Royal Commission for Jubail and Yanbu

Jubail Industrial College

Computer Science & Information Technology Department

Database Systems Project

Semester: 441

Course Code	CS350	Course Title	Database Systems
Assigned	Week 6	Submission	Week 14 (December 4, 2022.)

Students are requested to comply with all JUC examination rules and regulations strictly.

	MaxMarks	1'stStudent Name: Raneem Alshehri ID: 401200244 Section#: 202	2'ndStudent Name: Remas Almutairi ID: 401200074 Section#: 202	3'rdStudent Name: Manar Alali ID: 421200173 Section#: 202
Requirement analysis	2			
Conceptual design	2			
Logical design	4			
DB Implementation	8			
Application	5			
Normalization	2			
Report and presentation	4			
SIS	total marks*16/27			

Table of Contents

For this project, we have designed and created an “Event Finder” database. Where, users, whether they may be tourists or citizens can login to the application and check the available list of events that are hosted in Saudi Arabia. After the user has picked an event they will register for it, thus obtaining a ticket. Therefore, the end product of our database (the interface) will be the relationship between the user and the ticket booked for the event.

Stage 1: Requirement analysis

We discuss the integral requirements that our database must maintain for the theme of our application:

- The database first has its central component: **EVENT**. Each event has an Event ID, the type of event being hosted, the starting and ending time of every event. Where the total duration can be calculated by the timings stated in the database. An event may have multiple locations to be attended by a user.
- Each event is organized and managed by a number of **ORGANIZERS**. Each organizer has an organizer ID, their names (first name, middle initial, and last name), and contact information.
  - Many organizers can work on different events, and the DB stores how many hours each organizer worked on an event.
- A **TICKET** is generated for each event. A ticket has a unique ticket ID, price of the ticket, the seat number allocated to the user, and when said ticket will expire (date to be last used).
  - Each ticket has a single event where it corresponds to.
- The database will store information about the **USERS**. Each user has a unique User ID, first name and last name, email, phone number, Date of Birth, and sex.
  - A user can choose to attend multiple events as long as their timings don’t overlap.
  - Booking date is also recorded in the DB, simply for added reference to the user.
- An event may have a number of **SPONSORS** to promote them.
  - The DB keeps track of the sponsors name, as well as the funding costs offered.
  - A sponsor must only promote one event
- Lastly, an organizer may employ a **PERMORMER**/s. Each performer has a unique performer ID, name, and the type of performance to be given at the event.
  - A performer is employed by one organizer only.

Stage 2: Conceptual design

The ER diagram is a key diagram, that helped envision the relations and the possible relationships that could be formed in the database. For this database 'Event Finder', we designed the following ER diagram with the constrains shown below:

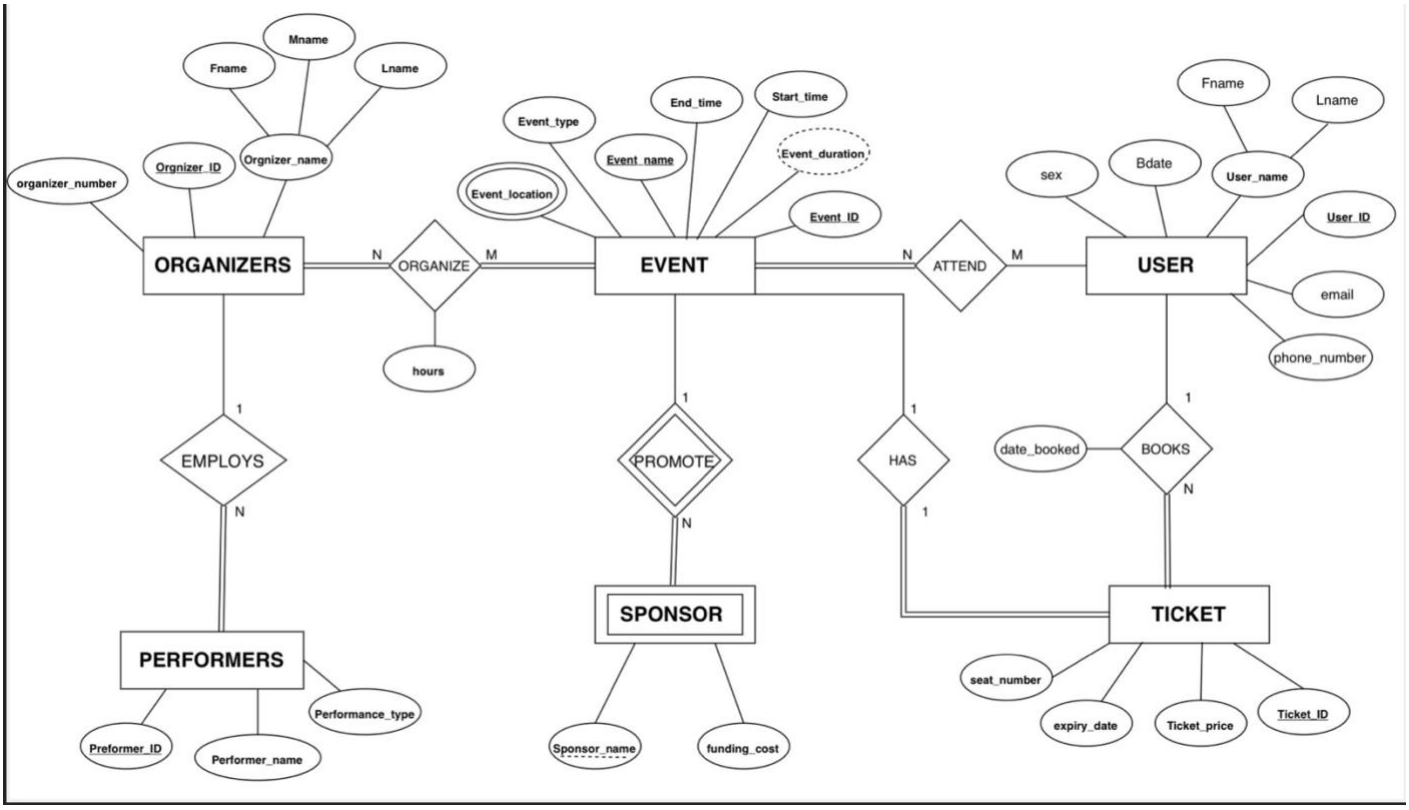

Figure 1: ER diagram

Stage 3: Data model mapping

The 7-rule mapping criteria was applied to create a relational database diagram. The relations and constraints are identified and clarified by the PK/FK relationships shown in the model.

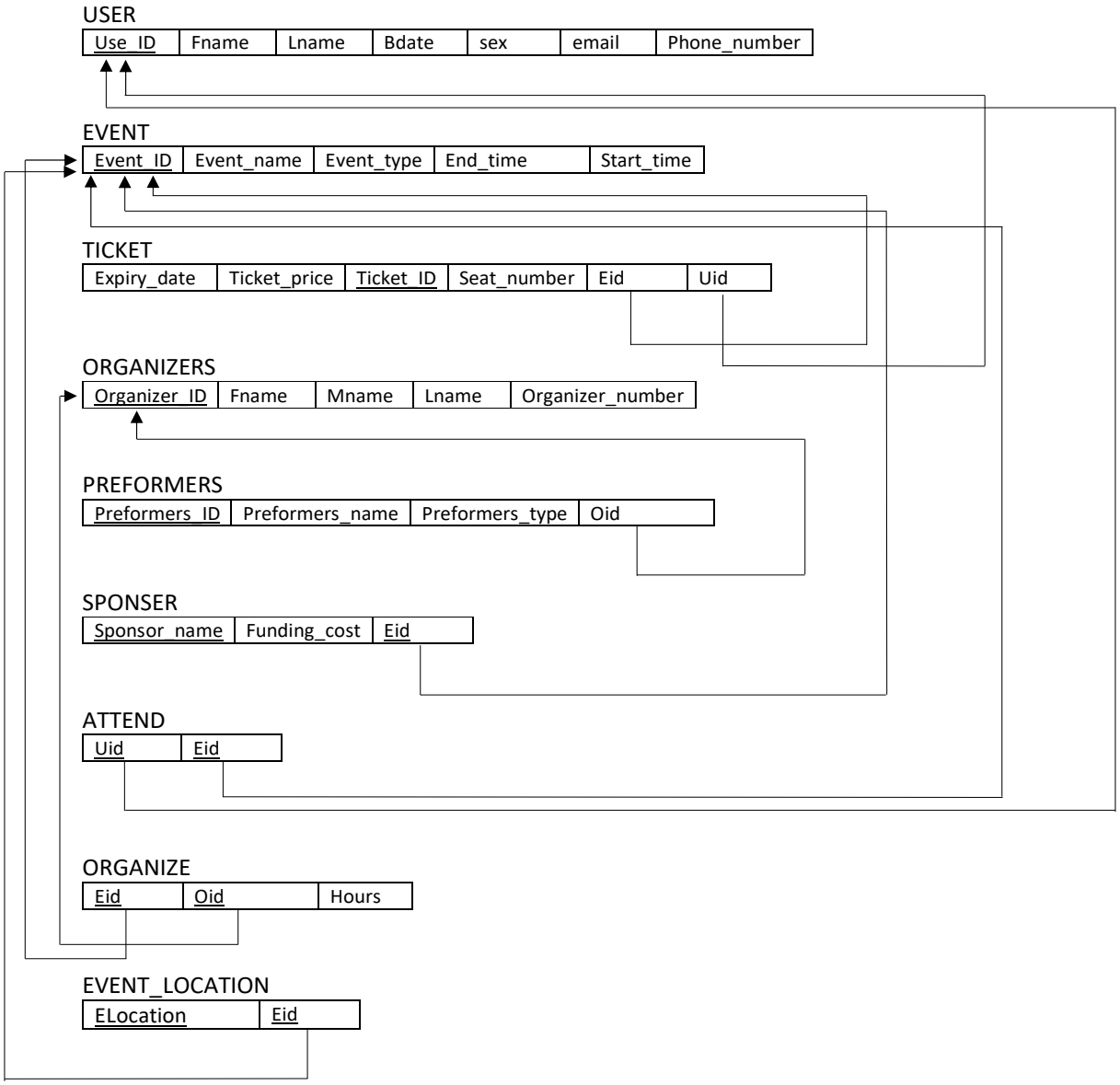

Figure 2: Relational Data Model

Stage 4: Implementation phase

For this phase, we used SQL commands to create the database that was designed in the relational data model. This includes specifying primary keys, maintaining PK/FK relationships, and checking for certain constraints. The SQL code and subsequently its populated database will be provided in the following images.

USER:

```sql
create table USER(
    User_ID char(8) PRIMARY KEY not null,
    Fname VARCHAR(10),
    Lname VARCHAR(10),
    Bdate DATE,
    sex CHAR(1),
    email varchar(50) unique not null,
    Phone_number char(10) unique not null,
    constraint chk_phone check (Phone_number not like '%[^0-9]%')
);
```

Figure 3: USER SQL

User_ID	Fname	Lname	Bdate	sex	email	Phone_number
20400000	John	Smith	1965-01-09	M	john_smith@email.com	0500000000
20400011	Abdullah	AlShahrani	1984-11-27	M	Adbulla_Alsh@email.com	0543401638
20400111	James	Borg	1977-03-12	M	james_borg@email.com	0501284729
20411111	Omar	AlQahatani	1994-07-03	M	omar_alqahtani@email.com	0558583520
20411112	Ahmed	Alessa	2001-08-21	M	ahmed_alessa@email.com	0501274628
20422222	Jill	Davis	2001-06-20	M	jill_davis@email.com	0539736112
20433333	Mohammed	AlNaser	2004-04-28	M	mohammed_alnaser@email.com	0540126715
20433344	Fahad	AlDossary	1998-05-22	M	fahad_aldossary@email.com	0556712986
20433444	Saad	AlShehri	1992-10-23	M	saad_alshehri@email.com	0538345280
20434444	Cole	Anderson	1995-11-11	M	cole_anderson@email.com	0507122930
20444445	Salman	AlGhamdi	1990-04-15	M	salman_alghamdi@email.com	0551238364
20400001	Fatima	Alali	1998-04-16	F	Fatima_ali@email.com	0547163881
20401111	Mariam	AlSalim	2000-02-15	F	mariam_alsalim@email.com	0502615373
20411122	Noora	AlOtaibi	2002-10-31	F	noora_alotaibi@email.com	0561274219
20411222	May	Walters	2004-12-14	F	mat_walters@email.com	0595123155
20412222	Emma	Brown	1997-03-07	F	emmam_brown@e3mail.com	0567234424
20412223	Jane	Smith	1980-09-10	F	jane_smith@email.com	0507282363
20412233	Shatha	AlKhaldi	1972-10-10	F	shatha_alkhaldi@email.com	0501222222
20412333	Hiba	AlHarbi	2002-02-24	F	hiba_alharbi@email.com	0558271632
20433334	Susan	Wilson	2003-12-01	F	susan_wilson@email.com	0533455633
20444444	Reem	AlShamrani	2000-07-26	F	reem_alshamrani@email.com	0567882332
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 4: USER DATA POPULATION

EVENT:

```sql
create table event (
  event_id char(8) NOT NULL,
  event_name varchar(150) ,
  event_type varchar(150) ,
  start_time datetime ,
  end_time datetime ,
  primary key(event_id),
  CONSTRAINT CHK_datetime CHECK (start_time between '2022-01-01 00:00:00' AND '2022-12-31 00:00:00' AND end_time between '2022-01-01 00:00:00' AND '2022-12-31 00:00:00' )
);
```

Figure 5: EVENT SQL

event_id	event_name	event_type	start_time	end_time
40100001	the saudi cup	sports	2022-01-23 18:00:00	2022-01-25 23:00:00
40100002	Explore Ithra	Culture & Visual Arts	2022-02-01 17:00:00	2022-02-28 22:00:00
40100003	Oasis	dining	2022-03-01 18:00:00	2022-03-28 00:30:00
40100004	Combat Field	sports	2022-01-01 15:00:00	2022-03-01 15:00:00
40100005	Winter Wonderland	Themed Attractions	2022-01-01 14:00:00	2022-03-31 22:00:00
40100006	Concert at Maraya	Show & Performance Art	2022-04-14 17:00:00	2022-04-30 23:00:00
40100007	AlUla Skies	Sightseeing	2022-02-27 13:00:00	2022-03-13 15:00:00
40100008	Riyadh Front	Shopping	2022-02-01 13:00:00	2022-02-28 20:00:00
40100009	diriyah season	sports	2022-10-20 15:00:00	2022-12-20 01:00:00
40100010	Janadriyah Festival	Culture & Visual Arts	2022-10-10 12:00:00	2022-10-20 20:00:00
40100011	Souq Okaz	Culture & Visual Arts	2022-10-01 12:00:00	2022-10-09 20:00:00
40100012	Riyadh Book Fair	Exhibitions	2022-07-14 17:00:00	2022-04-18 22:00:00
40100013	Biking Tour	soprts	2022-12-10 16:00:00	2022-12-10 23:00:00
40100014	Farmer and Cooking Souq	Dining	2022-11-05 12:00:00	2022-12-15 17:00:00
40100015	Winter at Tantora	Show & Performance Art	2022-01-01 16:00:00	2022-02-12 23:00:00
40100016	Boulevard Riyadh City	Themed Attractions	2022-01-01 18:00:00	2022-04-01 00:00:00
40100017	Symphony Under the Stars	Show & Performance Art	2022-03-11 16:00:00	2022-03-11 23:00:00
40100018	Saudi Coffee Exhibition	Exhibitions	2022-11-04 18:00:00	2022-11-08 20:00:00
40100019	International Cybersecurity Forum	IT & Technology	2022-06-05 18:00:00	2022-06-08 22:00:00
40100020	The Perfume Expo	Exhibition	2022-10-25 18:00:00	2022-12-29 23:00:00
NULL	NULL	NULL	NULL	NULL

Figure 6: EVENT DATA POPULATION

TICKET & ORGANIZERS:

```sql
create table TICKET(
    Expiry_date date,
    Ticket_price decimal(10,2),
    Ticket_ID char(8) primary key not null,
    Seat_number varchar(3),
    Eid char(8),
    Uid char(8),
    constraint chk_seat check(Seat_number>0 and Seat_number<1000),
    constraint chk_price check(Ticket_price>0 and Ticket_price<1000.00),
    foreign key(Eid) references event(event_id),
    foreign key(Uid) references USER(User_ID)
    );

CREATE TABLE ORGANIZERS(
        Organizer_ID char(8) not null,
        Fname VARCHAR(10),
        Mname VARCHAR(10),
        Lname VARCHAR(10),
        Organizer_number char(10) not null,
        PRIMARY KEY(Organizer_ID)
        );
```

Figure 7: TICKET AND ORGANIZERS SQL

Expiry_date	Ticket_pri...	Ticket_ID	Seat_number	Eid	Uid
2022-06-08	255.00	30300001	57	40100019	20400001
2022-06-08	255.00	30300002	12	40100019	20400011
2022-12-29	488.25	30300003	105	40100020	20401111
2022-12-29	488.25	30300004	17	40100020	20400001
2022-12-29	488.25	30300005	33	40100020	20411122
2022-11-08	70.00	30300006	10	40100018	20433333
2022-11-08	70.00	30300007	57	40100018	20400000
2022-11-08	70.00	30300008	04	40100018	20434444
2022-11-08	70.00	30300009	90	40100018	20400001
2022-11-08	70.00	30300010	45	40100018	20444444
2022-03-11	330.75	30300011	28	40100017	20411222
2022-03-11	330.75	30300012	44	40100017	20433333
2022-03-11	330.75	30300013	18	40100017	20412233
2022-01-25	630.00	30300014	06	40100001	20412223
2022-02-28	552.25	30300015	05	40100002	20422222
2022-03-28	200.00	30300016	17	40100003	20412222
2022-03-01	128.00	30300017	39	40100004	20411222
2022-03-31	762.25	30300018	24	40100005	20411112
2022-04-30	95.25	30300019	26	40100006	20411111
2022-03-13	266.75	30300020	21	40100007	20400001
2022-02-28	66.00	30300021	89	40100008	20422222
2022-12-20	198.20	30300022	104	40100009	20433333
2022-10-20	777.90	30300023	290	40100010	20400001
2022-10-09	550.00	30300024	13	40100011	20444445
2022-04-18	330.20	30300025	08	40100012	20433344
2022-12-10	168.00	30300026	12	40100013	20412333
2022-12-15	202.78	30300027	356	40100014	20411111
2022-02-12	630.20	30300028	235	40100015	20411122
2022-04-01	60.20	30300029	52	40100016	20412222

Figure 8: TICKET DATA POPULATION

Organizer_ID	Fname	Mname	Lname	Organizer_numb...
39909986	Rama	Tariq	Almaliki	0586603007
39911939	Abdullah	Husain	alMansour	0565332956
39914474	Basher	Ali	Ghazi	0561114889
39915655	Laila	Zaher	AlShahrani	0509815177
39921761	Hiba	Amen	ALDawood	0568059240
39929346	Aman	Othman	AlAsiri	0558322779
39932066	Jack	Eric	Elliott	0564411340
39935693	Fahad	Tariq	AlQahtani	0524668121
39936596	Akram	Mohammad	Alrashed	0501726511
39941020	Arnold	Robert	Gilbert	0560030205
39946256	Iffah	Malik	Alkhaldi	0550067409
39957968	Saad	Zaher	Salama	0565067211
39959962	Basel	Sharif	Alharbi	0559255991
39961336	Emily	Joey	Walters	0547356008
39961880	Carmen	Tyler	Oliver	0538903330
39963231	Razan	Yazid	AlQahtani	0555899844
39976778	Lara	Peter	Brown	0537120736
39980856	Soha	Khalaf	Mohammadi	0528326027
39982261	Amir	Naseem	AlSalim	0560139473
39985512	Zaniah	Rashed	AlShehri	0556747830

Figure 9: ORGANIZERS DATA POPULATION

PERFORMERS & SPONSORS & ATTEND:

```sql
create table PERFORMERS(
Performers_ID char(8) primary key not null,
Performers_name varchar(30) not null,
Performers_type varchar(150),
Oid char(8),
foreign key(Oid) references ORGANIZERS(Organizer_ID)
);

create table sponsor (
eid char(8) not null,
sponsor_name varchar(150)  ,
funding_cost double ,
FOREIGN KEY (eid) REFERENCES event(event_id));

create table ATTEND(
uid char(8),
eid char(8),
foreign key(uid) references USER(User_ID),
foreign key(eid) references event(event_id)
);
```

Figure 10: PERFORMERS & SPONSOR & ATTEND SQL

Performers_ID	Performers_name	Performers_type	Oid
31101405	Rowel Guevarra	executive chef for edafat+	39961880
31122943	Mery Acevedo	professional ice skating trainer	39929346
31135415	Saad AlDossary	Cultural Seminar	39914474
31148610	Basher Abdullah	tourist camping Guide	39963231
31150174	Abdul Samad Al-Qurashi	seminars about producer of Arabian Perfumes	39959962
31151021	BLACK HAT	cybersecurity professionals group trainer	39932066
31155532	Future	singer	39959962
31161882	Nawal El-Kuwaitia	singer	39936596
31194422	Muhammad Al-Ghazi	poetry evening	39985512
NULL	NULL	NULL	NULL

Figure 11: PERFORMERS DATA POPULATION

eid	sponsor_name	funding_cost
40100001	RIYAD BANK	250000
40100002	OCCASION	135000
40100002	Aramco	180000
40100004	LIKECARD	100000
40100005	hungerstation	160000
40100005	ARAMEX	90000
40100006	SELA	70000
40100006	Saudia Airlines	120000
40100008	STC PAY	275000
40100009	Ministry of Tourism	140000
40100013	PARKYY	90000
40100016	The Chefz	100000
40100018	COFE	880000
40100019	NETWORK INTERNATIONAL ARABIA	200000
40100020	Abdulsamad Al Qurashi	160000

Figure 12: SPONSOR DATA POPULATION

uid	eid
20411222	40100002
20444444	40100001
20412222	40100016
20411122	40100015
20411111	40100014
20412333	40100013
20433344	40100012
20444445	40100011
20400001	40100010
20433333	40100009
20422222	40100008
20412233	40100017
20433333	40100017
20411222	40100017
20400001	40100020
20401111	40100020
20400001	40100019
20400000	40100019
20412223	40100001
20422222	40100002

Figure 13: ATTEND DATA POPULATION

ORGANIZE & EVENT_LOCATION:

```sql
create table Organize
(
Eid char(8),
Oid char(8),
Hours DECIMAL(3,1),
foreign key(Eid) references event(event_id),
foreign key(Oid) references ORGANIZERS(Organizer_ID)
);

create table event_location(
 eid char(8) not null,
 elocation varchar(150) ,
 FOREIGN KEY (eid) REFERENCES event(event_id));
```

Figure 14: ORGANIZE & EVENT_LOCATION SQL

Eid	Oid	Hours
40100002	39909986	30.5
40100019	39915655	10.6
40100008	39963231	20.0
40100012	39929346	5.0
40100001	39982261	25.5
40100001	39976778	4.0
40100020	39976778	12.0
40100010	39936596	35.0

Figure 16: ORGANIZE DATA POPULATION

eid	elocation
40100001	Riyadh
40100002	Dhahran
40100003	Riyadh
40100004	Riyadh
40100005	Riyadh
40100006	AlUla
40100007	AlUla
40100008	Riyadh
40100009	Riyadh
40100010	Riyadh
40100011	Taif
40100012	Riyadh
40100013	Jeddah
40100014	Dhahran
40100015	AlUla
40100016	Riyadh
40100017	AlUla
40100018	Riyadh
40100019	Dhahran
40100020	Riyadh

Figure 15: EVENT_LOCATION DATA POPULATION

Stage 5: Application development phase:

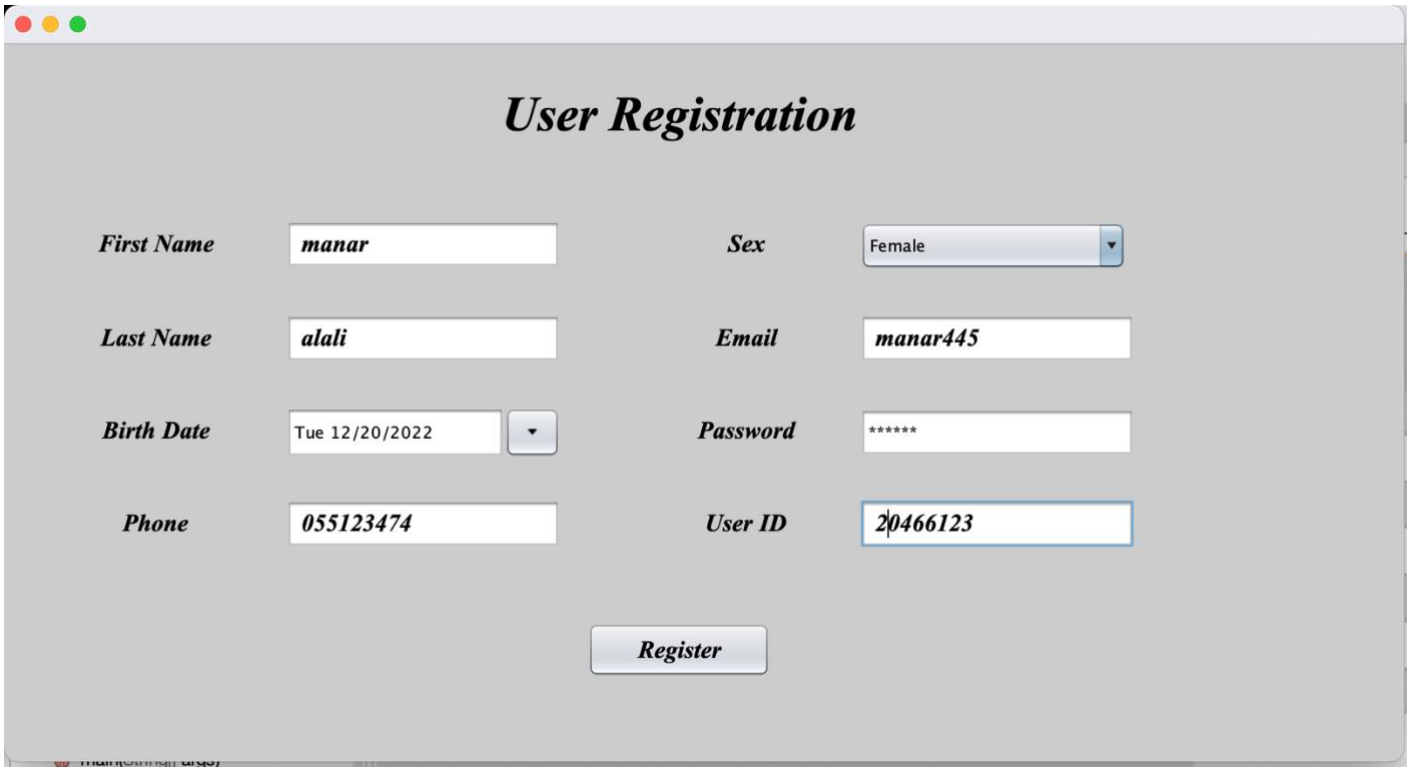

Figure 17: Users will register their information in this interface, where all relevant data concerning said user will be stored in the database

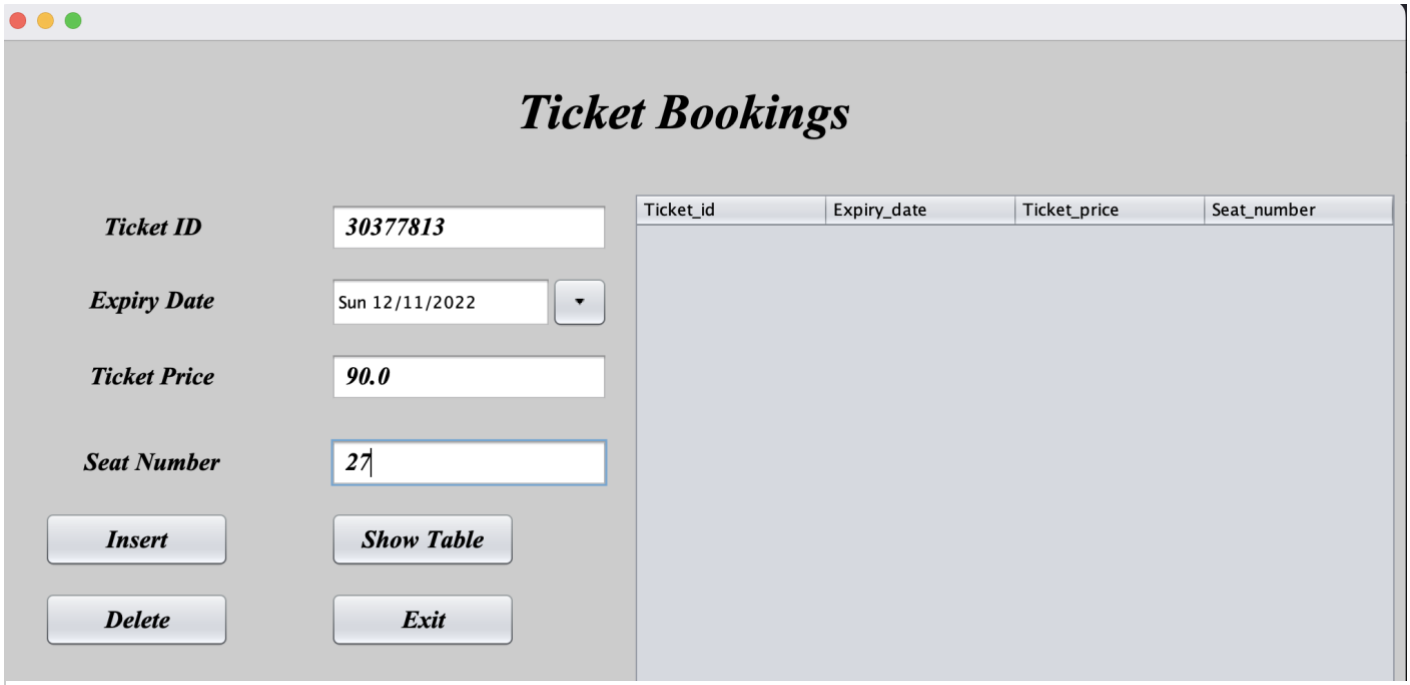

Figure 18: Users will be guided to book tickets through this interface. Where, users can insert, delete, or see any available tickets they have for certain events.

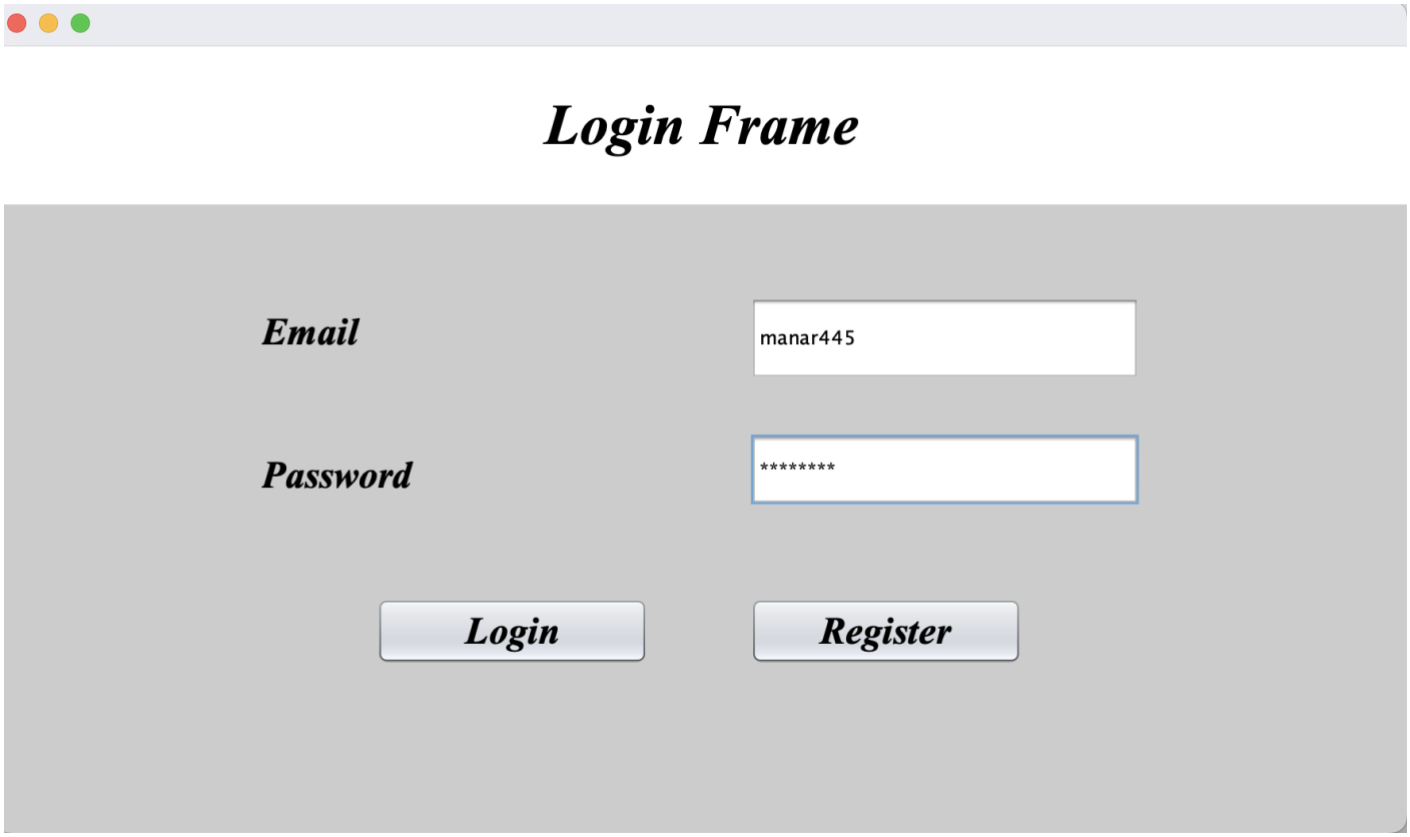

Figure 18: Users will be guided to a Login page, where they enter their email and password. This is done for already existing and registered users.

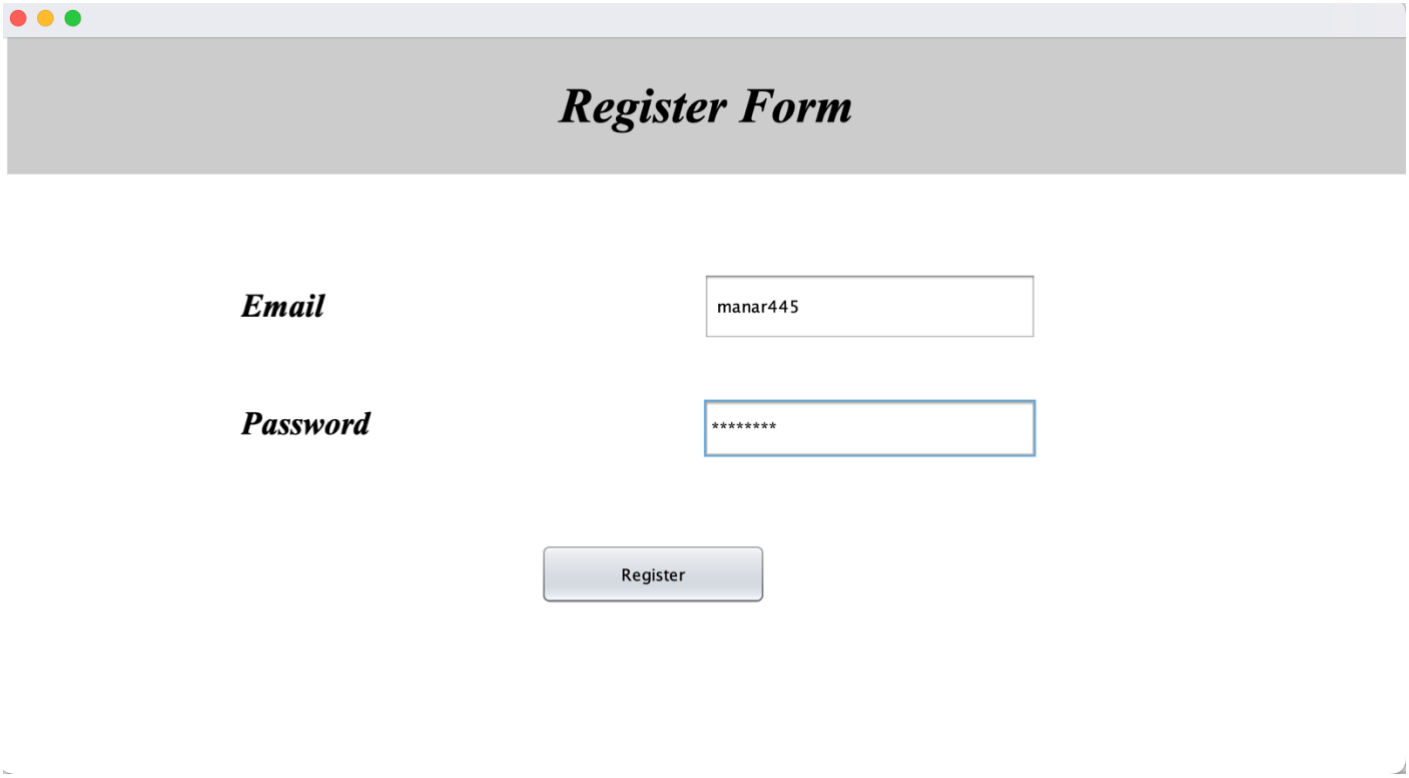

Figure 20: If data doesn't exist in the previous interface. Then users will have to register an email and password to login to their accounts

Stage 6: Refine your database:

This stage entails the normalization of each relation to the first, second, and third normal form;

'USER' Normalization:

First Normal Form "1NF"

USER_ID → Fname,Lname,Bdate,sex,email,Phone_number

Second Normal Form "2NF"

USER_ID → Fname,Lname,Bdate,sex,email,Phone_number

Third Normal Form "3NF"

USER_ID → Bdate,sex,email,Phone_number

Phone_number → Fname,Lname

'EVENT' Normalization:

First Normal Form "1NF"

Event_ID → Event_name,Event_type,End_time,Start_time

Second Normal Form "2NF"

Event_ID → Event_name,Event_type,End_time,Start_time

Third Normal Form "3NF"

Event_ID → Event_name

Event_name → Event_type,End_time,Start_time

'TICKET' Normalization:

First Normal Form "1NF"

Ticket_ID → Expiry_date,Ticket_price,Seat_number,Eid,Uid

Second Normal Form "2NF"

Ticket_ID → Expiry_date,Ticket_price,Seat_number,Eid,Uid

Third Normal Form "3NF"

Ticket_ID → Expiry_date,Ticket_price,Seat_number,Eid,Uid

‘ORGANIZERS’ Normalization:

First Normal Form “1NF”

Organizer_ID → Fname,Mname,Lname,Organizer_number

Second Normal Form “2NF”

Organizer_ID → Fname,Mname,Lname,Organizer_number

Third Normal Form “3NF”

Organizer_number → Fname,Mname,Lname

‘PERFORMERS’ Normalization:

First Normal Form “1NF”

Performers_ID → Performers_name,Performers_type,Oid

Second Normal Form “2NF”

Performers_ID → Performers_name,Performers_type,Oid

Third Normal Form “3NF”

Oid → Performers_name,Performers_type

‘SPONSOR’ Normalization:

First Normal Form “1NF”

Sponsor_name,EID → Funding_cost

Second Normal Form “2NF”

Sponsor_name,EID → Funding_cost

‘ORGANIZE’ Normalization:

First Normal Form “1NF”

Eid,Oid → Hours

Second Normal Form “2NF”

Eid,Oid → Hours

Appendix:

In this section, Screenshots of the Java application code will be provided. Note: only code pertaining to the CRUD operations i.e., insert, retrieve, update, and delete on the 'TICKETS' will be shown. The rest of the code is available on the source code attached.

Setting up a connection

```java
public class DBconnection {
    public static  Connection getConnection(){
        Connection con=null;
         try{
             Class.forName( className:"com.mysql.jdbc.Driver");
             con =DriverManager.getConnection( url:"jdbc:mysql://localhost:3306/EventFinder",  user:"root",  password:"111111");

             System.out.println( x:"Connected Successfully");
        }
        catch(Exception e){
            System.out.println("Error : " + e.getMessage());
            e.printStackTrace();
        }
         return con;

    }
}
```

Figure 21: Shows how to connect this java class with the mySQL schema

Insert statement

```java
private void insertbuttonActionPerformed(java.awt.event.ActionEvent evt) {
    DateFormat da    =    new SimpleDateFormat( pattern:"yyy-MM-dd");
    String expire = da.format( date:expirydate.getDate());
    String tkt_id = ticketid.getText();
    String price = ticketprice.getText();
    String seat = seatnbr.getText();


    try {
        Connection con;
        con = DBconnection.getConnection();
        PreparedStatement pst = con.prepareStatement( sql:"insert into ticket_booking(Ticket_id, Expiry_date, Ticket_price, Seat_number) VALUES(?, ?, ?, ?)");

        pst.setString( parameterIndex:1,  x:tkt_id);
        pst.setString( parameterIndex:2,  x:expire);
        pst.setString( parameterIndex:3,  x:price);
        pst.setString( parameterIndex:4,  x:seat);
        pst.executeUpdate();
    } catch (Exception e) {
    }
}
```

Figure 22: Shows an Insert command. For entering new tickets that users booked

Delete Statement

```
private void deletebuttonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
     DefaultTableModel Model = (DefaultTableModel)tickettable.getModel();
    int row = tickettable.getSelectedRow();

    String cell = tickettable.getModel().getValueAt( rowIndex:row, columnIndex:0).toString();
    String sql = "DELETE FROM ticket_booking where id = "+cell;
    try {
      PreparedStatement  pst = con.prepareStatement(sql);
      pst.execute();
      Model.removeRow(row);
      JOptionPane.showMessageDialog( parentComponent:this,  message:"Deleted Succesfully");

    } catch (Exception e) {
        JOptionPane.showMessageDialog( parentComponent:this,  message:e);

    }
```

Figure 23: delete command. Used when a user wants to delete a ticket as to not go to an event

Retrieve Statement

```
private void showtableActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
     try {
        PreparedStatement preparedStatement = con.prepareStatement( sql:"Select Ticket_id, Expiry_date, Ticket_price, Seat_number from ticket_booking");
        ResultSet resultSet = preparedStatement.executeQuery();
         ResultSetMetaData rsmd = resultSet.getMetaData();
         DefaultTableModel Model = (DefaultTableModel)tickettable.getModel();
         int cols = rsmd.getColumnCount();
         String[] colname = new String[cols];
         for(int i=0;i<cols;i++){
         colname[i] = rsmd.getColumnName(i+1);
         Model.setColumnIdentifiers( newIdentifiers:colname);
         while(resultSet.next()){
              String id = resultSet.getString( columnIndex:1);
              String date = resultSet.getString( columnIndex:2);
              String tktprice = resultSet.getString( columnIndex:3);
              String seatnumber = resultSet.getString( columnIndex:4);
             String[] row ={id,date,tktprice,seatnumber};
             Model.addRow( rowData:row);
             System.out.println( x:row);
         }
         }


        } catch (Exception e) {
    }
}
```

Figure 24: retrieves and displays all data inserted into the table

Update Statement

```
private void updatebuttonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    DefaultTableModel Model = (DefaultTableModel)tickettable.getModel();
    String tbl_id = Model.getValueAt( row:tickettable.getSelectedRow(),  column:0).toString();
    String exp_date  = Model.getValueAt( row:tickettable.getSelectedRow(),  column:0).toString();
    String price  = Model.getValueAt( row:tickettable.getSelectedRow(),  column:0).toString();
    String seat  = Model.getValueAt( row:tickettable.getSelectedRow(),  column:0).toString();

    int row = tickettable.getSelectedRow();

    String cell = tickettable.getModel().getValueAt( rowIndex:row, columnIndex:0).toString();
    try {

        PreparedStatement pst = con.prepareStatement("update ticket_booking set Ticket_id = ?, Expiry_date = ?, Ticket_price = ?, Seat_number = ? where id ="+cell);

        pst.setString( parameterIndex:1,  x:tbl_id);
        pst.setString( parameterIndex:2,  x:exp_date);
        pst.setString( parameterIndex:3,  x:price);
        pst.setString( parameterIndex:4,  x:seat);
        pst.executeUpdate();
    } catch (Exception e) {
    }
}
```

Figure 25: if any data was erroneous or the user wants to change certain elements like the location/time of an event. Then update command is used