ID NUMBER
REFERENCES SPONSORS(S_ID), E_ID NUMBER REFERENCES EVENTS(E_ID))")
    connection.commit()
    conn.execute("INSERT INTO SPONSOR_EVENT VALUES(?,?)",(s_id.get(),e_id.get()))
    connection.commit()
    conn.execute("SELECT * FROM SPONSOR_EVENT")
    for row in conn.fetchall():
        print(row)


def Sponsor_EventGUI():
    global s_id,e_id
    t=Tk()
    t.title("Sponsors for Event")
    t.geometry("450x160")
    t.configure(bg="black")
    l1=Label(t,text="Sponsors Event Details",font="Times 20 bold
underline",bg="black",foreground="snow")
    l1.grid(row=0,column=0)

    #Defining Labels
    Ls_id=Label(t,text="SPONSOR ID",bg="black",foreground="snow",font="Baskerville 12
bold")
    Le_id=Label(t,text="EVENT ID",bg="black",foreground="snow",font="Baskerville 12
bold")

    #Defining Entry Boxes
    s_id=Entry(t)
    e_id=Entry(t)

    #Adding Labels to window
    Ls_id.grid(row=1,column=0)
    Le_id.grid(row=2,column=0)

    #Adding Entry box to window
    s_id.grid(row=1,column=1)
    e_id.grid(row=2,column=1)

    ls=Label(t,text="The available event id and event name are:",font="Baskerville 12
bold",bg="black",foreground="snow")
    ls.grid(row=4,column=0)
    a1=Label(t,text="EVENT ID",font="Baskerville 12
bold",bg="black",foreground="snow").grid(row=5,column=0)
    a2=Label(t,text="EVENT NAME",font="Baskerville 12
bold",bg="black",foreground="snow").grid(row=5,column=1)

    find2=("SELECT E_ID,NAME FROM EVENTS")
    conn.execute(find2)
    c=6
    c1=0
    for row in conn.fetchall():
        for i in range(0,2):
            L1=Label(t,text=row[i],bg="black",foreground="snow",font="Baskerville 12 bold")
            L1.grid(row=c,column=c1)
            c1=c1+1
        c=0
        c+=1
    lss=Label(t,text="The available sponsor id and sponsor name are:",font="Baskerville 12
bold",bg="black",foreground="snow")
    lss.grid(row=c,column=0)
    a1=Label(t,text="SPONSOR ID",font="Baskerville 12
bold",bg="black",foreground="snow").grid(row=c+1,column=0)
    a2=Label(t,text="SPONSOR NAME",font="Baskerville 12
bold",bg="black",foreground="snow").grid(row=c+2,column=1)
    find3=("SELECT S_ID,NAME FROM SPONSORS")
    conn.execute(find3)
    k=c+3
    c1=0
    for row in conn.fetchall():
        for i in range(0,2):
            L1=Label(t,text=row[i],bg="black",foreground="snow",font="Baskerville 12 bold")
            L1.grid(row=k,column=c1)
            c1=c1+1
        c1=0
        k+=1
    #Recieving and passing all the values
    bSubmit=Button(t,text="Submit",command=creatingTableSponsor_Event,font="Times 12
bold",activebackground="red")
    bSubmit.grid(row=k,column=1)
    connection.commit()
sponsors
from tkinter import *
import sqlite3
connection = sqlite3.connect("Event_Management.db")
conn= connection.cursor()


def creatingTableSponsors():
    global s_name,person_contacted,_s_emailID,s_phoneNo,s_add,event_no,past_event_no
    #Creating table if it doesn't exits
    conn.execute("CREATE TABLE IF NOT EXISTS SPONSORS(S_ID INTEGER
PRIMARY KEY AUTOINCREMENT,NAME TEXT NOT NULL,PERSON_CONTACTED
TEXT,EMAIL_ID VARCHAR NOT NULL,PHONE_NO INTEGER NOT
NULL,WEBSITE VARCHAR NOT NULL,TYPE TEXT)")
    connection.commit()
    #Inserting values into the table
    conn.execute("INSERT INTO
SPONSORS(NAME,PERSON_CONTACTED,EMAIL_ID,PHONE_NO,WEBSITE,TYPE)
VALUES(?,?,?,?,?,?)",(s_name.get(),person_contacted.get(),s_emailID.get(),s_phoneNo.get()
,s_website.get(),s_type.get()))
    connection.commit()
    #Permanantly making changes to the database
    connection.commit()
    #Displaying the data for the user's convenience
    conn.execute("SELECT * FROM SPONSORS")
    for row in conn.fetchall():
        print(row)

def SponsorsGUI():
    global s_name,person_contacted,s_emailID,s_phoneNo,s_website,s_type
    #Setting window properties
    t=Tk()
    t.title("Sponsors")
    t.geometry("400x250")
    t.configure(bg="black")
    l1=Label(t,text="Sponsors Details",font="Times 20 bold
underline",bg="black",foreground="snow")
    l1.grid(row=0,column=0)

    #Defining LAbels
    Lsname=Label(t,text="SPONSOR
NAME",bg="black",foreground="snow",font="Baskerville 12 bold")
    Lpc=Label(t,text="PERSON
CONTACTED",bg="black",foreground="snow",font="Baskerville 12 bold")
    LemailID=Label(t,text="EMAIL ID",bg="black",foreground="snow",font="Baskerville 12
bold")
    LphoneNo=Label(t,text="PHONE
NUMBER",bg="black",foreground="snow",font="Baskerville 12 bold")
    Lweb=Label(t,text="WEBSITE",bg="black",foreground="snow",font="Baskerville 12
bold")
    Ltype=Label(t,text="SPONSOR TYPE",bg="black",foreground="snow",font="Baskerville
12 bold")

    #Defining Entry Boxes
    s_name=Entry(t)
    person_contacted=Entry(t)
    s_emailID=Entry(t)
    s_phoneNo=Entry(t)
    s_website=Entry(t)
    s_type=Entry(t)

    #Adding Labels to window
    Lsname.grid(row=2,column=0)
    Lpc.grid(row=3,column=0)
    LemailID.grid(row=4,column=0)
    LphoneNo.grid(row=5,column=0)
    Lweb.grid(row=6,column=0)
    Ltype.grid(row=7,column=0)
    #Adding Entry box to window
    s_name.grid(row=2,column=1)
    person_contacted.grid(row=3,column=1)
    s_emailID.grid(row=4,column=1)
    s_phoneNo.grid(row=5,column=1)
    s_website.grid(row=6,column=1)
    s_type.grid(row=7,column=1)


    #Recieving and passing all the values
    bSubmit=Button(t,text="Submit",command=creatingTableSponsors,font="Times 12
bold",activebackground="red")
    bSubmit.grid(row=11,column=1)
    connection.commit()
total cost
from tkinter import *
import sqlite3
connection = sqlite3.connect("Event_Management.db")
cursor= connection.cursor()
def destructTotalCost(window):
    window.destroy()
    TotalCostGUI()

def getID(val):
    global Id
    rem=val
    Id=val[0]

def calculate():
    global t,e1,rowCount
    spent=0
    rec=0
    f1=("SELECT SUM(CHARGE) FROM GUEST_EVENT WHERE E_ID=?")
    cursor.execute(f1,[(e1.get())])
    val=cursor.fetchone()
    val1=val[0]
    spent=spent+val1
    f2=("SELECT R_ID,QUANTITY FROM RESOURCE_EVENT WHERE E_ID=?")
    cursor.execute(f2,[(e1.get())])
    val2=cursor.fetchall()
    for row in val2:
        f3=("SELECT PRICE FROM RESOURCES WHERE R_ID=?")
        cursor.execute(f3,[(row[0])])
        val3=cursor.fetchone()
        val4=val3[0]
        spent=spent+(val4*row[1])
    f4=("SELECT SUM(MONEY_PAID) FROM COLLEGE_AUDIENCE WHERE
E_ID=?")
    cursor.execute(f4,[(e1.get())])
    val5=cursor.fetchone()
    val6=val5[0]
    rec=rec+val6
    f5=("SELECT SUM(MONEY_PAID) FROM EXTERNAL_AUDIENCE WHERE
E_ID=?")
    cursor.execute(f5,[(e1.get())])
    val7=cursor.fetchone()
    if val7 is not None:
        val8=val7[0]
        rec=rec+val8
    totalspent=spent-rec
    f6=("SELECT BUDGET FROM EVENTS WHERE E_ID=?")
    cursor.execute(f6,[(e1.get())])
    val9=cursor.fetchone()
    val10=val9[0]
    f9=("SELECT C_ID FROM EVENTS WHERE E_ID=?")
    cursor.execute(f9,[(e1.get())])
    val13=cursor.fetchone()
    cid=val13[0]
    f7=("SELECT FUNDS FROM COMMITTEE WHERE C_ID=?")
    cursor.execute(f7,[(cid)])
    val11=cursor.fetchone()
    val12=val11[0]
    saved=val10-totalspent
    finBudget=val12+saved
    f8=("UPDATE COMMITTEE SET FUNDS=? WHERE C_ID=?")
    cursor.execute(f8,[(finBudget),(cid)])
    str1="THE TOTAL MONEY SPENT FOR THIS EVENT(RESOURCE + GUEST
CHARGERS) IS: RS. "+str(spent)
    str2="THE TOTAL MONEY RECIEVED FROM AUDIENCE: RS. "+str(rec)
    str3="MONEY REMAINING IN EVENT BUDGET: RS. "+str(saved)
    str5="THE UPDATED FUNDS OF THE COMMITTEE IS NOW : RS. "+str(finBudget)
    str4="CURRENT ORIGINAL FUNDS OF THE COMMITTEE: RS. "+str(val12)
    rowCount=rowCount+1
    l2=Label(t,text=str1,font="Times 12
bold",bg="black",foreground="snow").grid(row=rowCount,column=0)
    rowCount=rowCount+1
    l3=Label(t,text=str2,font="Times 12
bold",bg="black",foreground="snow").grid(row=rowCount,column=0)
    rowCount=rowCount+1
    l4=Label(t,text=str3,font="Times 12
bold",bg="black",foreground="snow").grid(row=rowCount,column=0)
    rowCount=rowCount+1
    l5=Label(t,text=str4,font="Times 12
bold",bg="black",foreground="snow").grid(row=rowCount,column=0)
    rowCount=rowCount+1
    l6=Label(t,text=str5,font="Times 12
bold",bg="black",foreground="snow").grid(row=rowCount,column=0)
    rowCount=rowCount+1
```

```
def TotalCostGUI():
    global e1,t,Id,rowCount
    t=Tk()
    rowCount=4
    t.configure(bg="black")
    t.title("Calculating Total Cost")
    l=Label(t,text="HERE YOU CAN DO THE COST EVALUATION OF THE
EVENT!",font="Times 16 bold",bg="black",foreground="snow")
    l.grid(row=0,column=0)
    la=Label(t,text="The event available for your committee are: ",font="Times 12
italic",bg="black",foreground="snow").grid(row=1,column=0)
    lid=Label(t,text="ID",font="Times 12
italic",bg="black",foreground="snow").grid(row=2,column=0)
    lname=Label(t,text="Name",font="Times 12
italic",bg="black",foreground="snow").grid(row=2,column=1)
    f=("SELECT E_ID,NAME FROM EVENTS WHERE C_ID=?")
    cursor.execute(f,[(Id)])
    for row in cursor.fetchall():
        for i in range(0,2):
            lt=Label(t,text=row[i],font="Times 12
bold",bg="black",foreground="snow").grid(row=rowCount,column=i)
        rowCount=rowCount+1
    rowCount=rowCount+1
    l1=Label(t,text="Enter the event id whose cost is to be calculated: ",font="Times 15
italic",bg="black",foreground="snow").grid(row=rowCount,column=0)
    rowCount=rowCount+1
    e1=Entry(t)
    e1.grid(row=rowCount,column=0)
    rowCount=rowCount+1
    b1=Button(t,text="Submit",font="Times
10",activebackground="red",command=calculate).grid(row=rowCount,column=0)
    rowCount=rowCount+1



 upcoming
from tkinter import *
import sqlite3
connection = sqlite3.connect("Event_Management.db")
cursor= connection.cursor()


e1=StringVar
def ue():
    global t
    global e1
    cursor.execute("CREATE TEMP VIEW UPCOMING AS SELECT
E_ID,C_ID,NAME,TOPIC,EVENT_DATE,WEBSITE,CHARGE,TYPE,YEAR FROM
EVENTS")
    find2=("SELECT * FROM UPCOMING WHERE YEAR > ? ORDER BY YEAR ASC")
    cursor.execute(find2,[(e1.get())])
    c=3
    #cursor.execute("SELECT *,MONTHS_BETWEEN(x,y) FROM EVENTS")
    for row in cursor.fetchall():
        l3=Label(t,text=row,font="Times 15 bold",bg="black",foreground="snow")
        l3.grid(row=c,column=0,sticky=W)
        c+=1

def upcomingGUI():
    global t
    global e1
    t=Tk()
    t.title("Upcoming events")
    t.geometry("600x500")
    t.configure(bg="black")
    l1=Label(t,text="Upcoming Events!",font="Times 20 bold 
underline",bg="black",foreground="snow")
    l1.grid(row=0,column=0)
    #lets display the coming up events
    l2=Label(t,text="Enter the year for which you wish to see upcoming 
events",bg="black",foreground="snow",font="Baskerville 12 bold")
    l2.grid(row=1,column=0)
    e1=Entry(t)
    e1.grid(row=1,column=1)
    b1=Button(t,text="Submit",command=ue,font="Times 10 bold",activebackground="red")
    b1.grid(row=2,column=0)
```