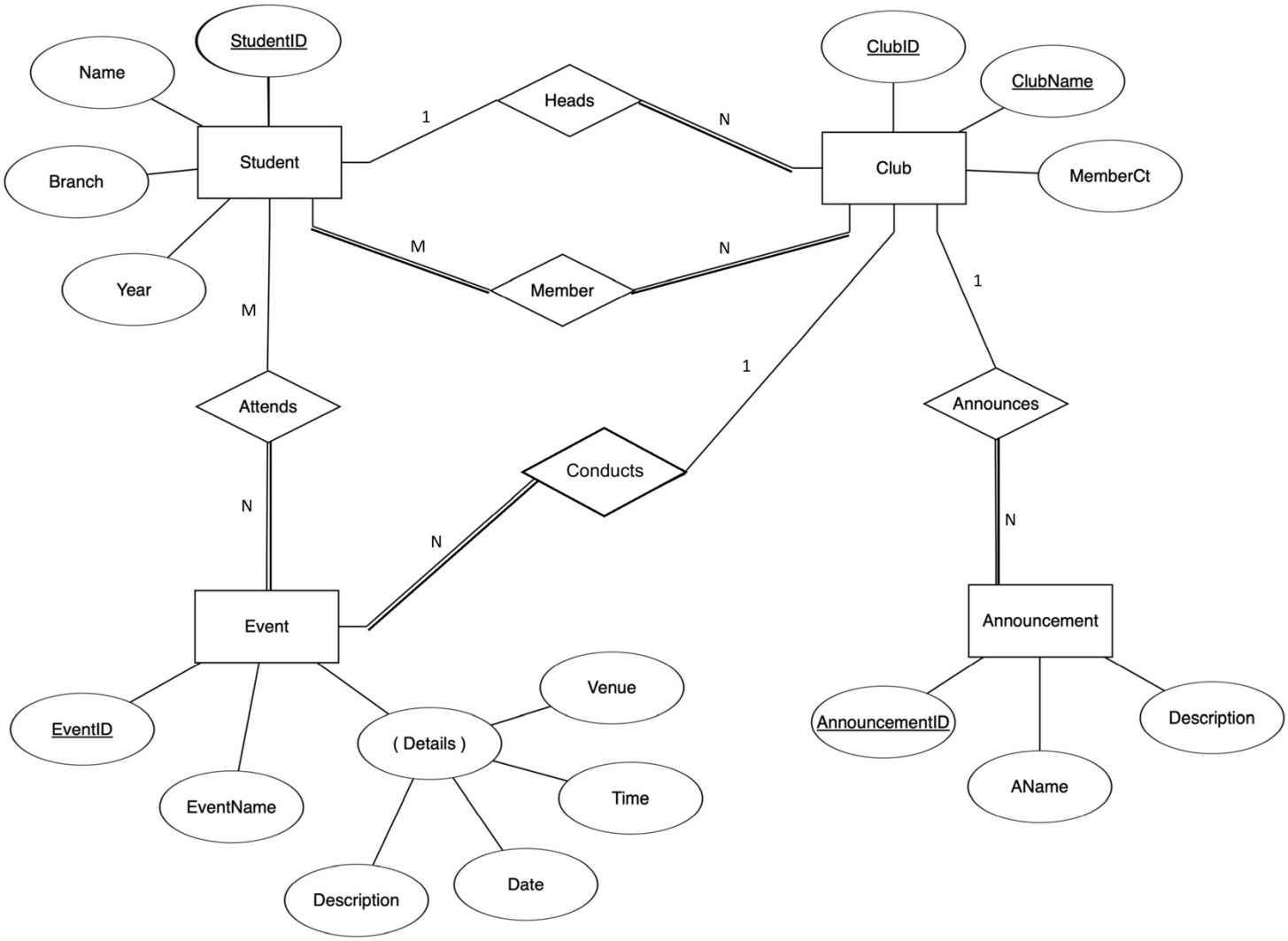
Abstract:

Most colleges have a variety of clubs and extra-curricular activities offered to the students. The primary difficulty faced by students is the lack of awareness about these events and activities. This database management system aims to act as an efficient location and portal for all the information, announcements and events of a college club. Club heads can update the events and announcements through this system, and students can also access these details easily.

Functions:

- Students can log into their accounts using a username and password
- They can
    - Join clubs
    - View announcements of their clubs
    - View and join/quit an event
- Club heads can
    - Add, delete and modify events
    - Add, delete and modify announcements of their respective clubs
- The number of members and attendees of each event are recorded and updated automatically

ER Diagram:



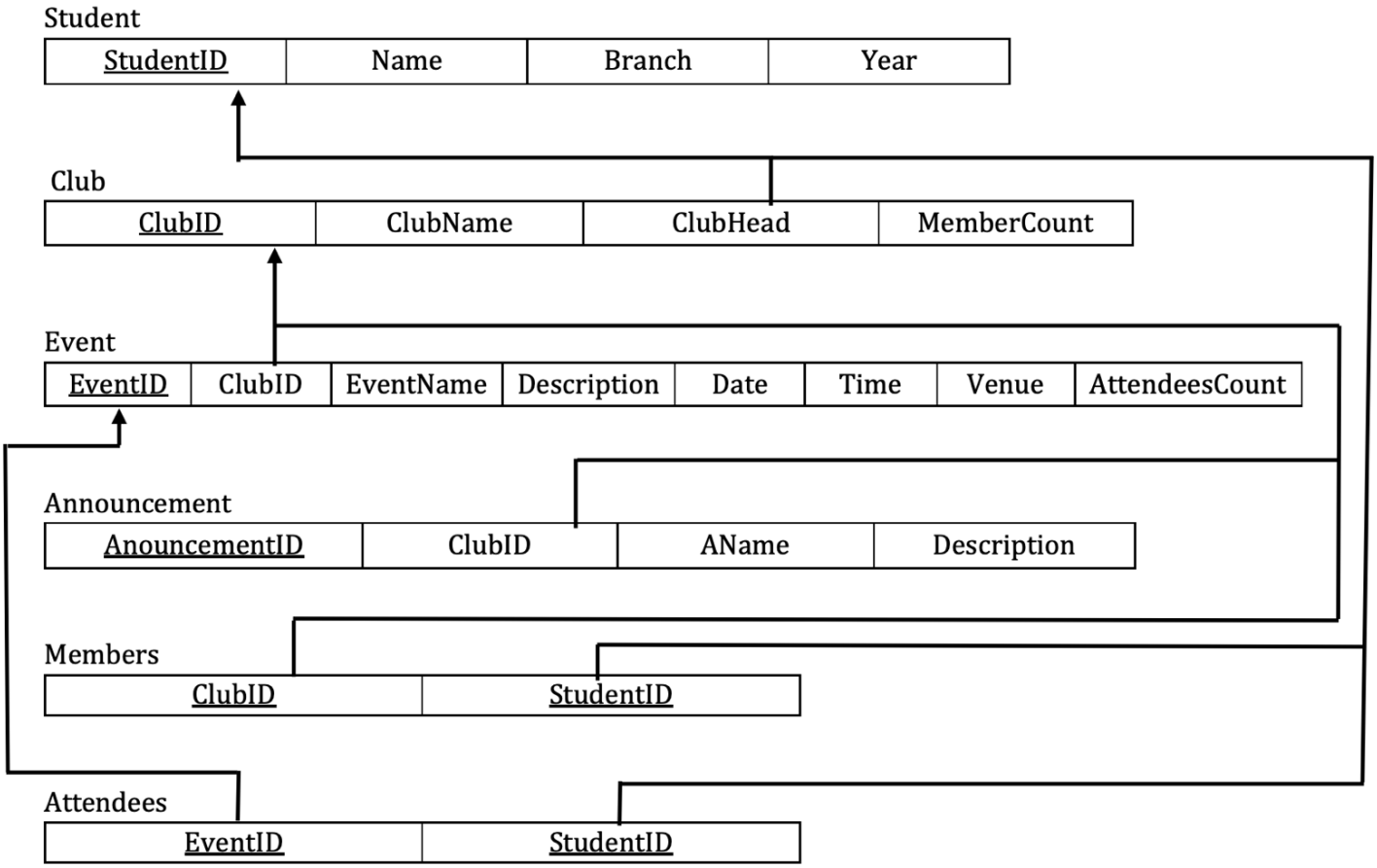
ER to Relational Mapping:

- The four regular entities- *Student*, *Club*, *Event* and *Announcement* are mapped to four relations consisting of their simple attributes
    - The composite attribute *Details* in *Event* is mapped using the simple attributes it consists of
- The keys are:

Relation	Primary Key	Candidate Key
Student	StudentID	
Club	ClubID	ClubName
Event	EventID	
Announcement	AnnouncementID	

- For the following 1:N relationship types:
    - Heads*: ClubHead* is included in *Club* relation as foreign key mapping to *StudentID* in *Student*
    - Conducts: *ClubID* is included in *Event* relation as foreign key mapping to *ClubId* in *Club*
    - Announces: *ClubID* is included in *Announcement* relation as foreign key mapping to *ClubId* in *Club*
- For the following M:N relationship types:
    - Member: A new relation *Members* was created with attributes *ClubID* and *StudentID* as foreign keys mapping to *Club* and *Student* relations respectively. The composite primary key is {*ClubID*, *StudentID*}
    - Attendees: A new relation *Attends* was created with attributes *EventID* and *StudentID* as foreign keys mapping to *Event* and *Student* relations respectively. The composite primary key is {*EventID*, *StudentID*}

The final schema diagram is:



Functional Dependencies and Normalisation:

A. *Student* Relation

Functional Dependencies:

1. StudentID → Name
2. StudentID → Branch
3. StudentID →Year

Student

StudentID	Name	Branch	Year

1st Normal Form

As the relation has no composite attributes, multivalued attributes, or nested relations, the relation ***Student* is in 1NF**.

2nd Normal Form

As every non-prime attribute is fully functionally dependent on the key *StudentID*, the relation ***Student* is in 2NF**.

3rd Normal Form

As the relation has no transitive dependencies, the relation ***Student* is in 3NF**.

Boyce-Codd Normal Form

As for every FD X → A, X is a superkey, the relation ***Student* is in BCNF**.

B. *Club* Relation

Functional Dependencies:

1. ClubID → ClubName
2. ClubName → ClubHead
3. ClubName → ClubID
4. ClubName → MemberCount

Club

ClubID	ClubName	ClubHead	MemberCount

1st Normal Form

As the relation has no composite attributes, multivalued attributes, or nested relations, the relation ***Club* is in 1NF**.

2nd Normal Form

As every non-prime attribute is fully functionally dependent on the primary key *ClubID* or candidate key *ClubName*, the relation ***Club* is in 2NF**.

3rd Normal Form

As the relation only has transitive dependencies X → Y and Y → Z where Y (*ClubName)* is a candidate key, the relation ***Club* is in 3NF**.

Boyce-Codd Normal Form

As for every FD X → A, X is a superkey, the relation ***Club* is in BCNF**.

C. *Event* Relation

Functional Dependencies:

1. EventID → EventName
2. EventID → ClubID
3. EventID → Description
4. EventID → Date
5. EventID → Time
6. EventID → Venue
7. EventID → AttendeesCount

Event

EventID	ClubID	EventName	Description	Date	Time	Venue	AttendeesCount

1st Normal Form

As the relation has no composite attributes, multivalued attributes, or nested relations, the relation ***Event* is in 1NF**.

2nd Normal Form

As every non-prime attribute is fully functionally dependent on the primary key *EventID*, the relation ***Event* is in 2NF**.

3rd Normal Form

As the relation only has no transitive dependencies, the relation ***Event* is in 3NF**.

Boyce-Codd Normal Form

As for every FD X → A, X is a superkey, the relation ***Event* is in BCNF**.

D. *Announcement* Relation

Functional Dependencies:

1. AnnouncementID → ClubID
2. AnnouncementID → AName
3. AnnouncementID →Description

Announcement

AnouncementID	ClubID	AName	Description

1st Normal Form

As the relation has no composite attributes, multivalued attributes, or nested relations, the relation ***Announcement* is in 1NF**.

2nd Normal Form

As every non-prime attribute is fully functionally dependent on the primary key *AnnouncementID*, the relation ***Announcement* is in 2NF**.

3rd Normal Form

As the relation only has no transitive dependencies, the relation ***Announcement* is in 3NF**.

Boyce-Codd Normal Form

As for every FD X → A, X is a superkey, the relation ***Announcement* is in BCNF**.

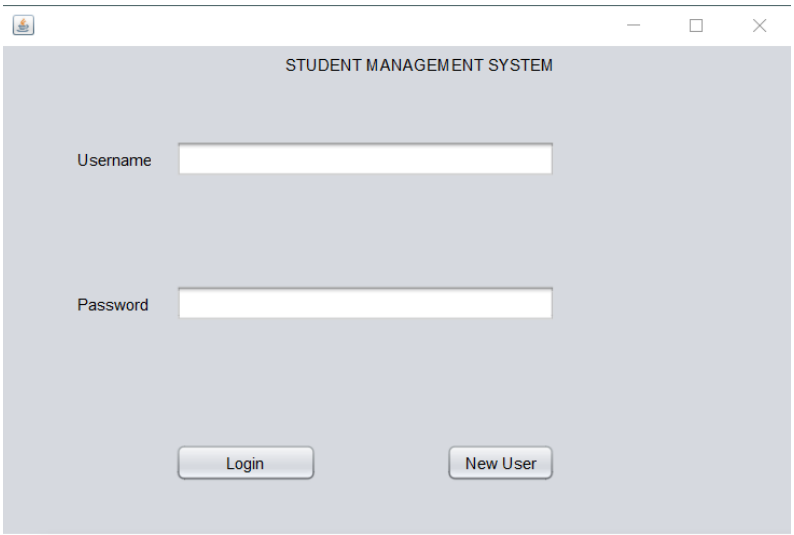
E. *Members* and *Attendees* Relations

As all the attributes of these relations are part of their respective primary keys, there are no FDs and hence there is no normalisation.

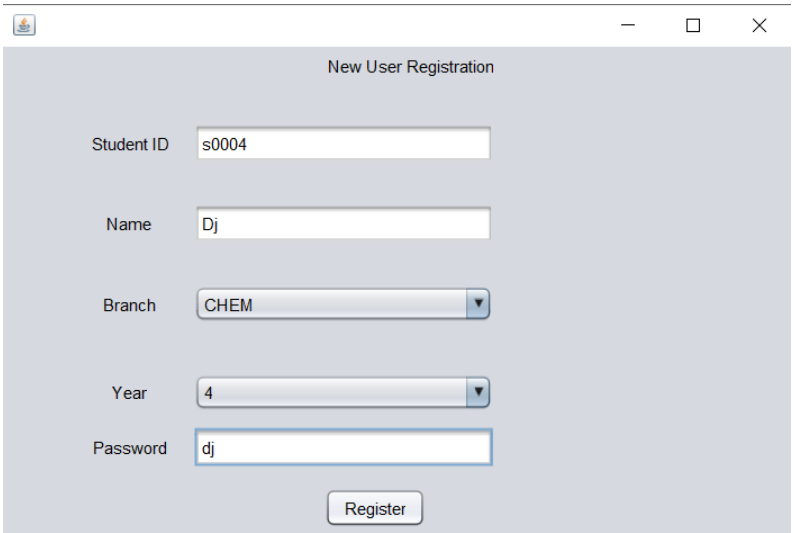
Module Screenshots:

Student view

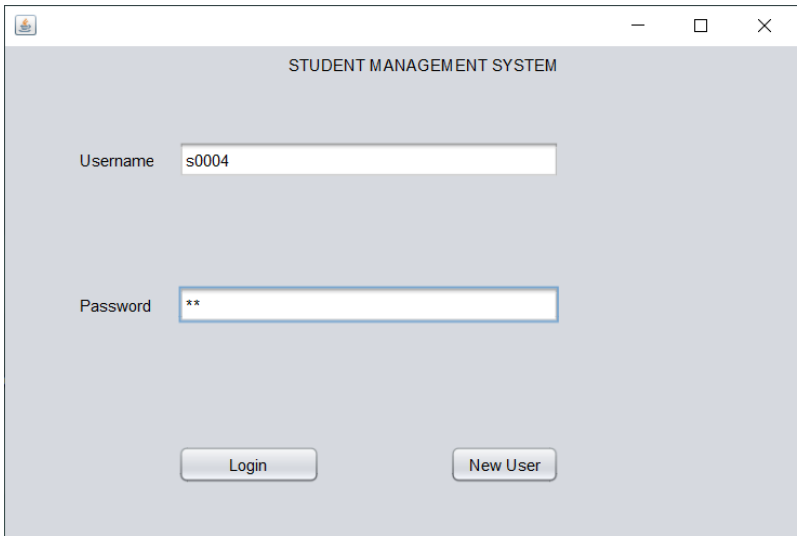
- Login/Register New User



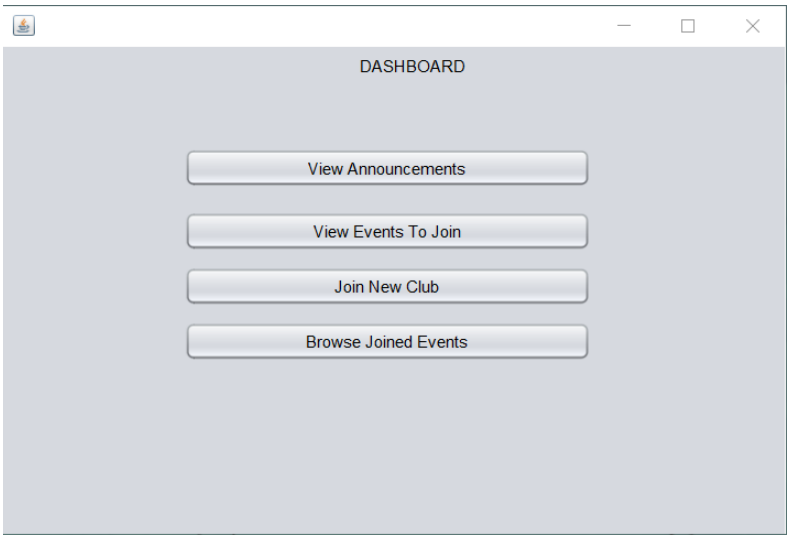
- Registering new user



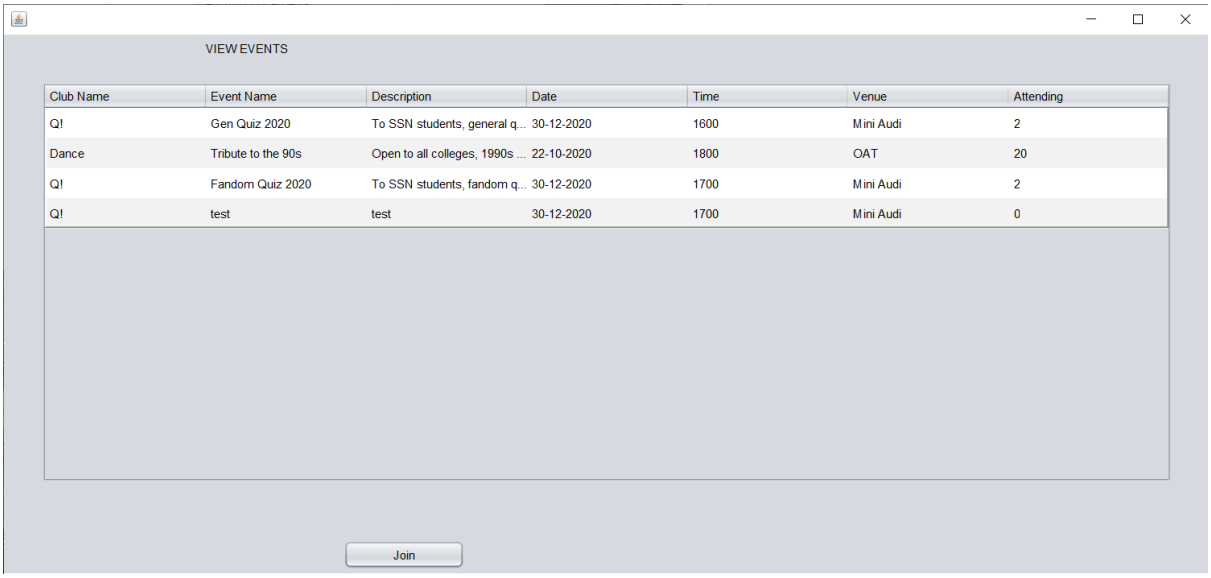
- Logging in

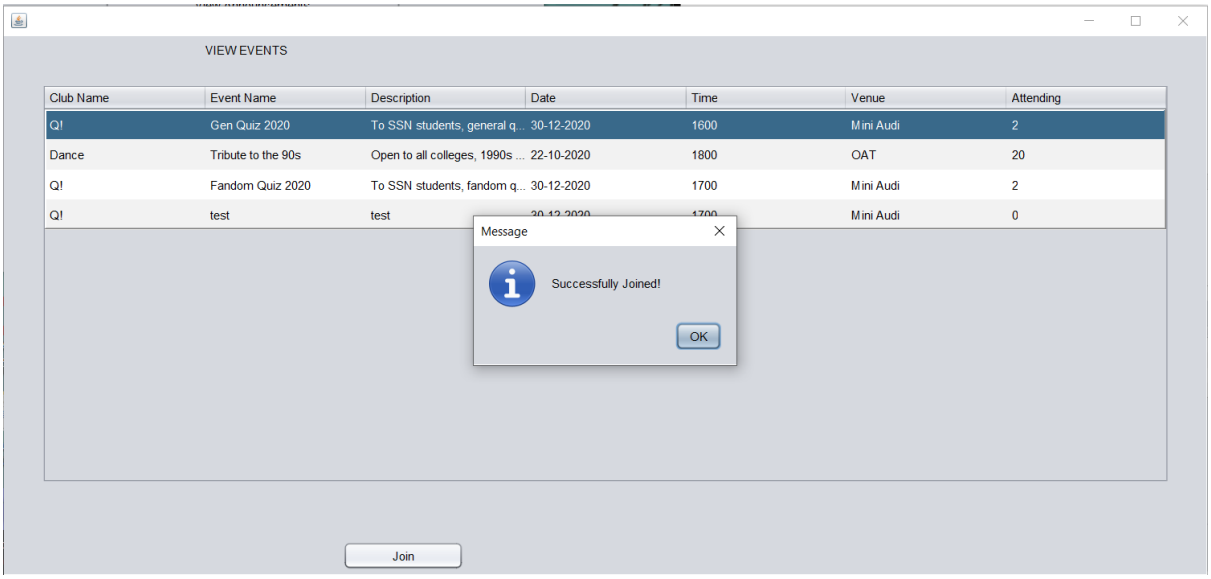


- Dashboard

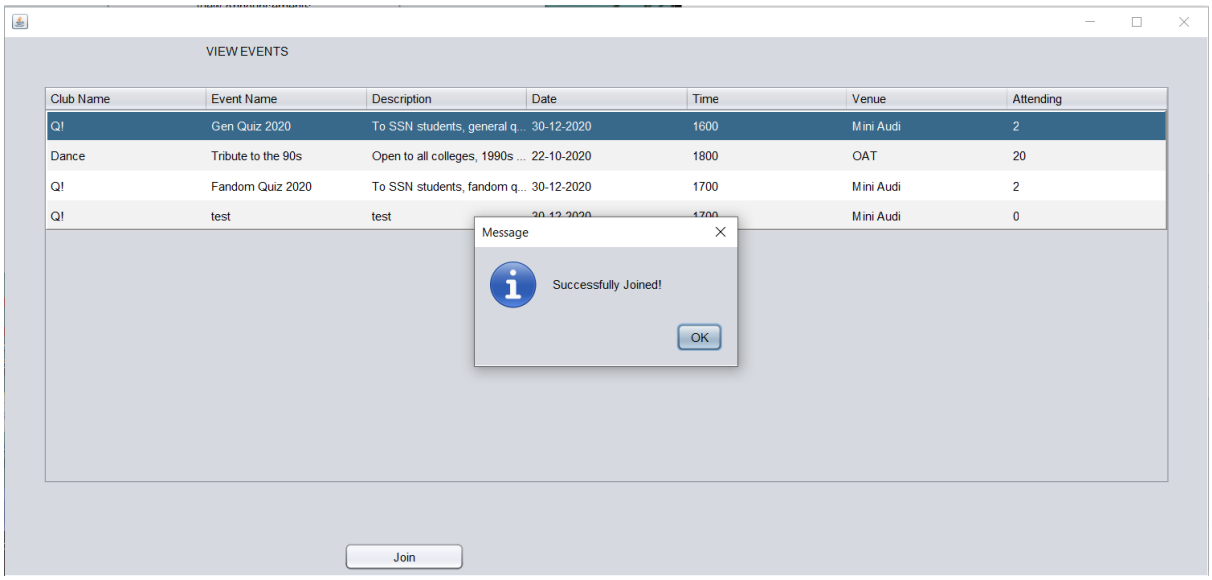


- **View Events to Join**
  - Students can view events and competitions conducted by all clubs and choose to join them

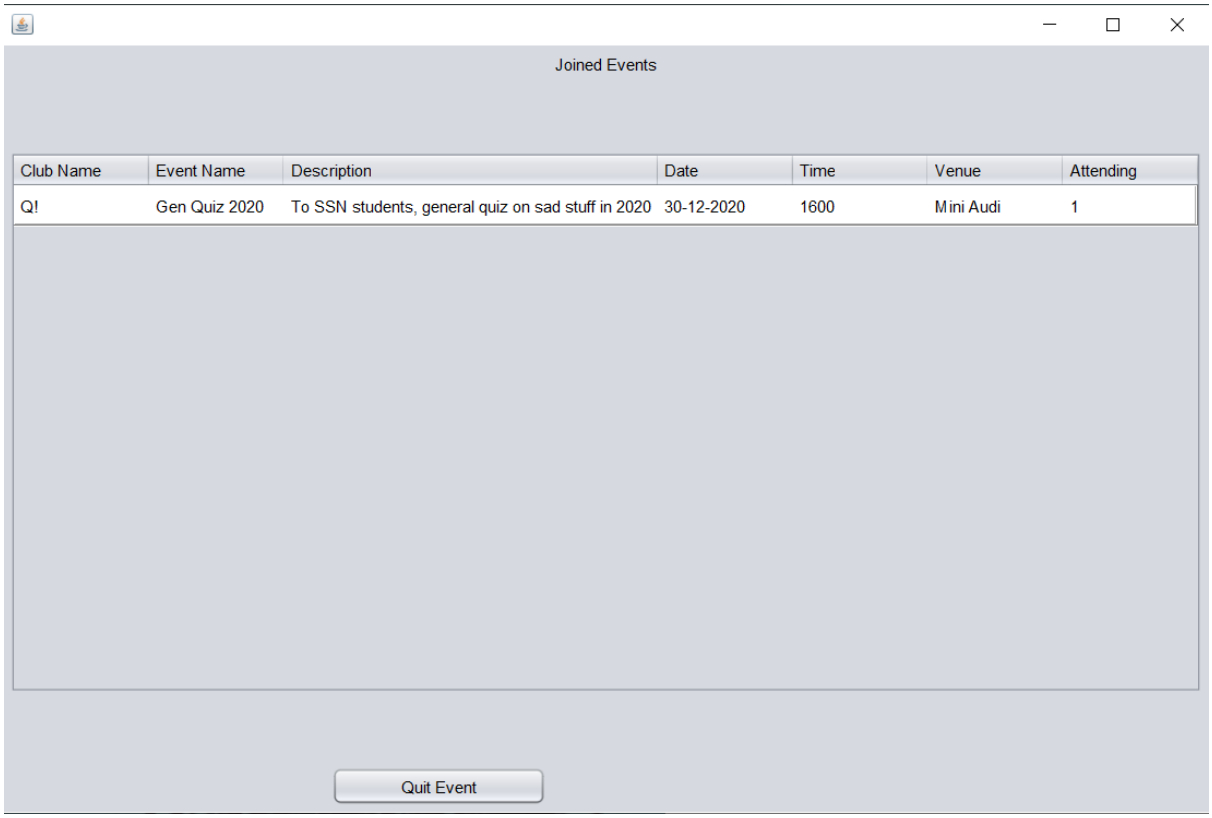




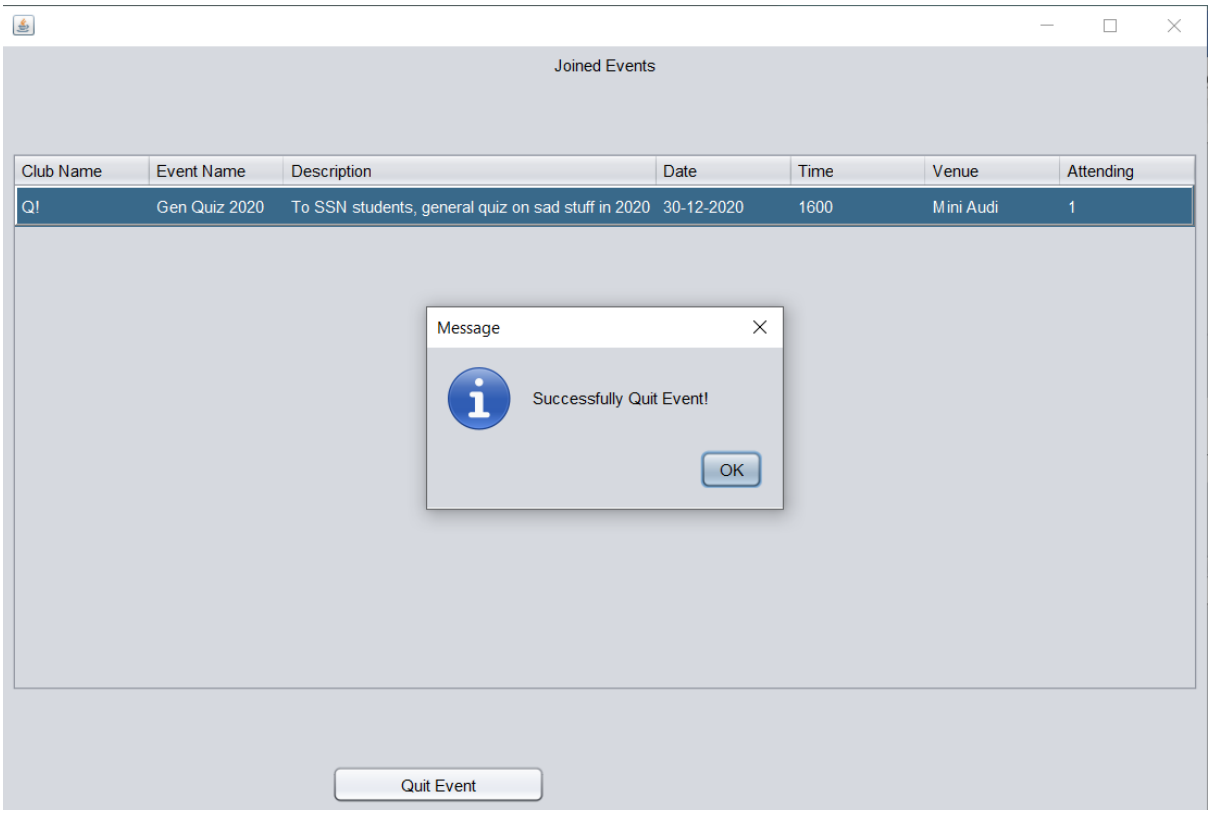
- Joined events are no longer displayed in **'View Events to Join'**

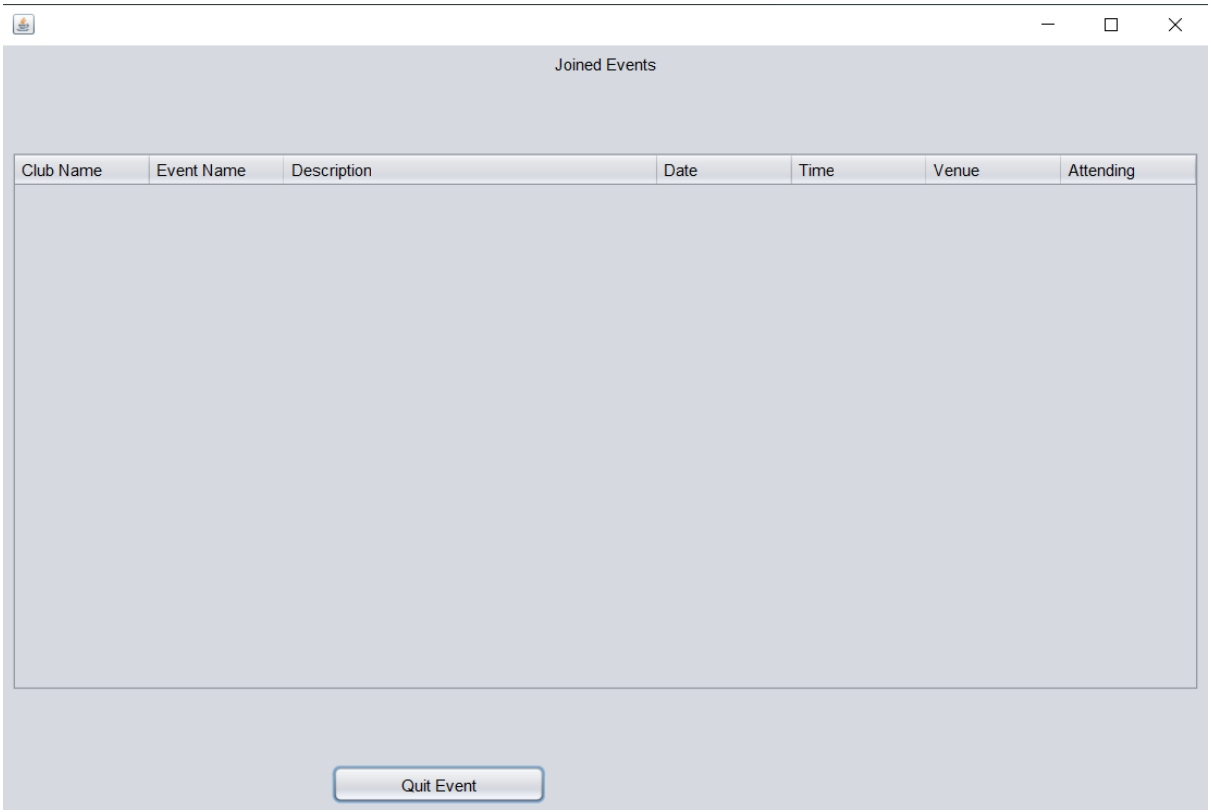


- **Browse Joined Events**

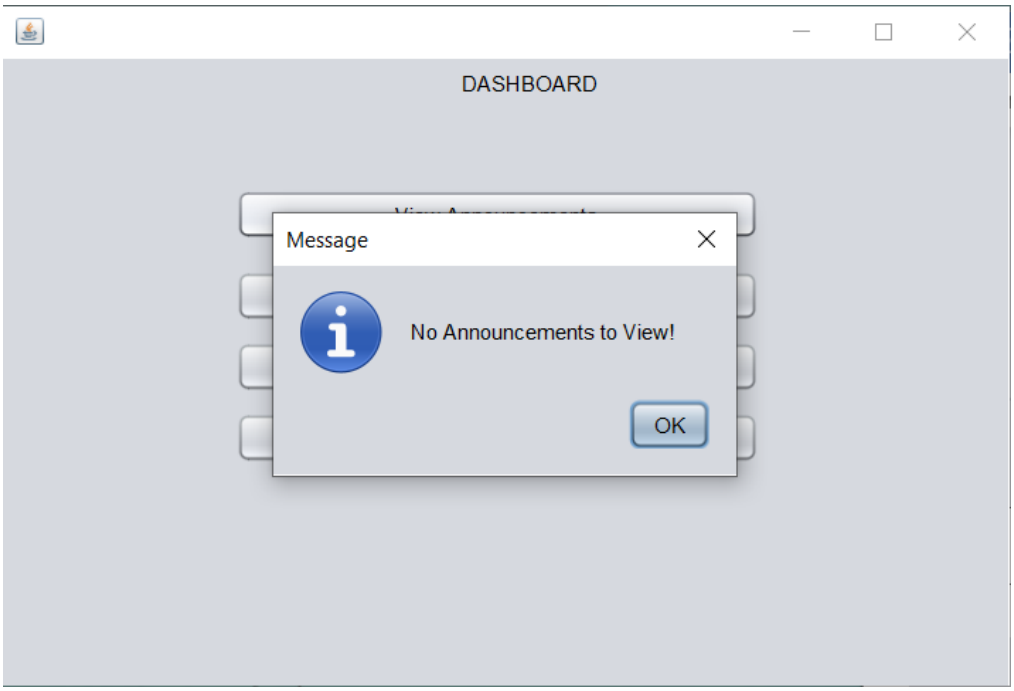


- Quitting an event in **'Browse Joined Events'**

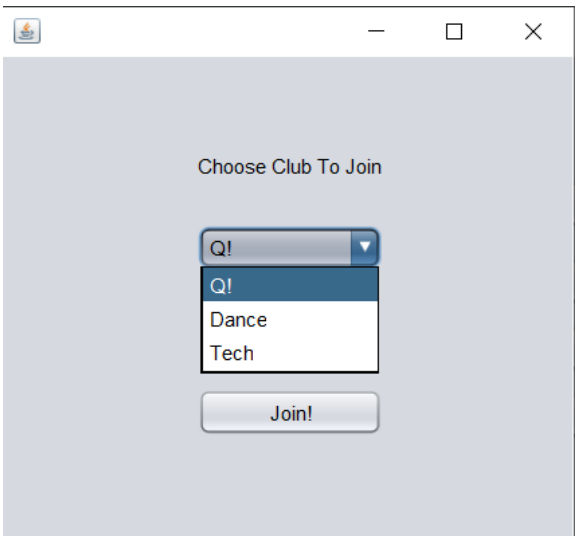




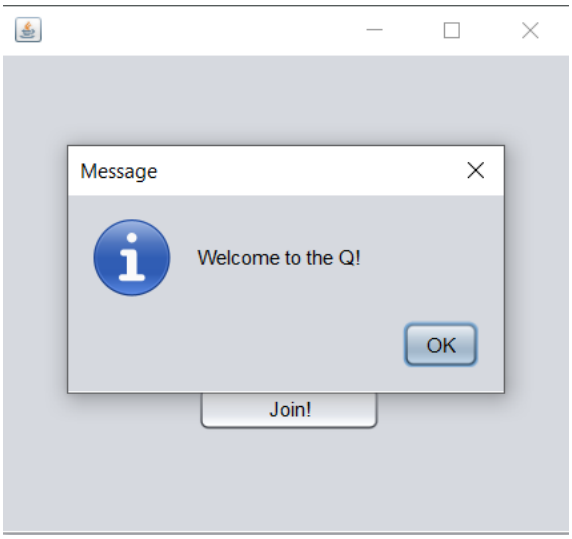
- **View Announcements**
  - Student can view announcements of the club he/she belongs to. The student does not belong to any club at the moment



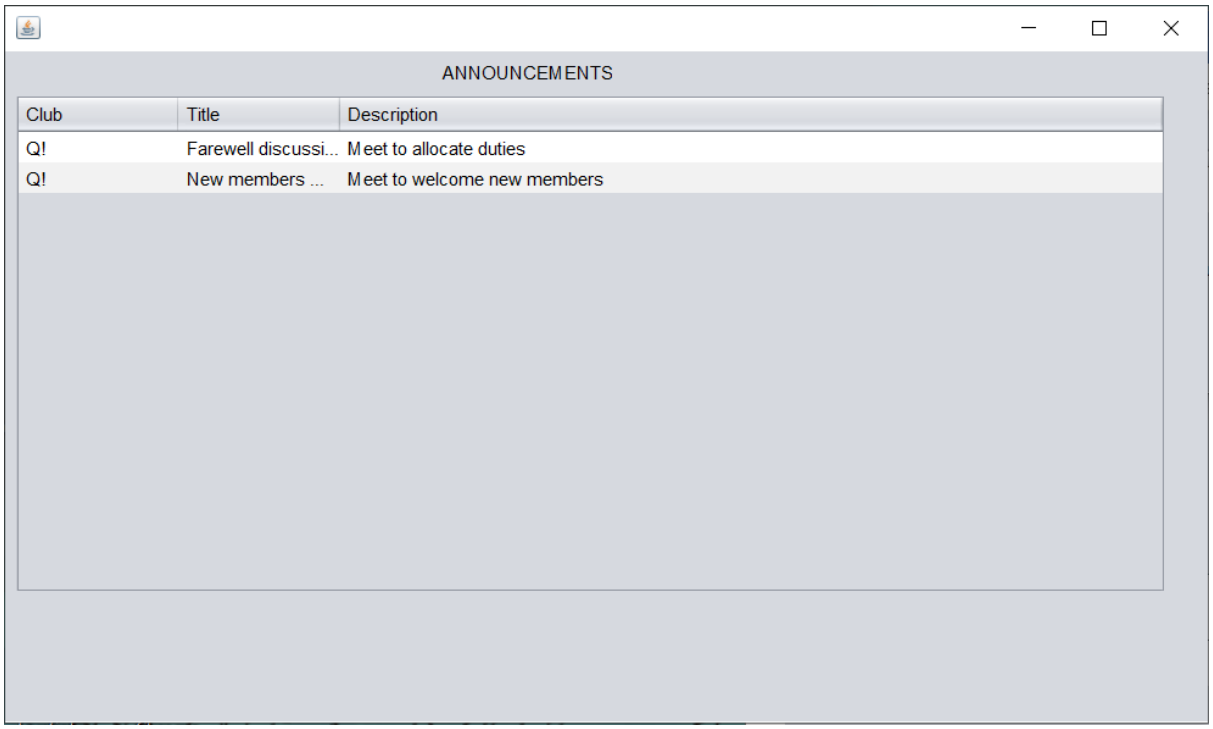
- **Join New Club**
  - Student is not a member of any club at the moment; all three clubs available



  - Joined Quiz Club- Q!



- **View Announcements**
  - Announcements of Q! Club shown



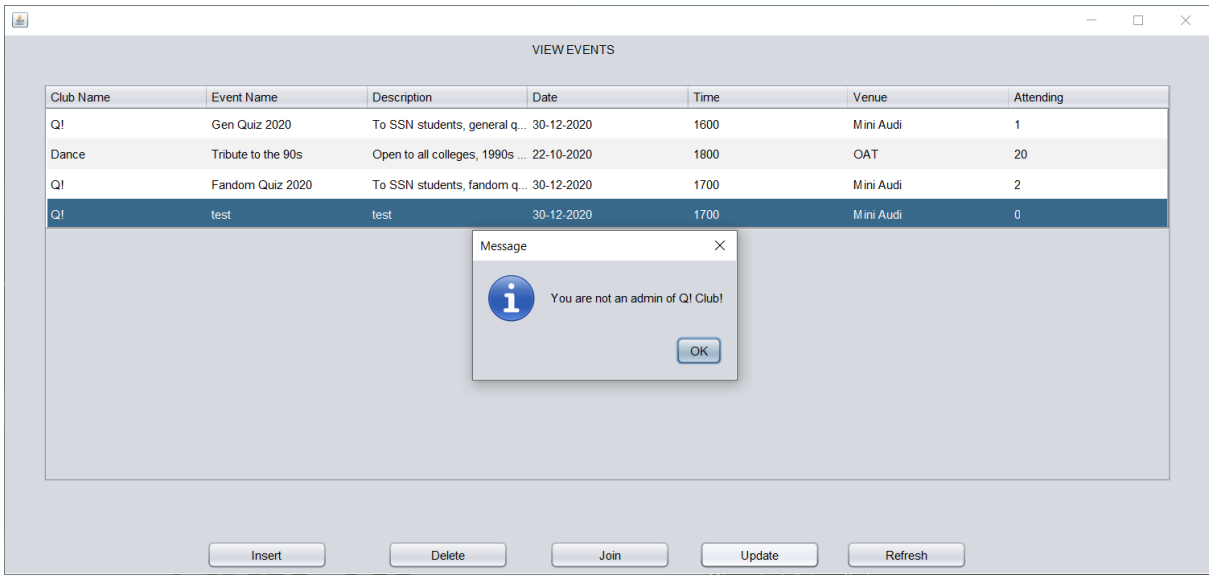
Admin/Club Head View

- Club Heads in Database

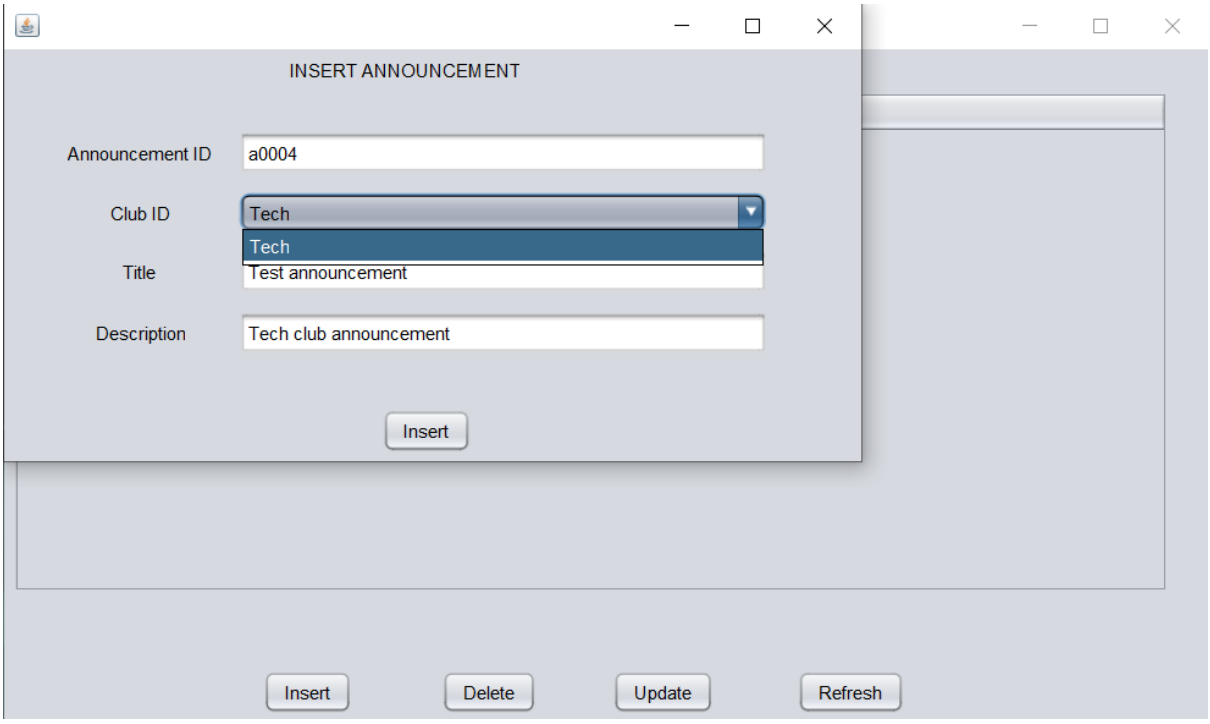
```
SQL> select * from Club;

CLUBI CLUBNAME             CLUBH MEMBERCOUNT
----- -------------------- ----- -----------
c0001 Q!                   s0002          15
c0002 Dance                s0003          20
c0003 Tech                 s0001          10
```

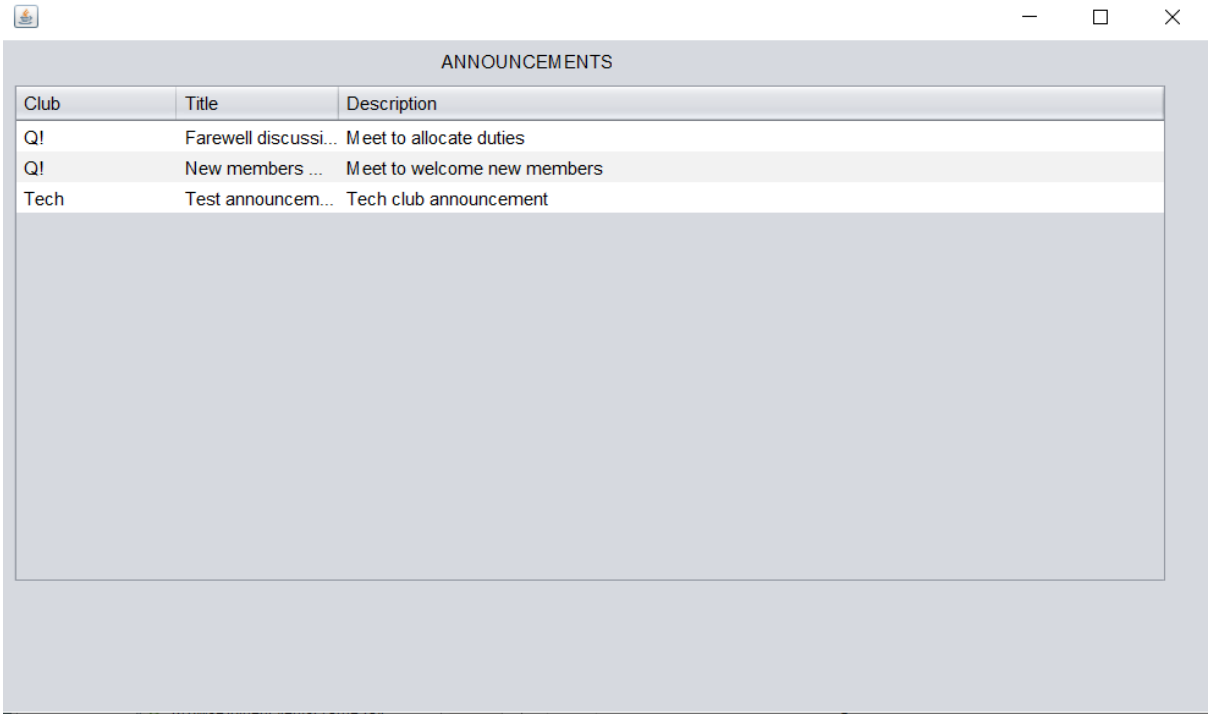
- Logged in as S0001- Tech Club Head
- Updating Club Information of other clubs



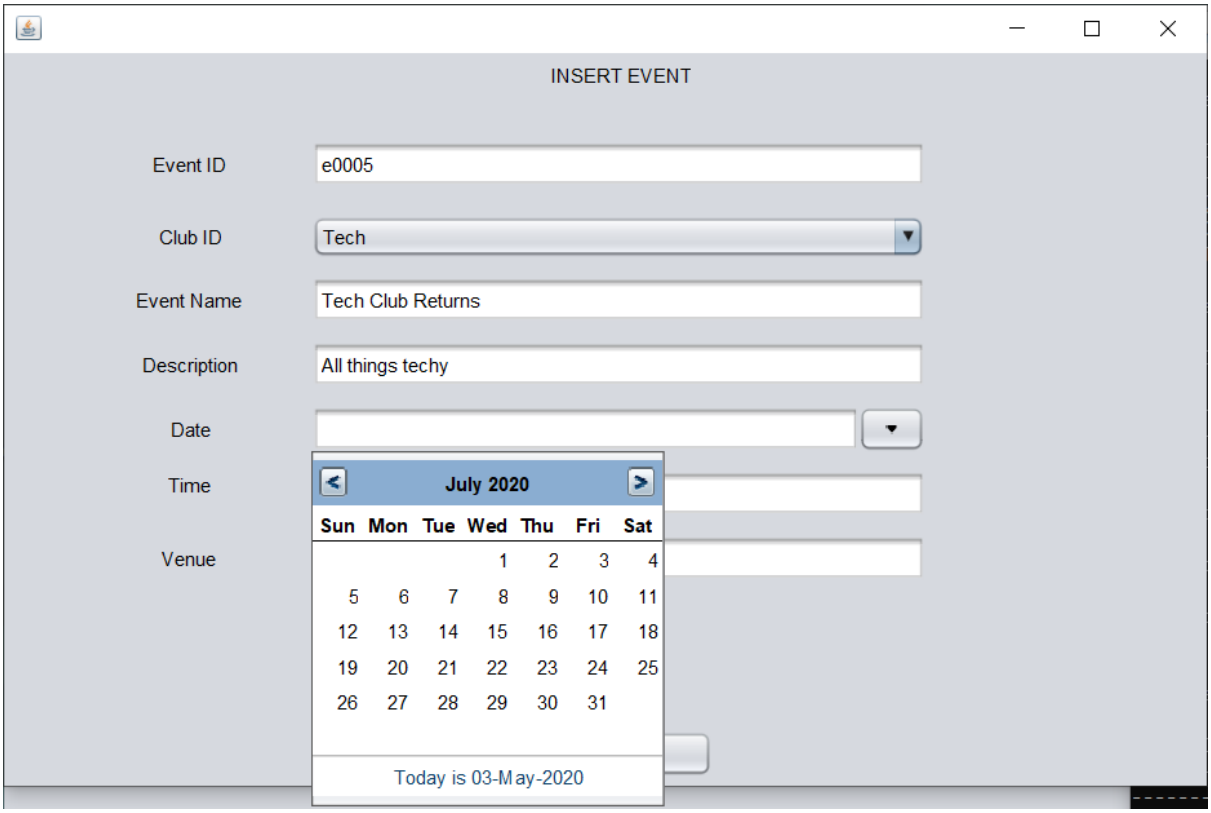
- Inserting Announcement
  - Only clubs that the user is Head of are available

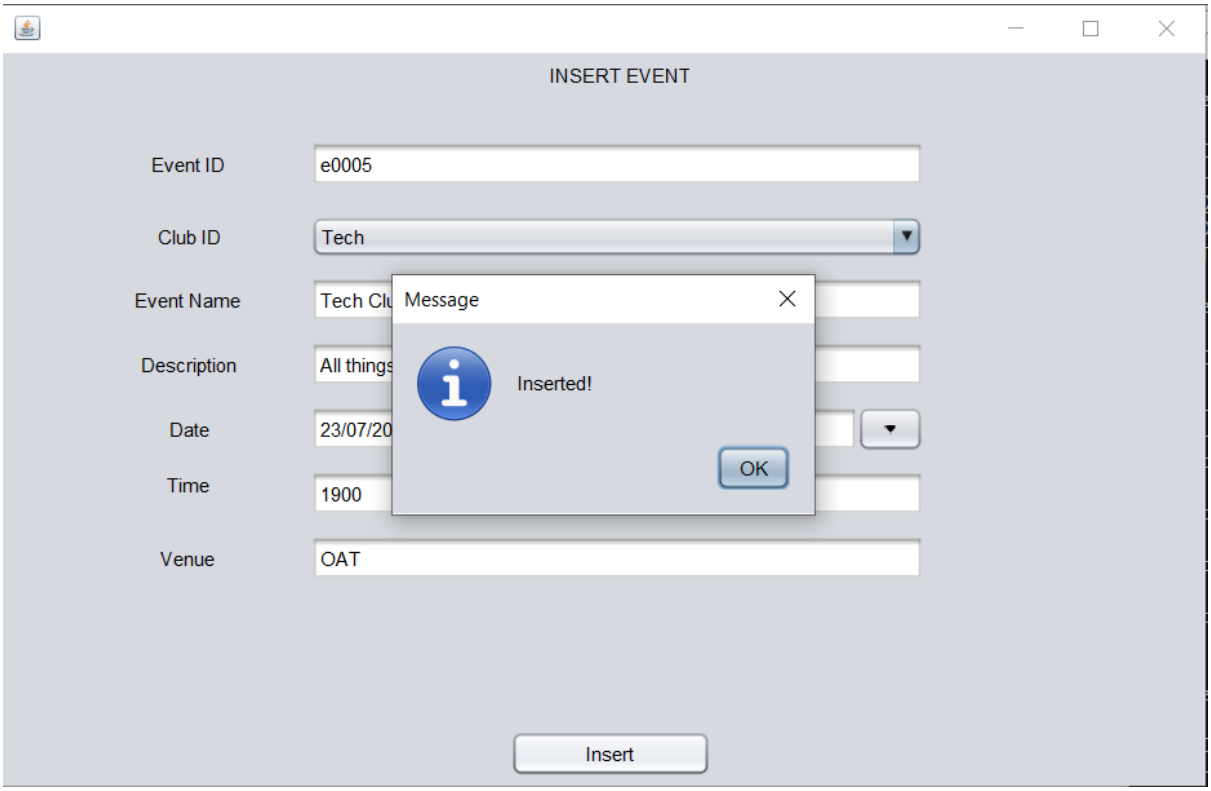


- The inserted announcement

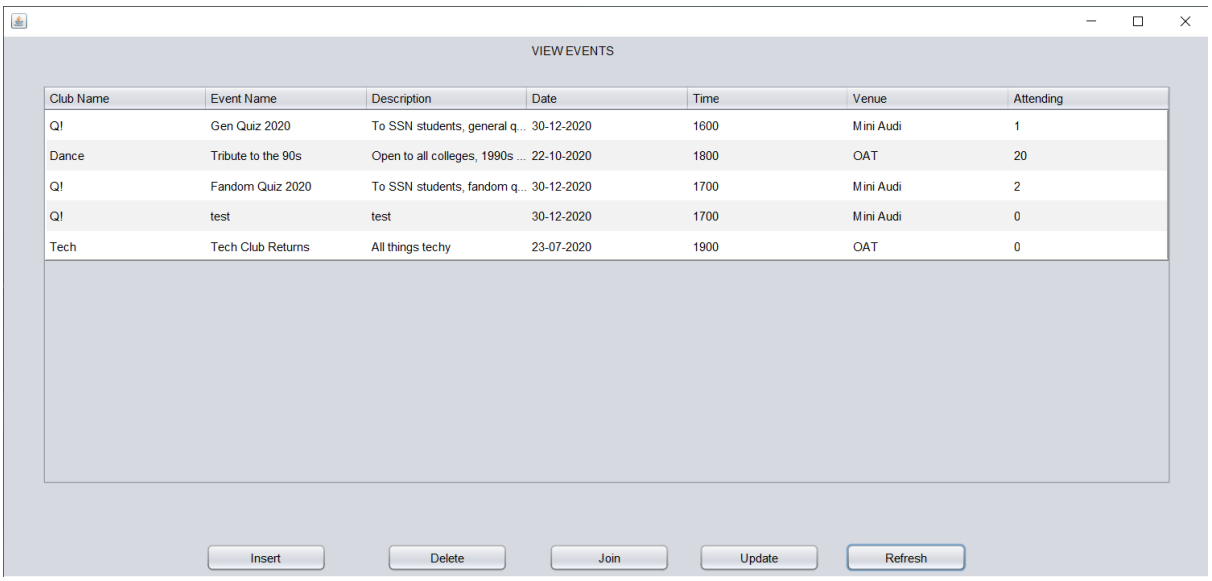


- Inserting new event



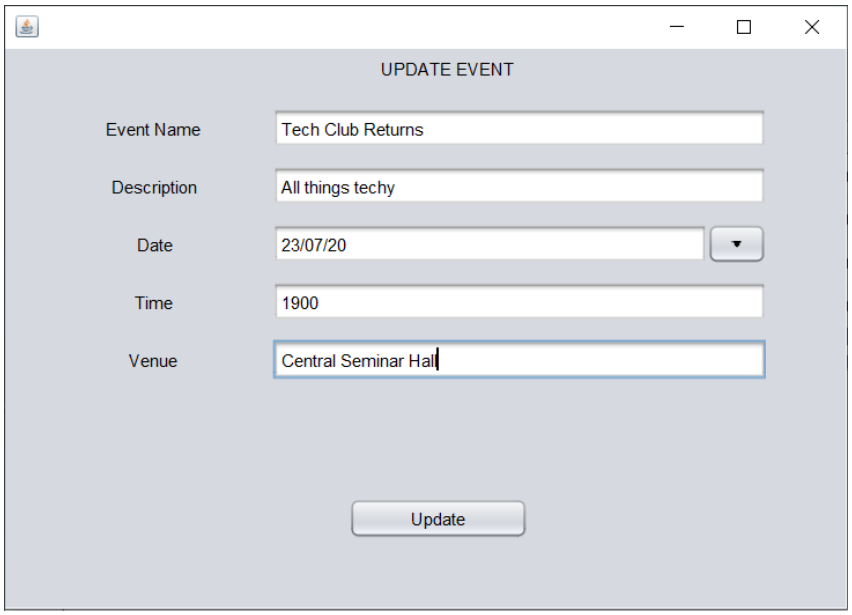


- Event added on clicking ‘Refresh’

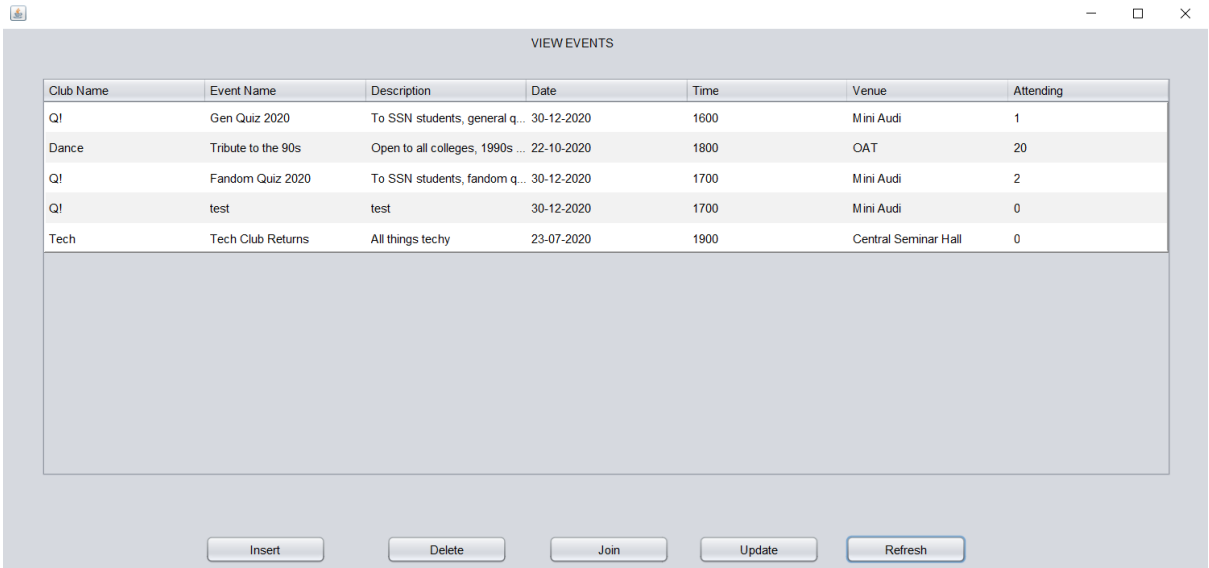


Club Name	Event Name	Description	Date	Time	Venue	Attending
Q!	Gen Quiz 2020	To SSN students, general q...	30-12-2020	1600	Mini Audi	1
Dance	Tribute to the 90s	Open to all colleges, 1990s ...	22-10-2020	1800	OAT	20
Q!	Fandom Quiz 2020	To SSN students, fandom q...	30-12-2020	1700	Mini Audi	2
Q!	test	test	30-12-2020	1700	Mini Audi	0
Tech	Tech Club Returns	All things techy	23-07-2020	1900	OAT	0

- Updating Event



- Updated Event information viewed on clicking ‘Refresh’



Club Name	Event Name	Description	Date	Time	Venue	Attending
Q!	Gen Quiz 2020	To SSN students, general q...	30-12-2020	1600	Mini Audi	1
Dance	Tribute to the 90s	Open to all colleges, 1990s ...	22-10-2020	1800	OAT	20
Q!	Fandom Quiz 2020	To SSN students, fandom q...	30-12-2020	1700	Mini Audi	2
Q!	test	test	30-12-2020	1700	Mini Audi	0
Tech	Tech Club Returns	All things techy	23-07-2020	1900	Central Seminar Hall	0

Source Code:

SQL Script

```
set echo on;
set linesize 200;
set serveroutput on;

drop table Attendees;
drop table Members;
drop table Announcement;
drop table Event;
drop table Club;
drop table Student;

create table Student (
 StudentID varchar2(5) constraint student_pk primary key ,
 Name varchar2(20) constraint chkstudent_n_nn not null,
 Branch varchar2(5) constraint chkstudent_b check (Branch in ( 'CSE',
'IT','MECH', 'BME', 'ECE', 'EEE', 'CIVIL', 'CHEM')),
 Year number(1) constraint chkstudent_year check (Year in (1,2,3,4)),
 Password varchar(20) constraint student_pw_nn not null      );

create table Club (
 ClubID varchar2(5) constraint club_pk primary key,
 ClubName varchar2(20) constraint chkclub_n_nn not null ,
 ClubHead varchar2(5) constraint club_fk references Student(StudentID) on
delete cascade,
 MemberCount number(4)      );

create table Event (
 EventID varchar2(5) constraint event_pk primary key,
 ClubID varchar2(5) constraint event_fk references Club(ClubID) on delete
cascade,
 EventName varchar2(20),
 Description varchar2(100),
 EventDate date constraint chkevent_d check ( extract(year from
EventDate)>=2020 ),
 EventTime varchar2(4) constraint chkevent_t check (EventTime between 0600 and
2100),
 Venue varchar2(20),
 AttendeesCount number(5)   );

create table Announcement (
 AnnouncementID varchar2(5) constraint announc_pk primary key,
 ClubID varchar2(5) constraint announc_fk references Club(ClubID) on delete
cascade,
 AName varchar2(20),
 Description varchar2(100)  );

create table Members (
 ClubID varchar2(5) constraint members_c_fk references Club(ClubID) on delete
cascade,
 StudentID varchar2(5) constraint members_s_fk references Student(StudentID) on
delete cascade,
 constraint members_pk primary key (ClubID, StudentID) );

create table Attendees (
 EventID varchar2(5) constraint attend_e_fk references Event(EventID) on delete
cascade ,
 StudentID varchar2(5) constraint attend_s_fk references Student(StudentID) on
delete cascade,
 constraint attend_pk primary key (EventID, StudentID) );
```

```
create or replace procedure inc_membercount(id IN Club.ClubID%Type) is
ct number;
cursor c is select count(*) from Members where ClubID=id;
begin
 ct := 0;
 open c;
 fetch c into ct;
 close c;
 update Club set MemberCount=ct where ClubID=id;
end;
/

create or replace procedure inc_attendeescount(id IN Event.EventID%Type) is
ct number;
cursor c is select count(*) from Attendees where EventID=id;
begin
 ct := 0;
 open c;
 fetch c into ct;
 close c;
 update Event set AttendeesCount= ct where EventID=id;
end;
/

insert into Student values('s0001', 'Akash','CSE', 2,'akash');
insert into Student values('s0002', 'Akshaya','CSE', 2,'akshaya');
insert into Student values('s0003', 'Bhavya','MECH', 2,'bhavya');

insert into Club values ( 'c0001','Q!','s0002', 1);
insert into Club values ( 'c0002','Dance','s0003', 1);
insert into Club values ( 'c0003','Tech','s0001', 1);

insert into Members values('c0001','s0002');
insert into Members values('c0002','s0003');
insert into Members values('c0003','s0001');

insert into Event values ( 'e0001','c0001','Gen Quiz 2020', 'To SSN students,
general quiz on sad stuff in 2020','30-dec-2020','1600','Mini Audi', 2);
insert into Event values ( 'e0002','c0002','Tribute to the 90s', 'Open to all
colleges, 1990s style','22-oct-2020','1800','OAT', 20);
insert into Event values ( 'e0003','c0001','Fandom Quiz 2020', 'To SSN
students, fandom quiz on TV shows','30-dec-2020','1700','Mini Audi', 2);
insert into Event values ( 'e0004','c0001','test', 'test','30-dec-
2020','1700','Mini Audi', 0);

insert into Announcement values ('a0001','c0001','New members meet','Meet to
welcome new members');
insert into Announcement values ('a0002','c0002','Dance off','1 on 1 dance
off');
insert into Announcement values ('a0003','c0001','Farewell discussion','Meet to
allocate duties');

select * from Student;
select * from Club;
select * from Event;
select * from Announcement;
select * from Members;
select * from Attendees;
```

Java Code Snippets:

********** **MAIN JAVA CLASS - Student_Management_System.java** **********

```
package student_management_system;

import java.sql.*;
import javax.swing.JOptionPane;
public class Student_Management_System {

 public static void main(String[] args) {
    new LoginFrame().setVisible(true);
 }}
```

********** **LoginFrame.java** **********

```
package student_management_system;

import java.sql.*;
import javax.swing.JOptionPane;

public class LoginFrame extends javax.swing.JFrame {
 public LoginFrame() {
  initComponents();
  try {
   conn = DriverManager.getConnection
("jdbc:oracle:thin:@//localhost:1522/orcl1.168.1.7","system","abc123");
   JOptionPane.showMessageDialog(this,"Connected to Oracle!");
  }
  catch(SQLException e) {
   JOptionPane.showMessageDialog(this,"Connection to Oracle FAILED!");
   System.exit(2);
  }}

 private void newuserButtonActionPerformed(java.awt.event.ActionEvent evt) {
  new NewUserFrame(conn).setVisible(true);
 }

 private void loginButtonActionPerformed(java.awt.event.ActionEvent evt) {
  try {
   String sql = "select StudentID,Password from student where StudentID =?";
   ps = conn.prepareStatement(sql);
   ps.setString(1, usernameField.getText());
   rs = ps.executeQuery();

   if(rs.next()) {
    if(rs.getString("Password").equals(passwordField.getText())) {
     JOptionPane.showMessageDialog(this,"Login Successful!");
     new DashboardFrame(rs.getString("StudentID"),conn).setVisible(true);
     this.setVisible(false); }
    else
     JOptionPane.showMessageDialog(this, "Invalid username/password!");}}

  catch(SQLException e){
   JOptionPane.showMessageDialog(this,"Invalid username!");
  } }

 public static void main(String args[]) {
  java.awt.EventQueue.invokeLater(new Runnable() {
   public void run() {
    new LoginFrame().setVisible(true);
   }
```

```
  });}
 Connection conn;
 PreparedStatement ps;
 ResultSet rs;     }
```

********** **NewUserFrame.java** **********

```
package student_management_system;

import java.sql.*;
import javax.swing.JOptionPane;

public class NewUserFrame extends javax.swing.JFrame {

 public NewUserFrame(Connection conn) {
  initComponents();
  this.conn=conn;}

 private void yearComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
}

 private void registerButtonActionPerformed(java.awt.event.ActionEvent evt) {
  try {
   String sql = "insert into Student values(?,?,?,?,?)";
   ps = conn.prepareStatement(sql);
   ps.setString(1,studentidField.getText());
   ps.setString(2,nameField.getText());
   ps.setString(3,(String)branchComboBox.getSelectedItem());
   ps.setString(4,(String)yearComboBox.getSelectedItem());
   ps.setString(5,passwordField.getText());
   ps.execute();
   JOptionPane.showMessageDialog(this, "Inserted!");
  }
  catch(SQLException ex){
   JOptionPane.showMessageDialog(this,ex.getMessage());
  }}

 private void studentidFieldKeyTyped(java.awt.event.KeyEvent evt) {
  if(studentidField.getText().length()>=5) {
   evt.consume();
  }}

 private void nameFieldKeyTyped(java.awt.event.KeyEvent evt) {
  if(nameField.getText().length()>=20) {
   evt.consume();
  }}

 private void passwordFieldKeyTyped(java.awt.event.KeyEvent evt) {
  if(passwordField.getText().length()>=20) {
   evt.consume();
  }}

 public static void main(String args[]) {
  java.awt.EventQueue.invokeLater(new Runnable() {
   public void run() {
      //new NewUserFrame().setVisible(true);
   }});}
 Connection conn;
 PreparedStatement ps;
}
```

********** **DashboardFrame.java** **********

```
package student_management_system;

import java.sql.*;
import javax.swing.JOptionPane;

public class DashboardFrame extends javax.swing.JFrame {

 public DashboardFrame() {
  initComponents();
  try {
   conn = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1521:orcl","system","1234");
  }
  catch(SQLException e) {
      JOptionPane.showMessageDialog(this, e);
  }}

 public DashboardFrame(String id,Connection conn) {
  this.id = id;
  initComponents();
  try {
   this.conn=conn;
   String sql = "select ClubHead from Club where ClubHead=?";
   ps = conn.prepareStatement(sql);
   ps.setString(1,id);
   rs = ps.executeQuery();
   if(rs.next())
    this.admin = true;
   else
    this.admin = false;
  }
  catch(SQLException e) {
      JOptionPane.showMessageDialog(this, e);
  }}

 private void viewEventsToJoinButtonActionPerformed(java.awt.event.ActionEvent evt)
{
  new ViewEventsToJoinFrame(id,admin,conn).setVisible(true);
 }

 private void browseJoinedEventsButtonActionPerformed(java.awt.event.ActionEvent
evt) {
  try {
   String sql = "select count(*) from Attendees where StudentID=?";
   ps = conn.prepareStatement(sql);
   ps.setString(1,id);
   rs = ps.executeQuery();
   rs.next();
   if(rs.getInt(1)>0) {
    new BrowseJoinedEventsFrame(id,conn).setVisible(true);
   }
   else
    JOptionPane.showMessageDialog(this, "You have not joined any events!");
  }
  catch(SQLException e){
   JOptionPane.showMessageDialog(this, e);
  }}

 private void joinNewClubButtonActionPerformed(java.awt.event.ActionEvent evt) {
  try {
   String sql = "select count(*) as ct from ((select ClubID from Club) minus
(select ClubID from Members where StudentID=?))";
```

```
   ps = conn.prepareStatement(sql);
   ps.setString(1, id);
   rs = ps.executeQuery();
   if(rs.next()) {
    if (rs.getInt(1) > 0) {
     new JoinNewClubFrame(id,conn).setVisible(true);
    } else {
     JOptionPane.showMessageDialog(this, "No new clubs to join!");
    }}
   else
    System.out.println("Error"); }
  catch(SQLException e) {
   JOptionPane.showMessageDialog(this, e);
  }}

 private void viewAnnouncementsButtonActionPerformed(java.awt.event.ActionEvent
evt) {
  try {
   String sql = "select count(*) as ct from Announcement a,Members m where
a.ClubID=m.ClubID and m.StudentID=?";
   ps = conn.prepareStatement(sql);
   ps.setString(1, id);
   rs = ps.executeQuery();
   rs.next();
   if(admin || rs.getInt("ct")>0){
    new viewAnnouncementsFrame(id,admin,conn).setVisible(true); }
   else
    JOptionPane.showMessageDialog(this, "No Announcements to View!"); }
  catch(SQLException e) {
   JOptionPane.showMessageDialog(this, e);
  }}

 public static void main(String args[]) {
  java.awt.EventQueue.invokeLater(new Runnable() {
   public void run() {
    new DashboardFrame().setVisible(true);
   }});}

 String id;
 Connection conn;
 ResultSet rs;
 PreparedStatement ps;
 Boolean admin; }
```

********** **viewAnnouncementsFrame.java** **********

```
package student_management_system;

import java.sql.*;
import java.util.HashMap;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

public class viewAnnouncementsFrame extends javax.swing.JFrame {

 public viewAnnouncementsFrame() {
  initComponents();
 }

 public viewAnnouncementsFrame(String id,Boolean admin, Connection conn) {
  this.id = id;
  this.admin = admin;
  this.conn = conn;
```

```
  hmap = new HashMap();
  cmap = new HashMap();
  initComponents();
  refresh();
  if(!admin){
   this.updateButton.setVisible(false);
   this.deleteButton.setVisible(false);
   this.insertButton.setVisible(false);
   this.refreshButton.setVisible(false);       } }

 private void deleteButtonActionPerformed(java.awt.event.ActionEvent evt) {
  int row = announcementsTable.getSelectedRow();

  if(row==-1) {
   JOptionPane.showMessageDialog(this, "Select An Announcement To Delete!");
   return; }
  else {
   try {
    String sql = "delete from Announcement where AnnouncementID=? and ClubID in
(select ClubID from Club where ClubHead=?)";
    ps = conn.prepareStatement(sql);
    ps.setString(1, hmap.get(row));
    ps.setString(2, id);
    int ct = ps.executeUpdate();

    if(ct>0)
     JOptionPane.showMessageDialog(this, "Deleted!");
    else
     JOptionPane.showMessageDialog(this, "You are not the admin of " +
(String)announcementsTable.getValueAt(row, 0) + " Club!");
    refresh();
   } catch (SQLException e) {
    JOptionPane.showMessageDialog(this, e); } } }

 private void refreshButtonActionPerformed(java.awt.event.ActionEvent evt) {
  refresh(); }

 private void updateButtonActionPerformed(java.awt.event.ActionEvent evt) {
  int row = announcementsTable.getSelectedRow();

  if(row==-1){
   JOptionPane.showMessageDialog(this, "Select An Announcement To Update!");    }
  else {
   try {
    String clubname = (String)announcementsTable.getValueAt(row,0);
    String clubid = cmap.get(clubname);
    String sql = "select ClubID from Club where ClubHead=?";
    ps = conn.prepareStatement(sql);
    ps.setString(1,id);
    rs = ps.executeQuery();
    boolean isAdmin = false;

    while(rs.next()){
     if(rs.getString(1).equals(clubid)) {
      isAdmin = true;
      break; }}
    if(isAdmin)
     new UpdateAnnouncementFrame(hmap.get(row),conn).setVisible(true);
    else
     JOptionPane.showMessageDialog(this,"You are not an admin of " + clubname + "
Club!");
   } catch (SQLException ex) {
    JOptionPane.showMessageDialog(this,ex);   }     }     }
```

```
 private void insertButtonActionPerformed(java.awt.event.ActionEvent evt) {
  new InsertAnnouncementFrame(id,conn).setVisible(true); }

 private void refresh() {
  DefaultTableModel model = (DefaultTableModel) announcementsTable.getModel();
  int rows = model.getRowCount();
  for (int i = rows - 1; i >= 0; i--) {
   model.removeRow(i); }
  try {
   String sql = "select a.AnnouncementID,c.ClubName,a.AName,a.Description,c.ClubID
from Announcement a,Club c where c.ClubID=a.ClubID and a.ClubID in (select ClubID
from Members where StudentID=?)";
   ps = conn.prepareStatement(sql);
   ps.setString(1, id);
   rs = ps.executeQuery();
   int ct = 0;
   while (rs.next()) {
    String aid = rs.getString(1);
    String cname = rs.getString(2);
    String aname = rs.getString(3);
    String desc = rs.getString(4);

    hmap.put(ct++, aid);
    cmap.put(cname, rs.getString(5));
    model.addRow(new Object[]{cname, aname, desc}); }
   announcementsTable.setModel(model);
  } catch (SQLException e) {
   JOptionPane.showMessageDialog(this, e);    }     }

 public static void main(String args[]) {
  java.awt.EventQueue.invokeLater(new Runnable() {
   public void run() {
    new viewAnnouncementsFrame().setVisible(true);  } }); }

 String id;
 HashMap<Integer,String> hmap;
 HashMap<String,String> cmap;
 Connection conn;
 ResultSet rs;
 PreparedStatement ps;
 Boolean admin;        }
```

********** **ViewEventsToJoinFrame.java** **********

```
package student_management_system;

import java.sql.*;
import java.text.SimpleDateFormat;
import java.util.HashMap;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

public class ViewEventsToJoinFrame extends javax.swing.JFrame {
 public ViewEventsToJoinFrame() {
  initComponents();
 }

 public ViewEventsToJoinFrame(String id,Boolean admin,Connection conn) {
  this.id = id;
  this.admin = admin;
  this.conn = conn;
  emap = new HashMap();
  cmap = new HashMap();
```

```
  initComponents();

  refresh();
  if(!admin) {
   insertButton.setVisible(false);
   deleteButton.setVisible(false);
   updateButton.setVisible(false);
   refreshButton.setVisible(false);     }     }

 private void refreshButtonActionPerformed(java.awt.event.ActionEvent evt) {
  refresh();     }

 private void deleteButtonActionPerformed(java.awt.event.ActionEvent evt) {
  int row = eventsTable.getSelectedRow();

  if(row==-1)
   JOptionPane.showMessageDialog(this,"Select An Event To Delete!");
  else {
   try {
    String sql = "delete from Event where EventID=? and ClubID in (select ClubID
from Club where ClubHead=?)";
    ps = conn.prepareStatement(sql);
    ps.setString(1,emap.get(row));
    ps.setString(2,id);
    int ct = ps.executeUpdate();

    if(ct>0)
     JOptionPane.showMessageDialog(this,"Deleted!");
    else
     JOptionPane.showMessageDialog(this,"You are not an admin of " +
(String)eventsTable.getValueAt(row, 0) + " Club!");
    refresh(); }

   catch(SQLException e) {
    JOptionPane.showMessageDialog(this,e);    } } }

 private void updateButtonActionPerformed(java.awt.event.ActionEvent evt) {
  int row = eventsTable.getSelectedRow();

  if(row==-1)
   JOptionPane.showMessageDialog(this, "Choose Am Event To Update!");
  else {
   try {
    String clubname = (String)eventsTable.getValueAt(row,0);
    String clubid = cmap.get(clubname);
    String sql = "select ClubID from Club where ClubHead=?";
    ps = conn.prepareStatement(sql);
    ps.setString(1,id);
    rs = ps.executeQuery();

    boolean isAdmin = false;

    while(rs.next()){
     if(rs.getString(1).equals(clubid)) {
      isAdmin = true;
      break;     }     }
    if(isAdmin)
     new UpdateEventFrame(emap.get(row),conn).setVisible(true);
    else
     JOptionPane.showMessageDialog(this,"You are not an admin of " + clubname + "
Club!");
   } catch (SQLException ex) {
    JOptionPane.showMessageDialog(this,ex);   }     }      }
```

```
 private void insertButtonActionPerformed(java.awt.event.ActionEvent evt) {
  new InsertEventFrame(id,conn).setVisible(true);   }

 private void joinButtonActionPerformed(java.awt.event.ActionEvent evt) {
  int row = eventsTable.getSelectedRow();

  if(row==-1) {
   JOptionPane.showMessageDialog(this,"Select An Event To Attend!"); }
  else {
   try {
    String sql = "insert into Attendees values(?,?)";
    ps = conn.prepareStatement(sql);
    ps.setString(1, emap.get(row));
    ps.setString(2,id);
    ps.executeQuery();

    String call = "call inc_attendeescount(?)";
    cs = conn.prepareCall(call);
    cs.setString(1, emap.get(row));
    cs.execute();

    JOptionPane.showMessageDialog(this,"Successfully Joined!");
    refresh();   }
   catch(SQLException e) {
    JOptionPane.showMessageDialog(this,e);   }  } }

 private void refresh() {
  try {
   DefaultTableModel model = (DefaultTableModel) eventsTable.getModel();
   int rows = model.getRowCount();
   for (int i = rows - 1; i >= 0; i--) {
    model.removeRow(i);
   }
   String sql;
   if(admin)
    sql = "select c.ClubName, e.EventName, e.Description,  e.EventDate,
e.EventTime, e.Venue, e.AttendeesCount, e.EventID,c .ClubID from Event e,Club c
where c.ClubID=e.ClubID";
   else
    sql = "select c.ClubName, e.EventName, e.Description, e.EventDate, e.EventTime,
e.Venue, e.AttendeesCount,e.EventID,c.ClubID from Event e,Club c where
c.ClubID=e.ClubID and e.EventID not in (select EventID from Attendees where
StudentID=?)";
   ps = conn.prepareStatement(sql);
   if(!admin)
    ps.setString(1, id);
   rs = ps.executeQuery();
   int ct = 0;
   while (rs.next()) {
    String clubname = rs.getString(1);
    String eventname = rs.getString(2);
    String desc = rs.getString(3);
    Date date = rs.getDate(4);
    SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");
    Integer time = rs.getInt(5);
    String venue = rs.getString(6);
    Integer act = rs.getInt(7);
    String eventid = rs.getString(8);
    emap.put(ct++,eventid);
    cmap.put(clubname, rs.getString(9));
    model.addRow(new Object[]{clubname, eventname, desc, sdf.format(date), time,
venue, act});    }
   eventsTable.setModel(model);
  } catch (SQLException e) {
```

```
   JOptionPane.showMessageDialog(this, e);    } }

 public static void main(String args[]) {
  java.awt.EventQueue.invokeLater(new Runnable() {
   public void run() {
    new ViewEventsToJoinFrame().setVisible(true);    }    });   }

 String id;
 Connection conn;
 ResultSet rs;
 PreparedStatement ps;
 CallableStatement cs;
 Boolean admin;
 HashMap<Integer,String> emap;
 HashMap<String,String> cmap;     }
```

********** **JoinNewClubFrame.java** **********

```
package student_management_system;

import java.sql.*;
import java.util.HashMap;
import javax.swing.JOptionPane;

public class JoinNewClubFrame extends javax.swing.JFrame {
 public JoinNewClubFrame() {
  initComponents();    }

 public JoinNewClubFrame(String id,Connection conn) {
  this.id = id;
  initComponents();
  try {
   this.conn=conn;
   clubComboBox.removeAllItems();
   hmap = new HashMap();
   String sql = "select ClubID,ClubName from Club where ClubID not in (select
ClubID from Members where StudentID=?)";
   ps = conn.prepareStatement(sql);
   ps.setString(1,id);
   rs = ps.executeQuery();
   while(rs.next()) {
    hmap.put(rs.getString("ClubName"),rs.getString("ClubID"));
    clubComboBox.addItem(rs.getString("ClubName"));  }    }
  catch(SQLException e) {
   JOptionPane.showMessageDialog(this, e);    } }

 private void joinButtonActionPerformed(java.awt.event.ActionEvent evt) {
  try {
   String sql ="insert into Members values(?,?)";
   ps = conn.prepareStatement(sql);
   ps.setString(1,hmap.get((String)clubComboBox.getSelectedItem()));
   ps.setString(2,id);
   ps.executeQuery();

   String call = "call inc_membercount(?)";
   cs = conn.prepareCall(call);
   cs.setString(1, hmap.get((String) clubComboBox.getSelectedItem()));
   cs.execute();
   JOptionPane.showMessageDialog(this,"Welcome to the " + (String)
clubComboBox.getSelectedItem ());  }
  catch(SQLException e) {
   JOptionPane.showMessageDialog(this, e);     }    }
```

```
 public static void main(String args[]) {
  java.awt.EventQueue.invokeLater(new Runnable() {
   public void run() {
    new JoinNewClubFrame().setVisible(true);
   }  });   }

 String id;
 Connection conn;
 ResultSet rs;
 PreparedStatement ps;
 CallableStatement cs;
 HashMap<String,String> hmap;       }
```

********** **BrowseJoinedEventsFrame.java** **********

```
package student_management_system;

import java.sql.*;
import java.text.SimpleDateFormat;
import java.util.HashMap;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;

public class BrowseJoinedEventsFrame extends javax.swing.JFrame {

 public BrowseJoinedEventsFrame() {
  initComponents();    }

 public BrowseJoinedEventsFrame(String id,Connection conn){
  initComponents();
  emap = new HashMap();
  this.id=id;
  this.conn=conn;
  refresh();     }

 private void quitEventButtonActionPerformed(java.awt.event.ActionEvent evt) {
  int row = eventsTable.getSelectedRow();

  if(row==-1) {
   JOptionPane.showMessageDialog(this, "Select An Event to Quit!");
   return;  }

  try{
   String sql = "delete from Attendees where EventID=? and StudentID=?";
   ps = conn.prepareStatement(sql);
   ps.setString(1,emap.get(row));
   System.out.println(emap.get(row));
   ps.setString(2, id);
   int ct = ps.executeUpdate();

   if(ct>0)
    JOptionPane.showMessageDialog(this, "Successfully Quit Event!");
   else
    JOptionPane.showMessageDialog(this, "Could Not Quit Event!");    }
  catch(SQLException e){
   JOptionPane.showMessageDialog(this, e);    }
  refresh();     }

 private void refresh() {
  try{
   DefaultTableModel model = (DefaultTableModel) eventsTable.getModel();
   int rows = model.getRowCount();
   for (int i = rows - 1; i >= 0; i--) {
```

```
    model.removeRow(i);      }
   String sql = "select
c.ClubName,e.EventName,e.Description,e.EventDate,e.EventTime,e.Venue,e.AttendeesCou
nt,e.EventID from Event e,Club c where c.ClubID=e.ClubID and e.EventID in (select
EventID from Attendees where StudentID=?)";
   ps = conn.prepareStatement(sql);
   ps.setString(1, id);
   rs = ps.executeQuery();
   int rowct=0;
   while (rs.next()) {
    String clubname = rs.getString(1);
    String eventname = rs.getString(2);
    String desc = rs.getString(3);
    Date date = rs.getDate(4);
    SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");
    Integer time = rs.getInt(5);
    String venue = rs.getString(6);
    Integer ct = rs.getInt(7);
    emap.put(rowct++,rs.getString(8));
    model.addRow(new Object[]{clubname, eventname, desc, sdf.format(date), time,
venue, ct});     }
   eventsTable.setModel(model);   }
  catch(SQLException e) {
   JOptionPane.showMessageDialog(this,e);     }     }

 public static void main(String args[]) {
  java.awt.EventQueue.invokeLater(new Runnable() {
   public void run() {
    new BrowseJoinedEventsFrame().setVisible(true);  }    });   }

 String id;
 Connection conn;
 ResultSet rs;
 PreparedStatement ps;
 HashMap<Integer,String> emap;    }
```

********** **InsertAnnouncementFrame.java** **********

```
package student_management_system;

import java.sql.*;
import java.util.HashMap;
import javax.swing.JOptionPane;

public class InsertAnnouncementFrame extends javax.swing.JFrame {

 public InsertAnnouncementFrame() {
  initComponents();    }

 public InsertAnnouncementFrame(String id, Connection conn) {
  this.id = id;
  this.conn = conn;
  hmap = new HashMap();
  initComponents();
  clubIdComboBox.removeAllItems();
  try {
   String sql = "select ClubName,ClubID from Club where ClubID in (select i.ClubID
from Club i where i.ClubHead=?)";
   ps = conn.prepareStatement(sql);
   ps.setString(1,id);
   rs = ps.executeQuery();
   while(rs.next()) {
    hmap.put(rs.getString(1),rs.getString(2));
```

```
    clubIdComboBox.addItem(rs.getString(1));
   }  }
  catch(SQLException e) {
   JOptionPane.showMessageDialog(this,e);      }    }

 private void insertButtonActionPerformed(java.awt.event.ActionEvent evt) {
  try {
   String sql = "insert into Announcement values(?,?,?,?)";
   ps = conn.prepareStatement(sql);
   ps.setString(1, aidTextField.getText());
   ps.setString(2,hmap.get((String)clubIdComboBox.getSelectedItem()));
   ps.setString(3,atitleTextField.getText());
   ps.setString(4,descTextField.getText());

   ps.executeQuery();
   JOptionPane.showMessageDialog(this,"Inserted!");
   this.setVisible(false);
  }
  catch(SQLException e) {
   JOptionPane.showMessageDialog(this,e);     } }

 public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
   public void run() {
    new InsertAnnouncementFrame().setVisible(true); }     });   }

 String id;
 Connection conn;
 ResultSet rs;
 PreparedStatement ps;
 HashMap<String,String> hmap;           }
```

********** **InsertEventFrame.java** **********

```
package student_management_system;

import java.sql.*;
import java.util.HashMap;
import java.text.SimpleDateFormat;
import javax.swing.JOptionPane;

public class InsertEventFrame extends javax.swing.JFrame {
 public InsertEventFrame() {
  initComponents();    }

 public InsertEventFrame(String id, Connection conn) {
  this.id = id;
  this.conn = conn;
  initComponents();
  hmap = new HashMap();
  sdf = new SimpleDateFormat("dd-MMM-yyyy");
  clubIdComboBox.removeAllItems();
  try {
   String sql = "select ClubName,ClubID from Club where ClubID in (select i.ClubID
from Club i where i.ClubHead=?)";
   ps = conn.prepareStatement(sql);
   ps.setString(1, id);
   rs = ps.executeQuery();
   while (rs.next()) {
    hmap.put(rs.getString(1), rs.getString(2));
    clubIdComboBox.addItem(rs.getString(1));  } }
 catch (SQLException e) {
   JOptionPane.showMessageDialog(this, e);     }    }
```

```
 private void insertButtonActionPerformed(java.awt.event.ActionEvent evt) {
  try {
   String sql = "insert into Event values(?,?,?,?,?,?,?,0)";
   ps = conn.prepareStatement(sql);
   ps.setString(1, eventIdTextField.getText());
   ps.setString(2,hmap.get((String)clubIdComboBox.getSelectedItem()));
   ps.setString(3, eventNameTextField.getText());
   ps.setString(4,descTextField.getText());
   ps.setString(5,sdf.format(datePicker.getDate()));
   ps.setString(6, timeTextField.getText());
   ps.setString(7, venueTextField.getText());
   rs = ps.executeQuery();
   rs.next();

   JOptionPane.showMessageDialog(this, "Inserted!");
   this.setVisible(false);
  } catch (SQLException e) {
   JOptionPane.showMessageDialog(this, e);    }     }

 public static void main(String args[]) {
  java.awt.EventQueue.invokeLater(new Runnable() {
   public void run() {
    new InsertEventFrame().setVisible(true);  }});}

 String id;
 Connection conn;
 ResultSet rs;
 PreparedStatement ps;
 SimpleDateFormat sdf;
 HashMap<String,String> hmap;      }
```

********** **UpdateAnnouncementFrame.java** **********

```
package student_management_system;

import java.sql.*;
import javax.swing.JOptionPane;

public class UpdateAnnouncementFrame extends javax.swing.JFrame {

 public UpdateAnnouncementFrame() {
  initComponents();
 }

 public UpdateAnnouncementFrame(String aid,Connection conn) {
  this.aid = aid;
  this.conn = conn;
  initComponents();

  try {
   String sql = "select AName,Description from Announcement where
AnnouncementID=?";
   ps = conn.prepareStatement(sql);
   ps.setString(1,aid);
   rs = ps.executeQuery();
   rs.next();
   atitleTextField.setText(rs.getString(1));
   descTextField.setText(rs.getString(2));
  }
  catch(SQLException e) {
   JOptionPane.showMessageDialog(this,e);      } }
```

```
 private void updateButtonActionPerformed(java.awt.event.ActionEvent evt) {
  try {
   String sql = "update Announcement set AName=?,Description=? where
AnnouncementID=?";
   ps = conn.prepareStatement(sql);
   ps.setString(1,atitleTextField.getText());
   ps.setString(2, descTextField.getText());
   ps.setString(3,aid);
   ps.executeQuery();

   JOptionPane.showMessageDialog(this,"Updated!");
   this.setVisible(false);
  }
  catch(SQLException e) {
   JOptionPane.showMessageDialog(this,e);      } }

 public static void main(String args[]) {
  java.awt.EventQueue.invokeLater(new Runnable() {
   public void run() {
    new UpdateAnnouncementFrame().setVisible(true); }     }); }

 String aid;
 Connection conn;
 ResultSet rs;
 PreparedStatement ps;       }
```

********** **UpdateEventFrame.java** **********

```
package student_management_system;

import java.sql.*;
import java.text.SimpleDateFormat;
import javax.swing.JOptionPane;

public class UpdateEventFrame extends javax.swing.JFrame {


 public UpdateEventFrame() {
  initComponents();     }

 public UpdateEventFrame(String eid, Connection conn) {
  this.eid = eid;
  this.conn = conn;
  sdf = new SimpleDateFormat("dd-MMM-yyyy");
  initComponents();

  try {
   String sql = "select EventName,Description,EventDate,EventTime,Venue from Event
where EventID=?";
   ps = conn.prepareStatement(sql);
   ps.setString(1,eid);
   rs = ps.executeQuery();
   rs.next();
   eventNameTextField.setText(rs.getString(1));
   descTextField.setText(rs.getString(2));
   datePicker.setDate(rs.getDate(3));
   timeTextField.setText(rs.getString(4));
   venueTextField.setText(rs.getString(5));
  }
  catch(SQLException e) {
   JOptionPane.showMessageDialog(this,e);
  } }
```

```
 private void updateButtonActionPerformed(java.awt.event.ActionEvent evt) {
  try {
   String sql = "update Event set
EventName=?,Description=?,EventDate=?,EventTime=?,Venue=? where EventID=?";
   ps = conn.prepareStatement(sql);
   ps.setString(1, eventNameTextField.getText());
   ps.setString(2,descTextField.getText());
   ps.setString(3,sdf.format(datePicker.getDate()));
   ps.setString(4, timeTextField.getText());
   ps.setString(5, venueTextField.getText());
   ps.setString(6,eid);
   rs = ps.executeQuery();
   rs.next();

   JOptionPane.showMessageDialog(this, "Updated!");
   this.setVisible(false);
  } catch (SQLException e) {
   JOptionPane.showMessageDialog(this, e);    }     }

public static void main(String args[]) {
  java.awt.EventQueue.invokeLater(new Runnable() {
   public void run() {
    new UpdateEventFrame().setVisible(true);  } }); }

 String eid;
 Connection conn;
 ResultSet rs;
 PreparedStatement ps;
 SimpleDateFormat sdf;    }
```