**DBMS - Mini Project**

**Event Management System**

Submitted By:

Name: K S Ramalakshmi

SRN: PES1UG20CS186

V Semester Section D

**Short Description and Scope of the Project**

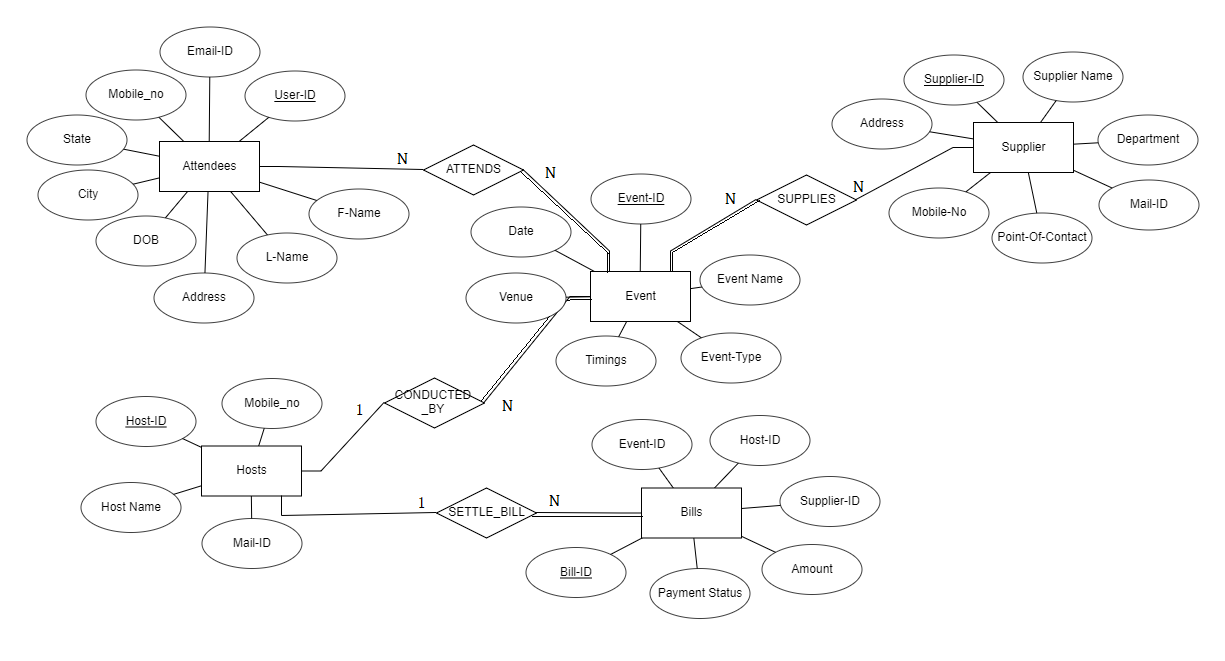
**What is an Event Management System?**

Data organisation is a very essential skill to ensure effective customer service and business improvements. A simple full stack web application to keep track of ongoing events, attendee details, host details and event organisation records. Target Users who include the event managers, and event hosts can use the system efficiently to keep track of events and aid in better organisation of the same.

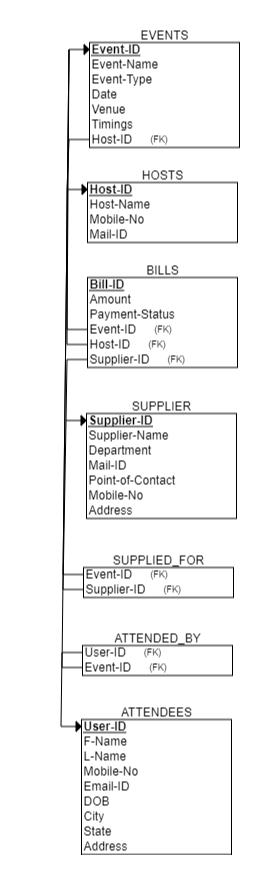
Scope of the Project

Currently the project accepts records, displays on request, updates specific fields for specific records and deletes records on request. The existing CRUD operations can be made more dynamic and optimized. Increased views to display user and event statistics can be included to provide end-to-end features of a data management platform.

ER Diagram



**Relational Schema**



**DDL statements - Building the database**

CREATE TABLE IF NOT EXISTS HOSTS(

    HOST\_ID INT AUTO\_INCREMENT NOT NULL,

    HOST\_NAME VARCHAR(255),

    MOBILE\_NUMBER VARCHAR(10),

    MAIL\_ID VARCHAR(25),

    PRIMARY KEY (HOST\_ID));

CREATE TABLE IF NOT EXISTS EVENTS(

    EVENT\_ID INT AUTO\_INCREMENT NOT NULL,

    EVENT\_NAME VARCHAR(255),

    EVENT\_TYPE VARCHAR(255),

    EVENT\_DATE\_START DATE,

    EVENT\_DATE\_END DATE,

    EVENT\_TIME\_START TIME,

    EVENT\_TIME\_END TIME,

    VENUE VARCHAR(255),

    HOST\_ID INT,

    PRIMARY KEY (EVENT\_ID),

    FOREIGN KEY (HOST\_ID) REFERENCES HOSTS(HOST\_ID));

CREATE TABLE IF NOT EXISTS SUPPLIER(

    SUPPLIER\_ID INT AUTO\_INCREMENT NOT NULL,

    SUPPLIER\_NAME VARCHAR(255),

    DEPARTMENT VARCHAR(255),

    MAIL\_ID VARCHAR(255),

    POINT\_OF\_CONTACT VARCHAR(255),

    MOBILE\_NO VARCHAR(25),

    ADDRESS VARCHAR(255),

    PRIMARY KEY (SUPPLIER\_ID));

CREATE TABLE IF NOT EXISTS BILLS(

    BILL\_ID INT NOT NULL,

    DEALER VARCHAR(255),

    AMOUNT INT,

    PAYMENT\_STATUS VARCHAR(25),

    EVENT\_ID INT,

    HOST\_ID INT,

    SUPPLIER\_ID INT,

    PRIMARY KEY (BILL\_ID),

    FOREIGN KEY (EVENT\_ID) REFERENCES EVENTS(EVENT\_ID),

    FOREIGN KEY (HOST\_ID) REFERENCES HOSTS(HOST\_ID),

    FOREIGN KEY (SUPPLIER\_ID) REFERENCES SUPPLIER(SUPPLIER\_ID));

CREATE TABLE IF NOT EXISTS SUPPLIED\_FOR(

    EVENT\_ID INT,

    SUPPLIER\_ID INT,

    FOREIGN KEY (EVENT\_ID) REFERENCES EVENTS(EVENT\_ID),

    FOREIGN KEY (SUPPLIER\_ID) REFERENCES SUPPLIER(SUPPLIER\_ID));

CREATE TABLE IF NOT EXISTS ATTENDEES(

    USER\_ID INT NOT NULL AUTO\_INCREMENT,

    F\_NAME VARCHAR(255),

    L\_NAME VARCHAR(255),

    MOBILE\_NUMBER INT,

    MAIL\_ID VARCHAR(255),

    DOB DATE,

    CITY VARCHAR(255),

    STATE VARCHAR(255),

    ADDRESS VARCHAR(255),

    PRIMARY KEY (USER\_ID));

CREATE TABLE IF NOT EXISTS ATTENDED\_BY(

    USER\_ID INT,

    EVENT\_ID INT,

    FOREIGN KEY (USER\_ID) REFERENCES ATTENDEES(USER\_ID),

    FOREIGN KEY (EVENT\_ID) REFERENCES EVENTS(EVENT\_ID));

Alternatively, the SQL Query box in the GUI can also be used to execute SQL Queries to create tables from the frontend itself.

**Populating the Database**

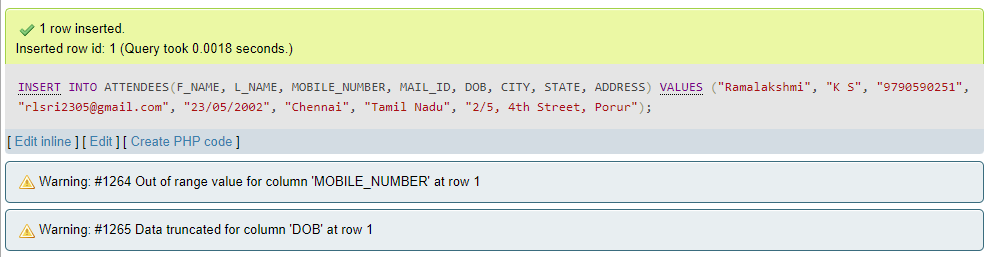
Database can be populated using 2 methods –

1. Implemented Streamlit GUI
2. Direct command line SQL Queries

**Implemented Streamlit GUI**

****

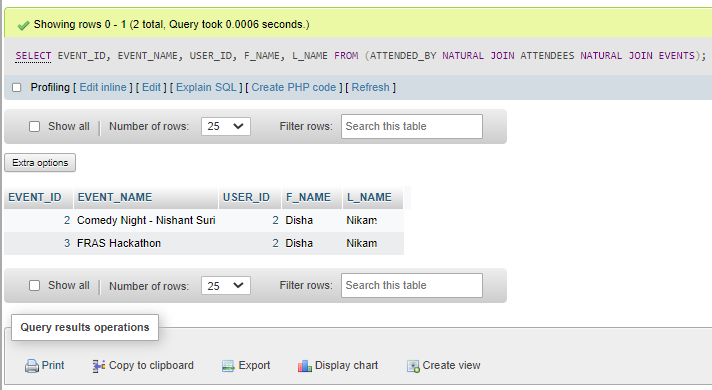
**Direct command line SQL Queries**



**Join Queries**

**JOIN QUERY 1** – Retrieve information about the attendees of a particular event

SELECT EVENT\_ID, EVENT\_NAME, USER\_ID, F\_NAME, L\_NAME FROM (ATTENDED\_BY NATURAL JOIN ATTENDEES NATURAL JOIN EVENTS)

****

**JOIN QUERY 2 –** Retrieve Event Names whose bills are pending

**Aggregate Functions**

Showcase at least 4 Aggregate function queries

Write the query in English Language, Show the equivalent SQL statement and also a screenshot of the query and the results

**Set Operations**

Showcase at least 4 Set Operations queries

Write the query in English Language, Show the equivalent SQL statement and also a screenshot of the query and the results

**Functions and Procedures**

Create a Function and Procedure. State the objective of the function / Procedure. Run and display the results.

**Triggers and Cursors**

Create a Trigger and a Cursor. State the objective. Run and display the results.

**Developing a Frontend**

The frontend should support

1. Addition, Modification and Deletion of records from any chosen table 2. There should be an window to accept and run any SQL statement and display the result