Group Report

College Event Website

Course Number: COP 4710

Section: 17

Term: Summer 2017

Group #: 26

Group Member: Tuan Pham

Contents

Project Description 3
GUI 3
Feature Screenshots 3
ER-Model 8
Relational Data Model 9
Sample Data 10
SQL Examples and Results 12
Constraints 19
Installation Instructions 21
Conclusion/Observation 21

Project Description

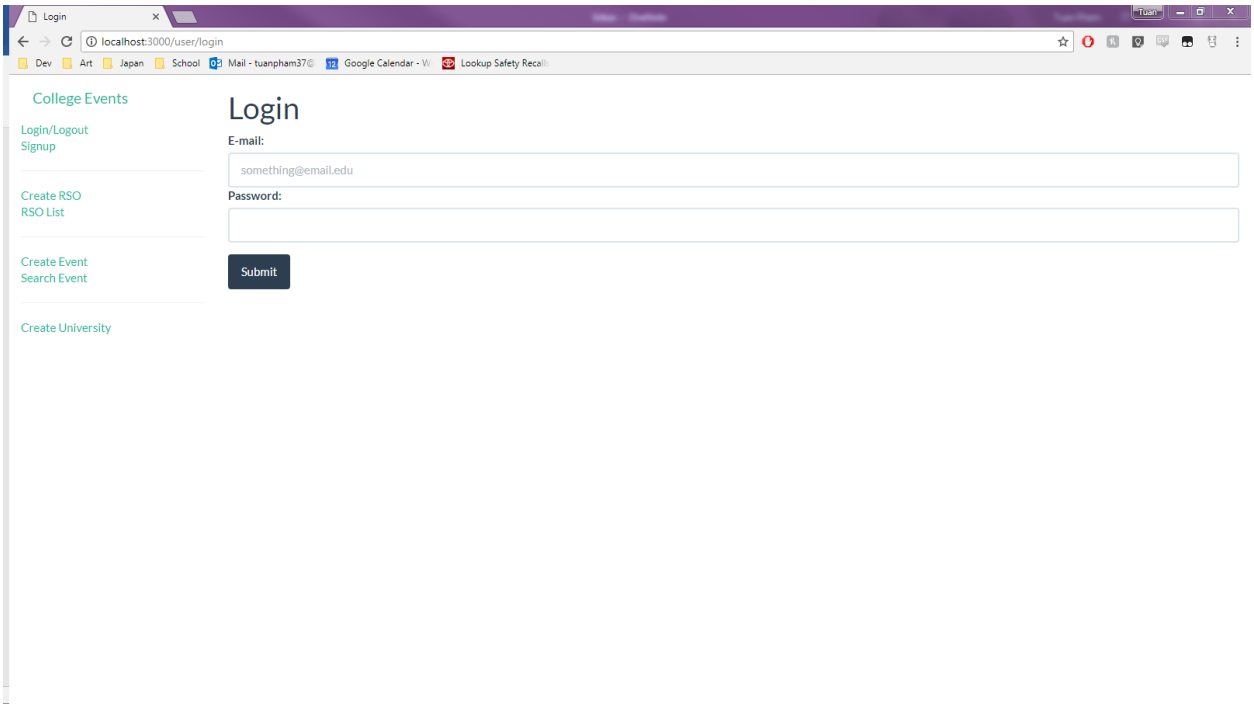
This project is a website for students to create and keep track of events surrounding universities. In addition, they will also be able to create and join RSO's who are the groups that organizes the events. The data powering the website is stored on a database and should be accessible to multiple, concurrent users from universities around the country. The project demonstrates a real-world example of database usage in a way that is relevant to us students.

GUI

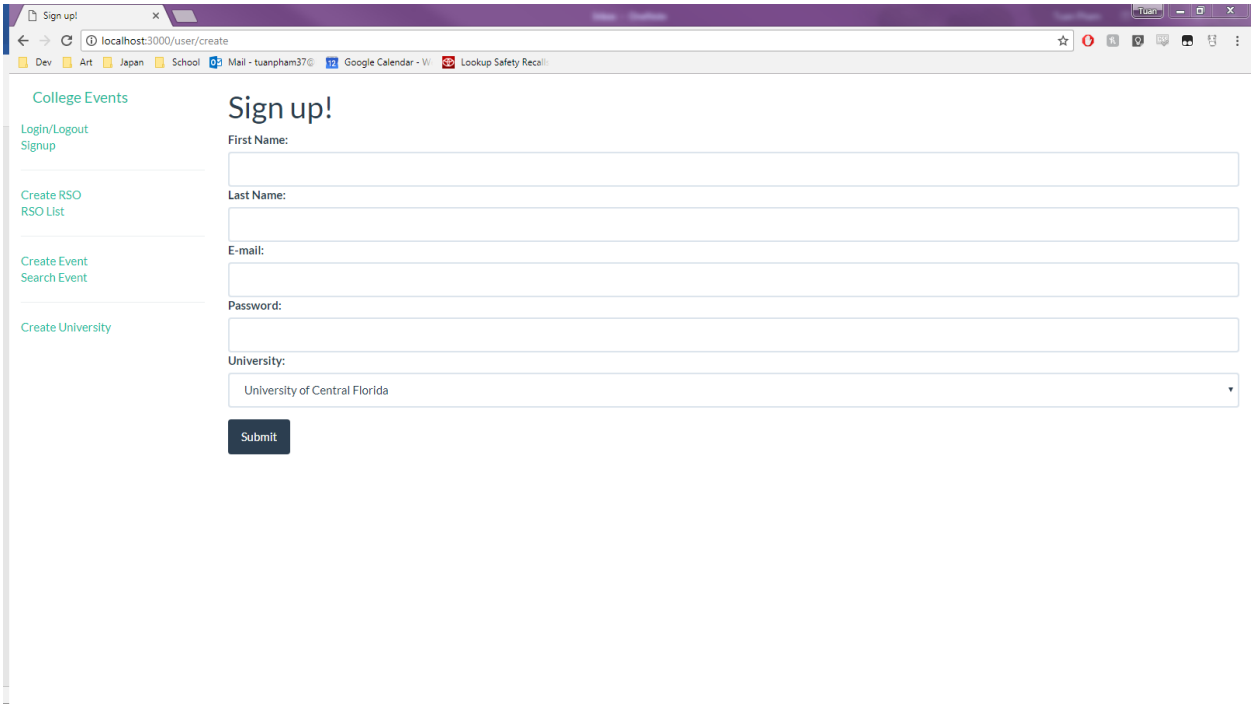
The front end was written using PUG templates to dynamically generate content and Bootstrap to easily create the forms. The back end was written using Node.js and Express because of previous experience with JavaScript. MySQL was the choice for DBMS.

Feature Screenshots

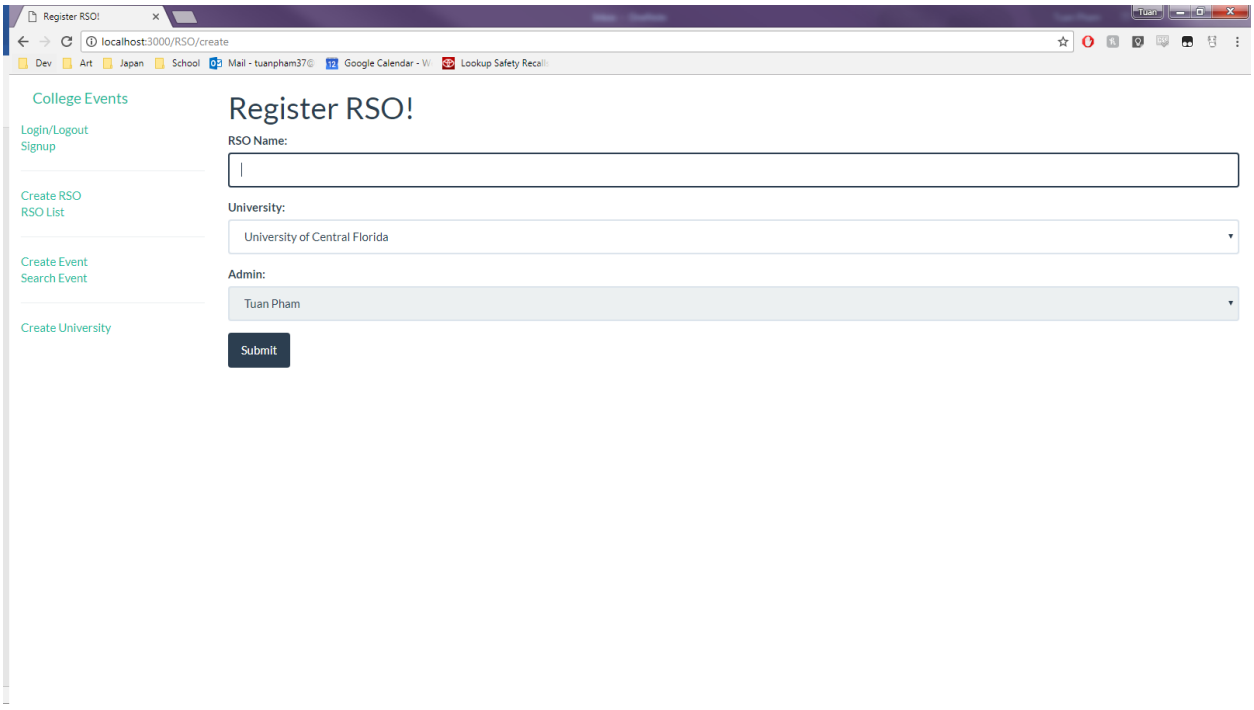
Login:



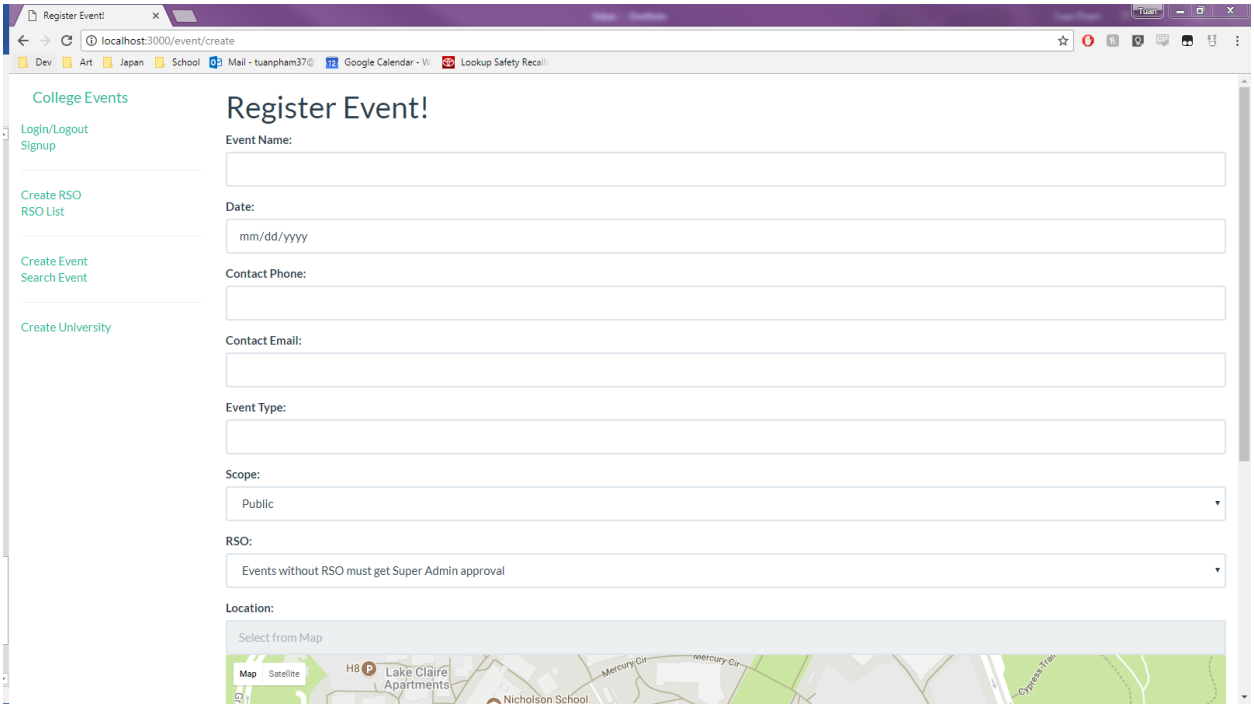
Sign Up:

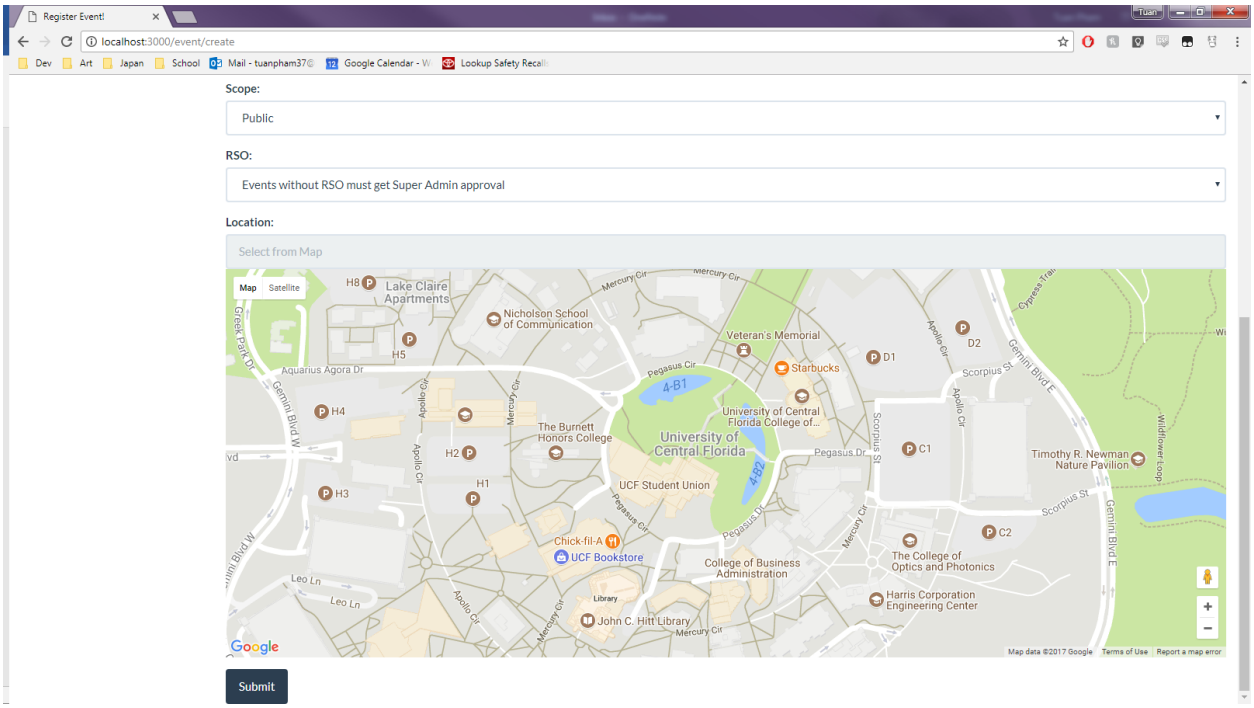


Register RSO:

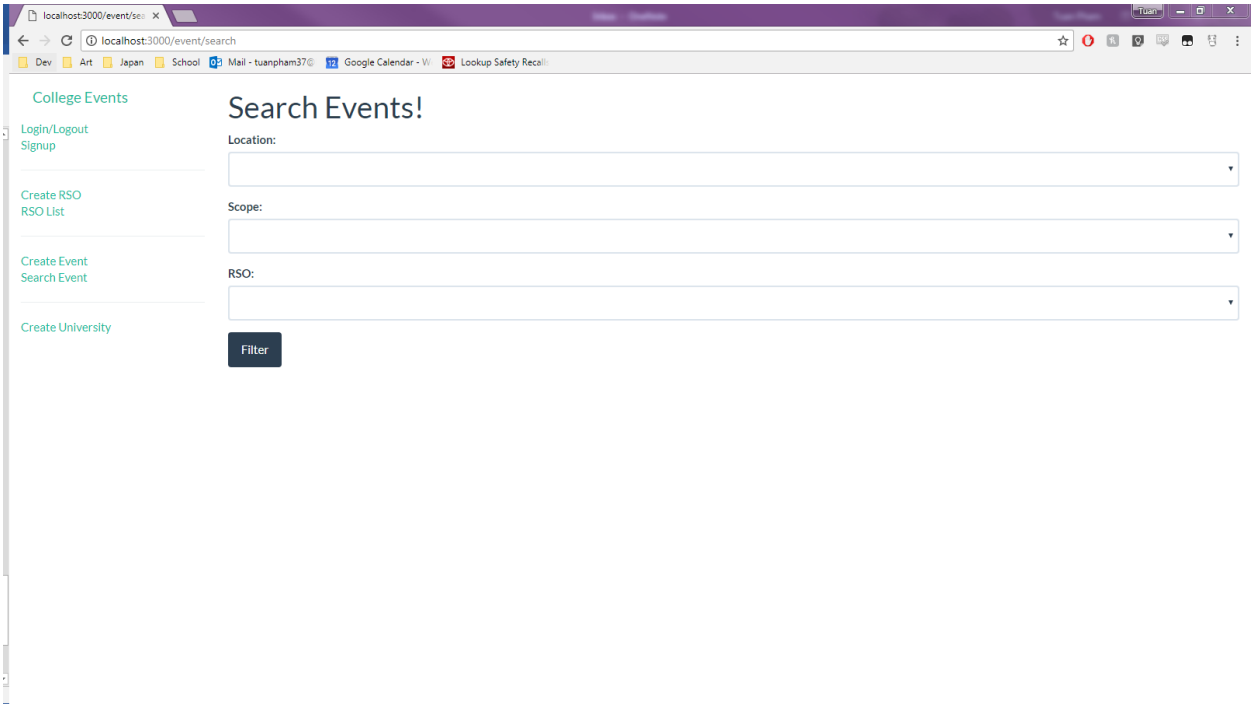


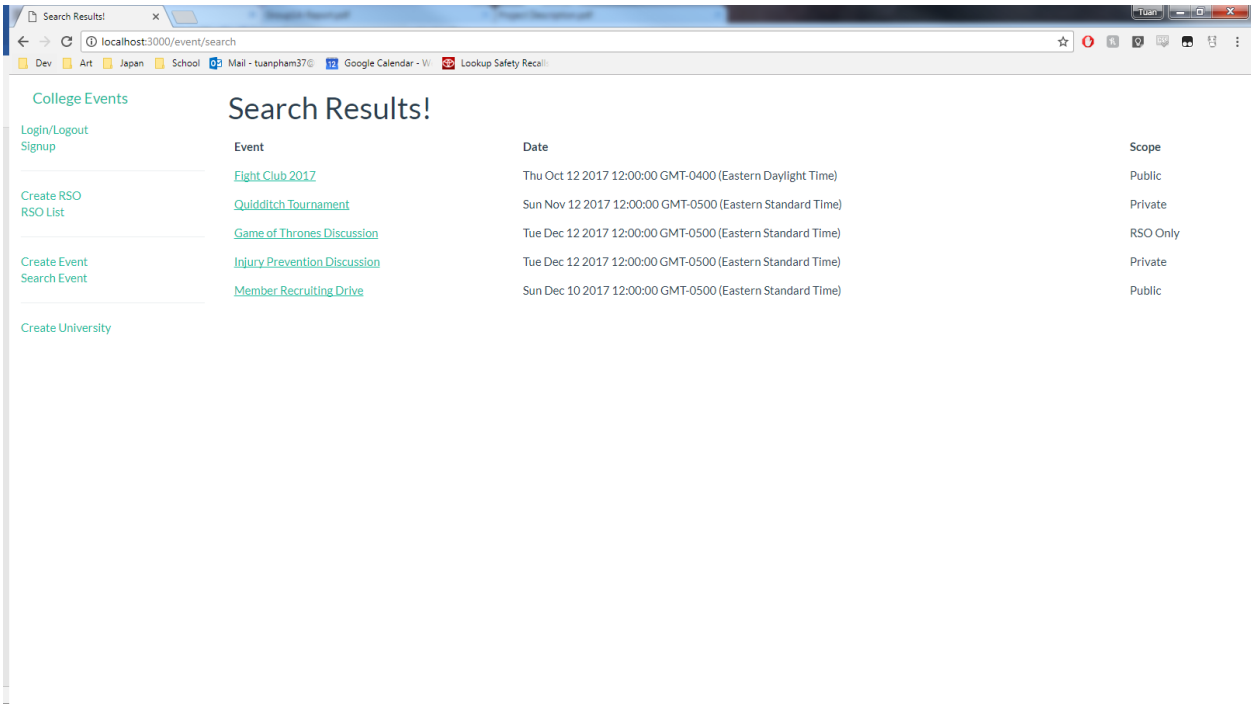
Register Event:



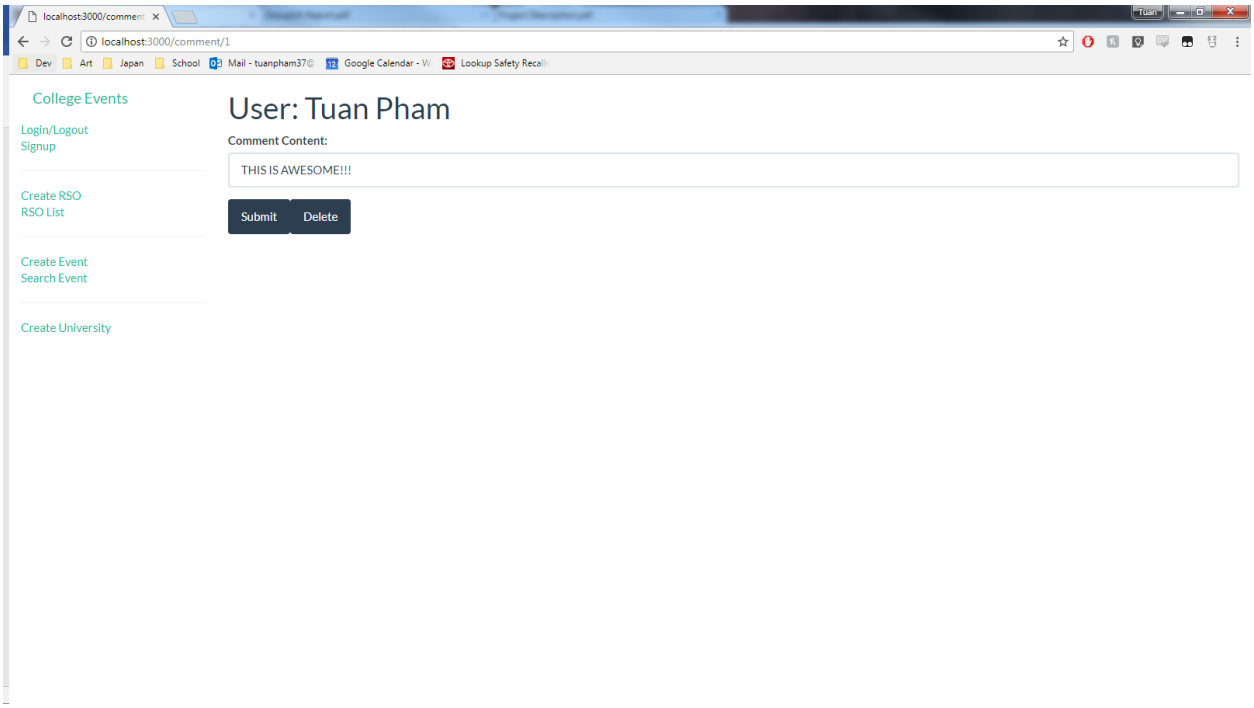


Event Search:

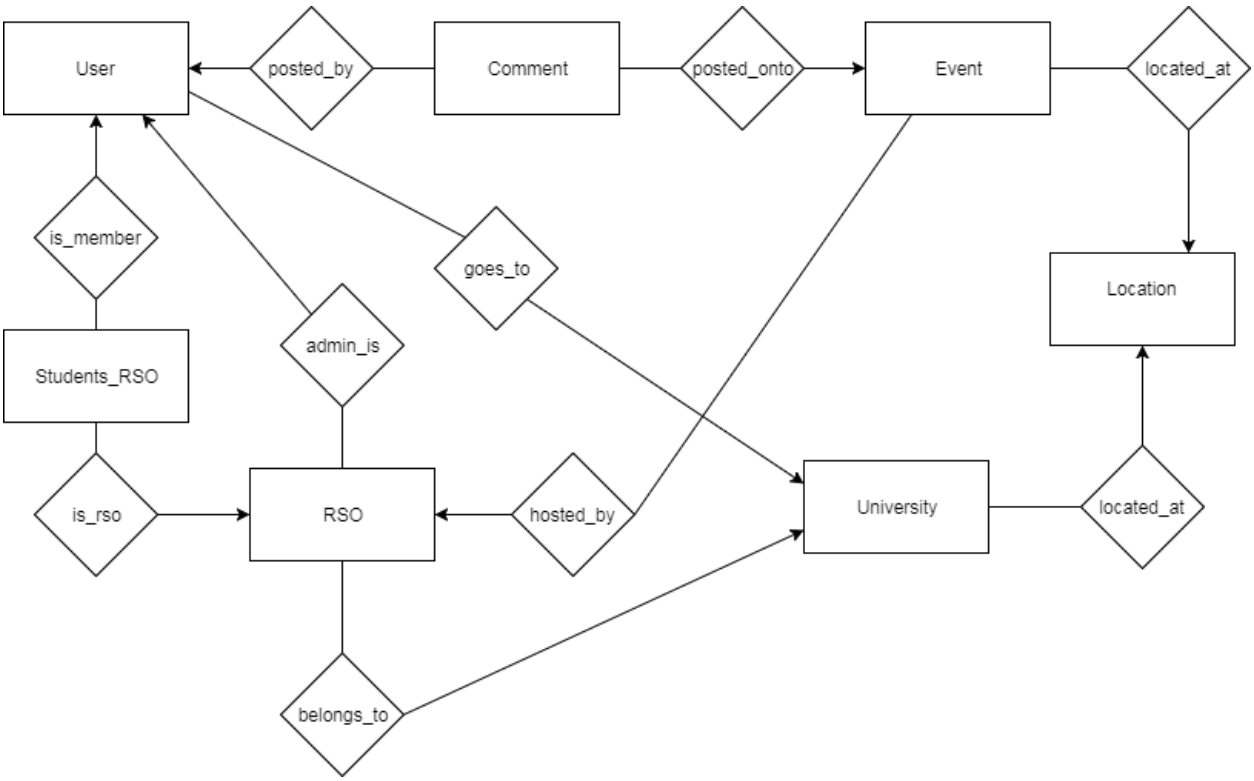




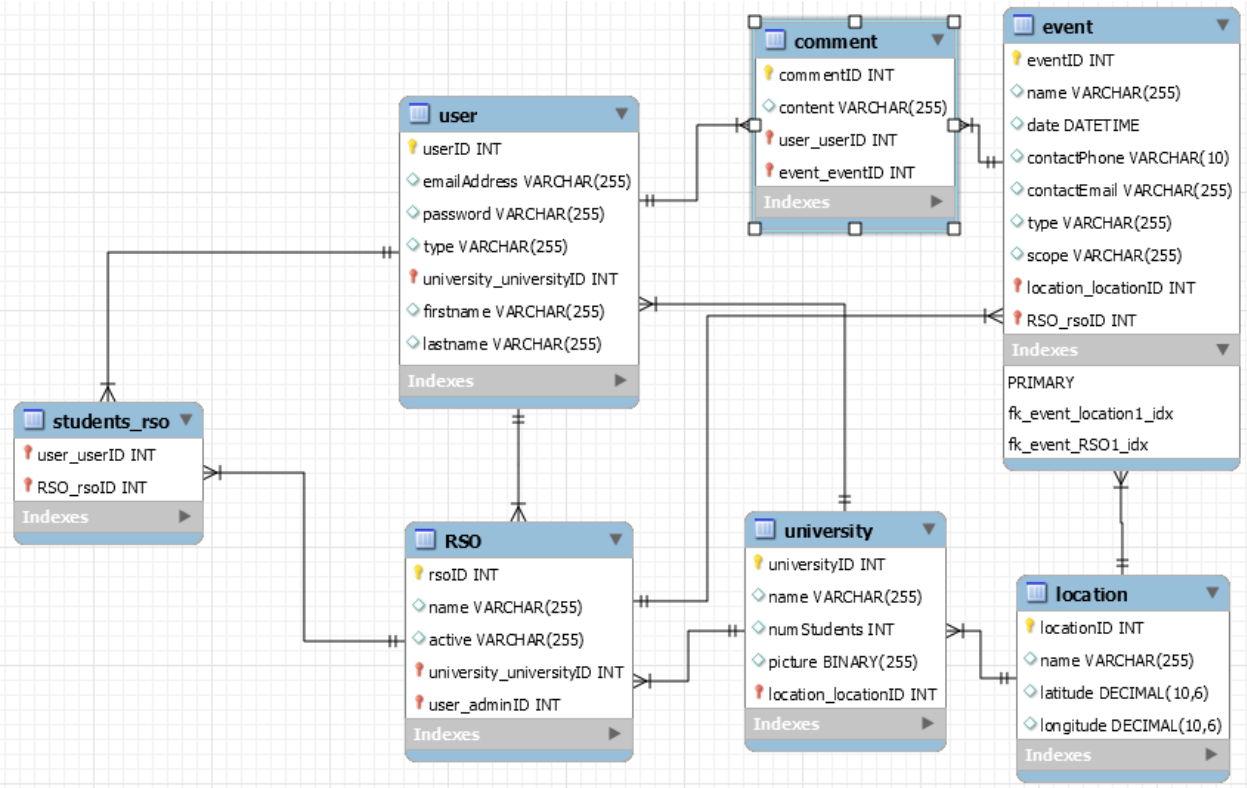
Event View, Comment Add/Edit/Delete:



ER-Model


User
posted_by
Comment
posted_onto
Event
located_at
is_member
goes_to
Location
admin_is
Students_RSO
is_rso
RSO
hosted_by
University
located_at
belongs_to

Relational Data Model



Sample Table Creation SQL (Rest of DB creation code in Project Folder):

```
DROP TABLE IF EXISTS `sitedb`.`university` ;

CREATE TABLE IF NOT EXISTS `sitedb`.`university` (
  `universityID` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(255) NULL,
  `numStudents` INT NULL,
  `picture` BINARY(255) NULL,
  `location_locationID` INT NOT NULL,
  PRIMARY KEY (`universityID`, `location_locationID`),
  INDEX `fk_university_location1_idx` (`location_locationID` ASC),
  CONSTRAINT `fk_university_location1`
    FOREIGN KEY (`location_locationID`)
    REFERENCES `sitedb`.`location` (`locationID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Sample Data

Location:

locationID	name	latitude	longitude
1	4000 Central Florida Blvd. Orlando. FL 32816	-81.200081	28.602427
2	Gainesville. FL 32611	-82.354932	29.644012

University:

universityID	name	numStudents	picture	location_locationID
1	University of Central Florida	64318	NULL	1
2	University of Florida	52286	NULL	2

User:

userID	emailAddress	password	type	university_universityID	firstname	lastname
1	tuan@ucf.edu	asdf	Student	1	Tuan	Pham
2	kyle@ucf.edu	password	Student	1	Kyle	Jones
3	john@ucf.edu	idksomething	Student	1	John	Deere
4	maxi@ucf.edu	secure	Student	1	Maxi	Faxi
5	lexi@ucf.edu	autogen	Student	1	Lexi	Steele
6	aden@ucf.edu	randy	Student	1	Aden	Nguyen
7	jake@ucf.edu	LIVE	Student	1	Jake	Statefarm
8	jack@ucf.edu	rawr	Student	1	Jack	Bauer
9	luke@ucf.edu	alphonse	Student	1	Luke	Cage
10	erin@ucf.edu	okay	Student	1	Erin	Cage
11	addeline@ucf...	lerandom	Student	1	Addeline	Zeus
12	Ziek@uf.edu	lelalela	Student	2	Ziek	Geek
13	doof@uf.edu	maloof	Student	2	Doof	Moof
14	garrus@uf.edu	areuserious	Student	2	Garrus	Alterian
15	vera@uf.edu	blaze	Student	2	Vera	Wang
16	lucille@uf.edu	nonon	Student	2	Lucille	Ball
17	soren@uf.edu	aveeavee	Student	2	Soren	Pham
18	alfie@uf.edu	alpharetta	Student	2	Alfie	Pham
19	rena@uf.edu	NOOOO	Student	2	Rena	Le
20	dylan@uf.edu	reasdfikl	Student	2	Dylan	Johns
21	harley@uf.edu	Xiied8X	Student	2	Harley	Johns
22	admin@ucf.edu	asdf	Admin	1	Regular	Admin
23	aiden@ucf.edu	asdf	Admin	1	Aiden	Jayden
24	kicker@uf.edu	asdf	Admin	2	Kicker	McLicker
25	super@ucf.edu	asdf	Super ...	1	Super	Admin
26	duper@ucf.edu	asdf	Super ...	2	Duper	Admin

RSO:

rsoID	name	active	university_universityID	user_adminID
1	Kickboxing Club	active	1	22
2	Witchcraft and Wizardry Club	inactive	1	23
3	Book Club	active	2	24

Event:

eventID	name	date	contactPhone	contactEmail	type	scope	location_locationID	RSO_rsoID
1	Fight Club 2017	2017-10-12 12:00:00	4078008000	admin@ucf.edu	Athletic	Public	1	1
2	Quidditch Tournament	2017-11-12 12:00:00	8004119871	aiden@ucf.edu	Athletic	Private	1	2
3	Game of Thrones Discussion	2017-12-12 12:00:00	8080808080	kicker@uf.edu	Social	RSO Only	2	3
4	Injury Prevention Discussion	2017-12-12 12:00:00	4078008000	admin@ucf.edu	Social	Private	1	1
5	Member Recruiting Drive	2017-12-10 12:00:00	8004119871	aiden@ucf.edu	Social	Public	1	2

Students_rso:

user_userID	RSO_rsoID
1	1
2	1
3	1
4	1
5	1
6	2
7	2
8	2
13	3
14	3
15	3
16	3
17	3

Comment:

commentID	content	user_userID	event_eventID
1	THIS IS AWESOME!!!	1	1
2	Can not wait.	2	1
3	This event seems a little violent	3	1
4	QUIDDITCH!!!	8	2

SQL Examples and Results

Insert RSO, Student, Event, Comment SQL:

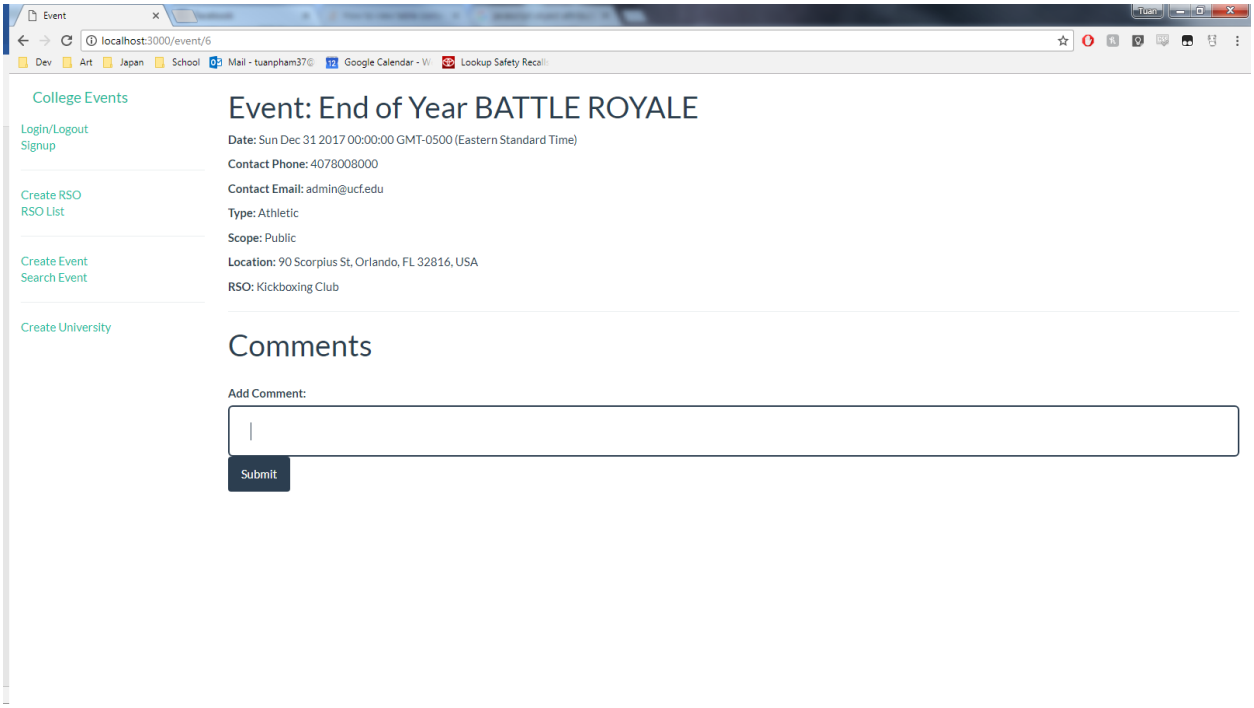
```
INSERT INTO rso
SET ?;
INSERT INTO user
SET ?;
INSERT INTO event
SET ?;
INSERT INTO comment
SET ?;
```

All of the insert SQL syntax followed the same template. Question marks were used in JavaScript code as placeholders for appropriate values. To use this syntax, JavaScript objects were created with attribute names matching the appropriate column name in MySQL. Most data validation was performed on the site itself to easier send appropriate errors back to users.

Example Event Insert JavaScript Code:

```
var newEvent = {
  name: req.body.name,
  date: req.body.date,
  contactPhone: req.body.contactPhone,
  contactEmail: req.body.contactEmail,
  type: req.body.type,
  scope: req.body.scope,
  location_locationID: results[0][0].insertId,
  rso_rsoID: req.body.rso
}
let query = mysql.format('INSERT INTO event SET ?;', newEvent);
db.query(query, function(error, results2, fields) {
    res.redirect('/event/' + results2.insertId);
});
```

Example Insert Event GUI+MySQL Result:

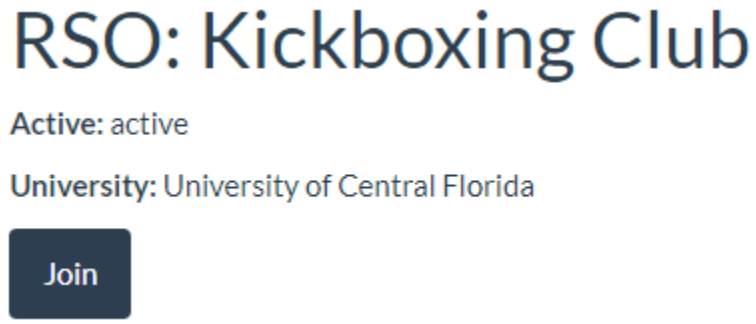


eventID	name	date	contactPhone	contactEmail	type	scope	location_locationID	RSO_rsoID
1	Fight Club 2017	2017-10-12 12:00:00	4078008000	admin@ucf.edu	Athletic	Public	1	1
2	Quidditch Tournament	2017-11-12 12:00:00	8004119871	aiden@ucf.edu	Athletic	Private	1	2
3	Game of Thrones Discussion	2017-12-12 12:00:00	8080808080	kicker@uf.edu	Social	RSO Only	2	3
4	Injury Prevention Discussion	2017-12-12 12:00:00	4078008000	admin@ucf.edu	Social	Private	1	1
5	Member Recruiting Drive	2017-12-10 12:00:00	8004119871	aiden@ucf.edu	Social	Public	1	2
6	End of Year BATTLE ROYALE	2017-12-31 00:00:00	4078008000	admin@ucf.edu	Athletic	Public	3	1

Insert/Delete Student in RSO SQL:

```
--Insert
INSERT INTO students_rso
SET ?;
--Delete
DELETE FROM students_rso
WHERE user_userID = ?
AND RSO_rsoID = ?;
```

Example Insert Student into RSO GUI+MySQL Result:



user_userID	RSO_rsoID
1	1
2	1
3	1
4	1
5	1
6	2
7	2
8	2
13	3
14	3
15	3
16	3
17	3

---

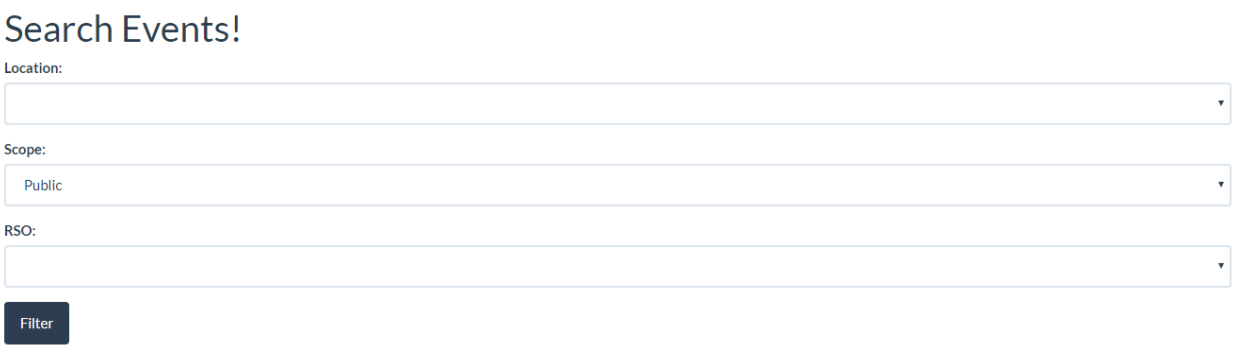
user_userID	RSO_rsoID
1	1
2	1
3	1
4	1
5	1
22	1
6	2
7	2
8	2
13	3
14	3
15	3
16	3
17	3

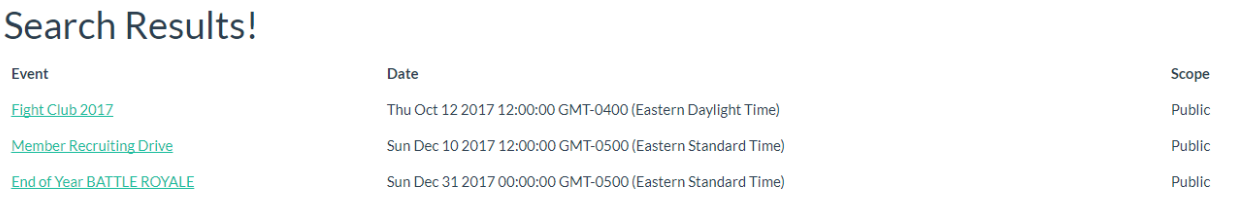
Search Events SQL+JavaScript:

```
var string = 'SELECT * FROM event';
var filters = [];
if (req.body.location !== '' || req.body.scope !== '' || req.body.rso !== '') {
  string += ' WHERE ';
  if (req.body.location !== '') {
    string += 'event.location_locationID = ?'
    filters.push(req.body.location);
  }
  if (req.body.scope !== '') {
    if (string.endsWith('?')) string += ' AND ';
    string += 'event.scope = ?'
    filters.push(req.body.scope);
  }
  if (req.body.rso !== '') {
    if (string.endsWith('?')) string += ' AND ';
    string += 'event.RSO_rsoID = ?'
    filters.push(req.body.rso);
  }
}
string += ';';
var query = mysql.format(string, filters);
db.query(query, function(error, results, fields) {
  res.render('event_list', {title: 'Search Results!', event_list: results, filters: filters});
});
```

Event searching was implemented by providing the user select boxes for Location, Scope and RSO to filter their results. The entire JavaScript code was included to convey how searches with all, some, or no filters were performed.

Example Search Events GUI+MySQL Results:





eventID	name	date	contactPhone	contactEmail	type	scope	location_locationID	RSO_rsoID
1	Fight Club 2017	2017-10-12 12:00:00	4078008000	admin@ucf.edu	Athletic	Public	1	1
5	Member Recruiting Drive	2017-12-10 12:00:00	8004119871	aiden@ucf.edu	Social	Public	1	2
6	End of Year BATTLE ROYALE	2017-12-31 00:00:00	4078008000	admin@ucf.edu	Athletic	Public	3	1

Event Details and Comments SQL:

```
--Event Details
SELECT *,e.name as eventName, r.name as rsoName, l.name as locName
FROM event e, rso r, location l
WHERE eventID = ?
AND e.rso_rsoID = r.rsoID
AND e.location_locationID = l.locationID;
--Comments
SELECT *
FROM comment,user
WHERE event_eventID = ?
AND userID = user_userID;
```

Example Event Details GUI+MySQL Results:

Event: Fight Club 2017

Date: Thu Oct 12 2017 12:00:00 GMT-0400 (Eastern Daylight Time)

Contact Phone: 4078008000

Contact Email: admin@ucf.edu

Type: Athletic

Scope: Public

Location: 4000 Central Florida Blvd, Orlando, FL 32816

RSO: Kickboxing Club

Comments

Tuan Pham	THIS IS AWESOME!!!
Kyle Jones	Can not wait.
John Deere	This event seems a little violent

eventID	name	date	contactPhone	contactEmail	type	scope	location_locationID	RSO_rsoID	rsoID	name	active	university_universityID	user_adminID	locationID
1	Fight Club 2017	2017-10-12 12:00:00	4078008000	admin@ucf.edu	Athletic	Public	1	1	1	Kickboxing Club	active	1	22	1

name	latitude	longitude	eventName	rsoName	locName
4000 Central Florida Blvd. Orlando. FL 32816	-81.200081	28.602427	Fight Club 2017	Kickboxing Club	4000 Central Florida Blvd. Orlando. FL 32816

commentID	content	user_userID	event_eventID	userID	emailAddress	password	type	university_universityID	firstname	lastname
1	THIS IS AWESOME!!!	1	1	1	tuan@ucf.edu	asdf	Student	1	Tuan	Pham
2	Can not wait.	2	1	2	kyle@ucf.edu	password	Student	1	Kyle	Jones
3	This event seems a little violent	3	1	3	john@ucf.edu	idksomething	Student	1	John	Deere

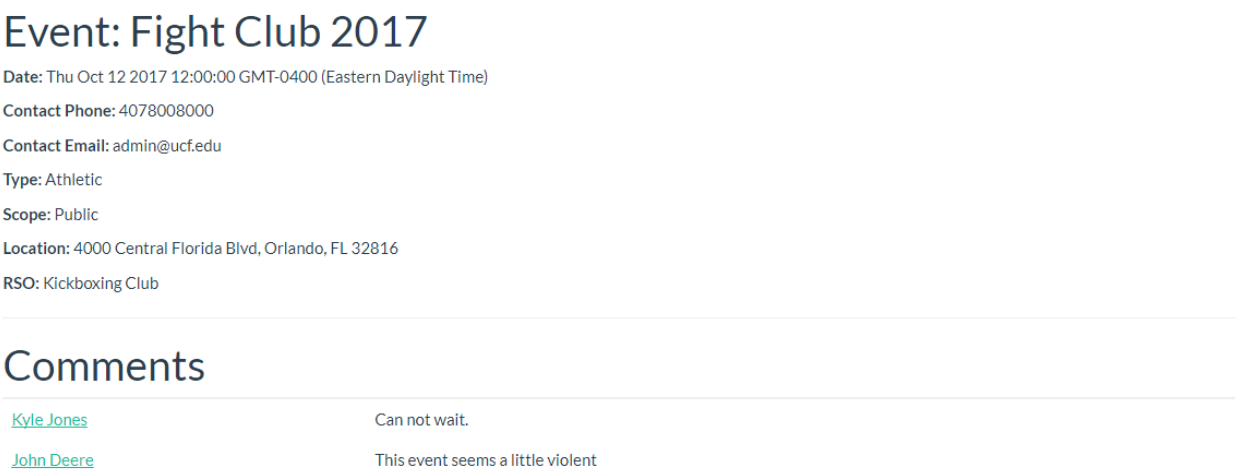
Comments are clickable which takes you to a comment details page. If you made the comment, then you have the option to update or delete your comment.

Update, Delete Comment SQL:

```
--Delete
DELETE FROM comment
WHERE commentID = ?;
--Update
UPDATE comment
SET content = ?
WHERE commentID = ?;
```

Example Comment Details GUI:

Example Delete Comments GUI Result:

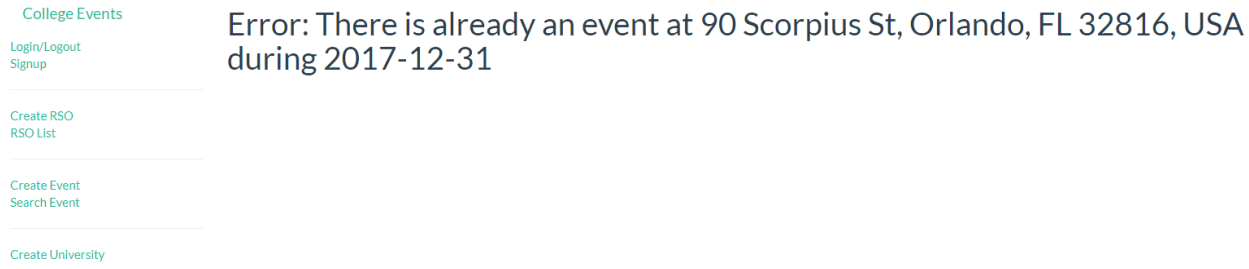


Constraints

Events with Overlapping Location and Dates:

```
SELECT COUNT(*) as count
FROM event e, location l
WHERE e.location_locationID = l.locationID
AND e.date = ?
AND l.name = ?;
```

If count is not zero, then there is a conflicting event.

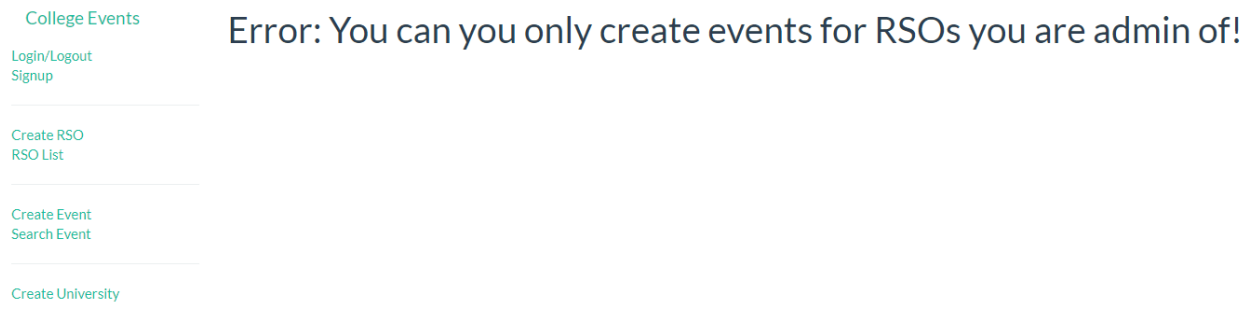


Incorrect Admin for RSO:

```
SELECT user_adminID
FROM rso
WHERE rsoID = ?;
```

The admin ID is obtained from the RSO entered in the previous form.

It is then checked against the ID of the logged in user.



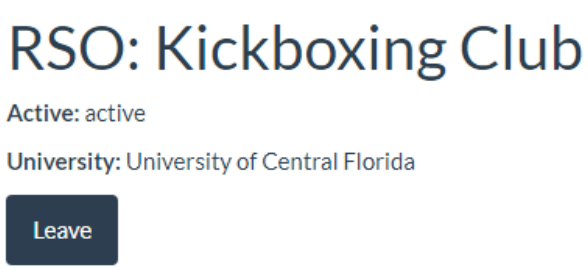
Active/Inactive RSOs:

```
SELECT COUNT(*) as count
FROM students_rso
WHERE RSO_rsoID = ?;
```

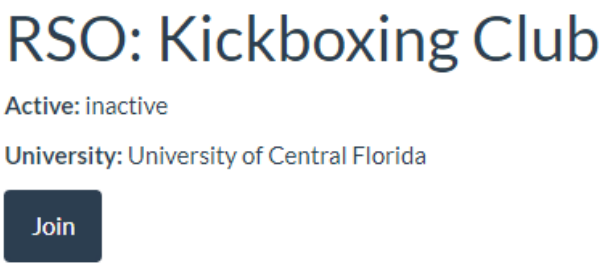
After insert/delete from students_rso, the count for an RSO's members is obtained.

If the count changes from five to four, the active column is changed from 'active' to 'inactive' and vice-versa.

count
5



count
4



Installation Instructions

MySQL Setup:

1) Install MySQL
2) Run MySQL Service
3) Navigate to path: *FOLDER CONTAINING PROJECT* \college-event-app\db\model
4) Run createSiteDB.sql (will drop tables inside siteDB schema)
5) Run seedSiteDB.sql (Some sample data necessary for forms)

Web Server Setup:

1) Install Node.js
2) Open Command Prompt
3) Navigate to path: *FOLDER CONTAINING PROJECT* \college-event-app
4) Run command 'npm install'
5) Run command 'npm start'
6) Open web browser
7) Navigate to: http://localhost:3000
8) (Optional) Login to one of the Sample Logins:

Account Type:	Email	Password
Student	tuan@ucf.edu	asdf
Admin	admin@ucf.edu	asdf
Super Admin	super@ucf.edu	asdf

Conclusion/Observation

A few features that were left out due to time constraints were security features for logging in, feeds from university sites and some social media integration. One possibility was to use social media API's for logging in and comments. Although that may not be a good idea for students who don't have social media or wish to separate it from their school life. One of the major hurdles was the initial website and database design. A longer and more detailed planning phase would have streamlined the project. However, there were many things I learned along the way that I would not have been able to foresee before attempting this project. The biggest issue was learning web dev from a very minimal web dev background. Additional MySQL and even general SQL knowledge would have been very helpful. Most of the validation was performed on the site itself but more SQL knowledge could have added it to the DBMS instead. In the end, I was able to implement the features and constraints requested due to proper scheduling and utilizing technologies I was more comfortable with.