Group Report, College Event Website

Databases COP 4710 Spring 2017

Group 14: Aditya Shankar, Daniel Belalcazar, Joseph Landry, Nicholas Ho Lung

Table of Contents

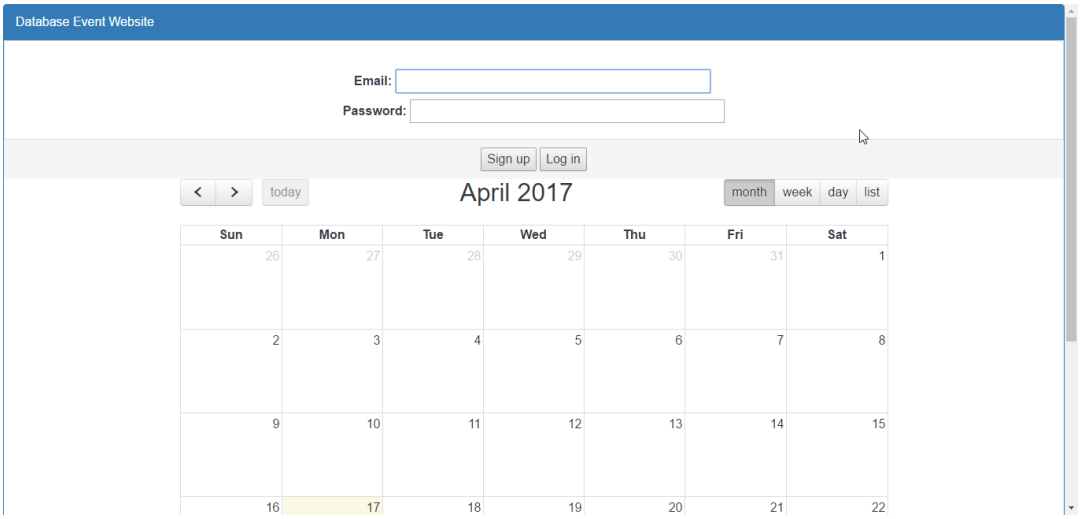
Project Description:

The goal of this project is to create a website application that can keep track of the people that go to certain events at a given university. All of these people, events, and universities should be kept track of using a set of tables within a database that can store any amount of each item along with relevant user information. Students should be able to join organizations. Organizations should be able hold events. Users should also be able leave comments on events that they are interested in. Our app should keep track of who is involved with what and maintain the integrity of the data received from users.

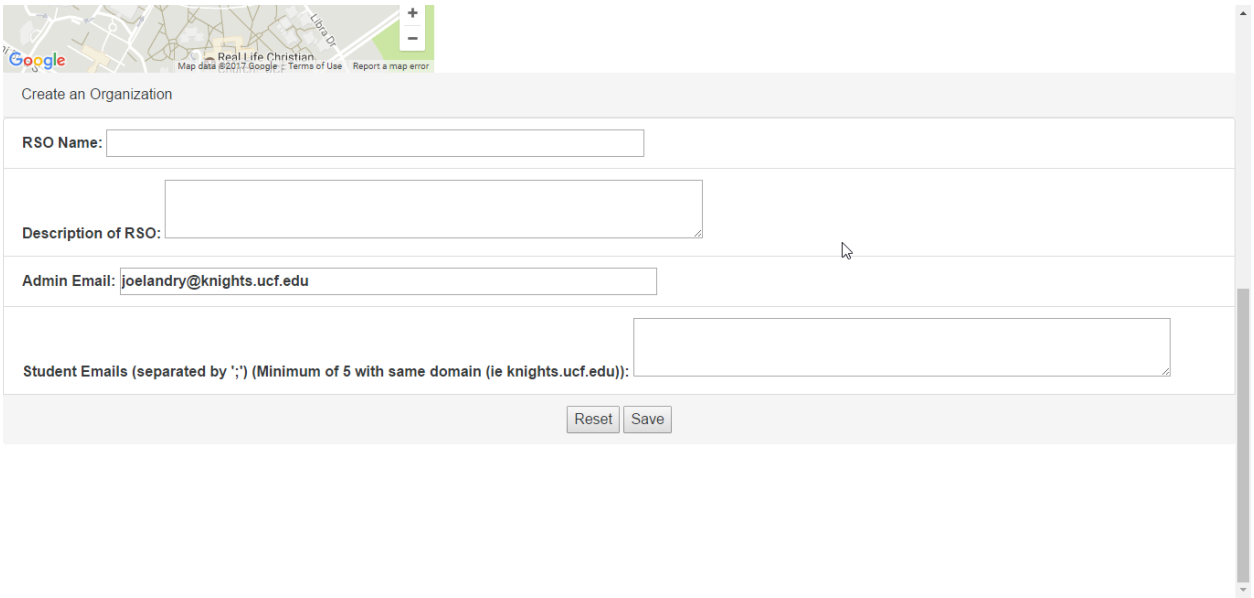
GUI:

The front-end was made using AngularJS and Bootstrap while the back-end/API was made with NodeJS and Express. The database component of the project was created through the use of MySQL. The next few images showcase what the application looks like when creating RSO's and Events along with viewing said events.

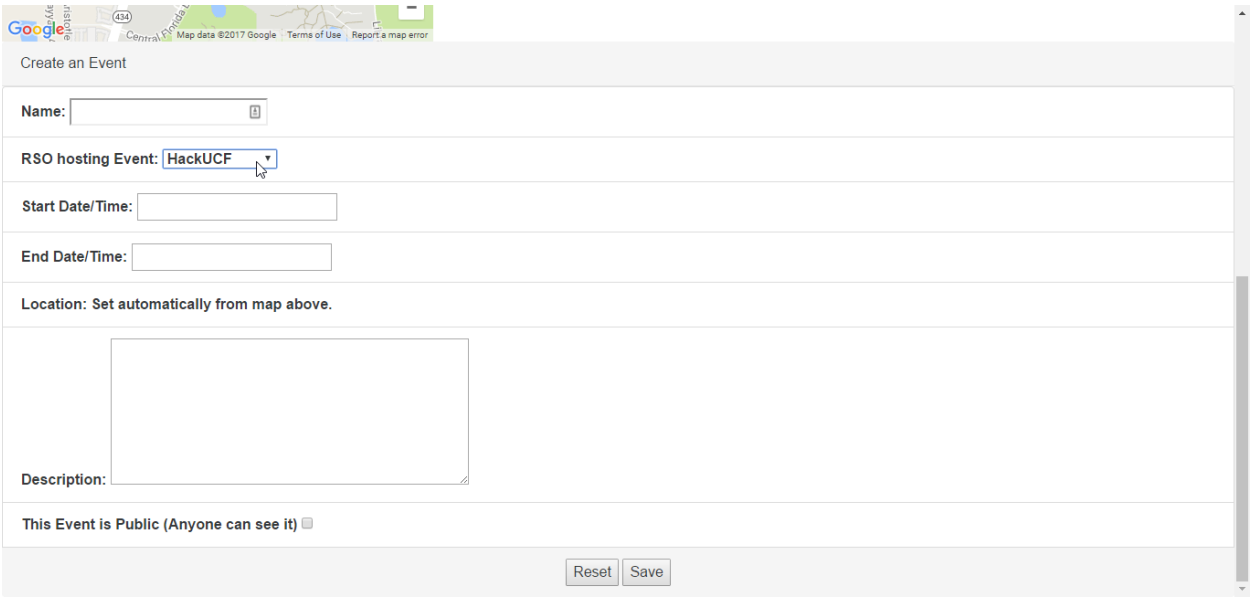
Login Page:



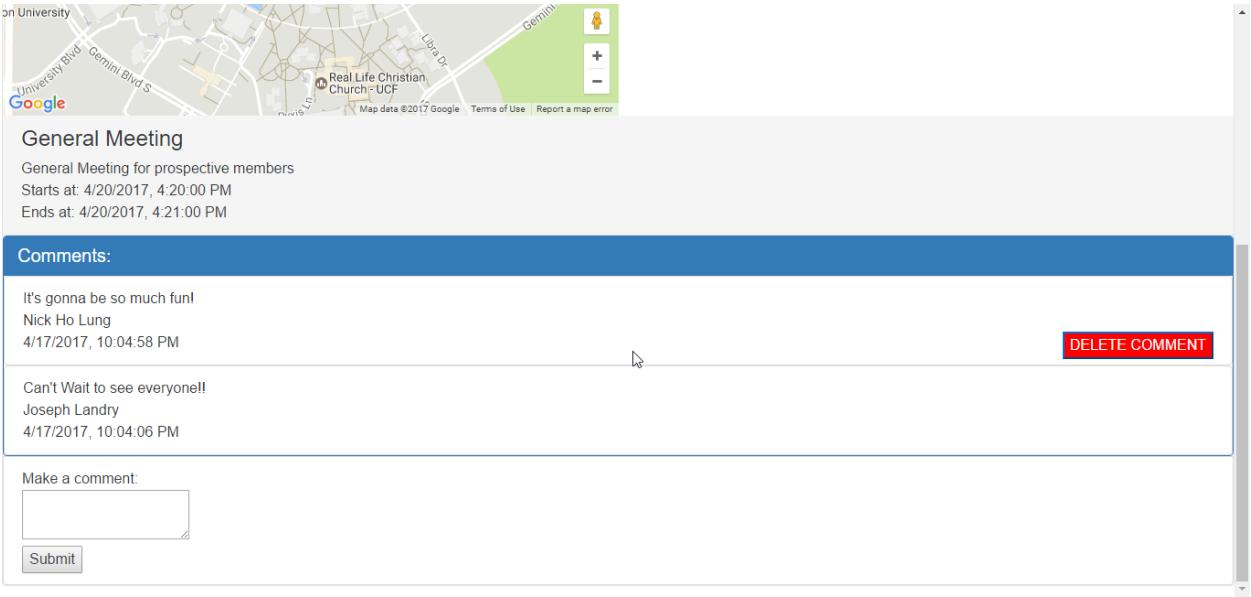
Creating an RSO:



Creating an Event:

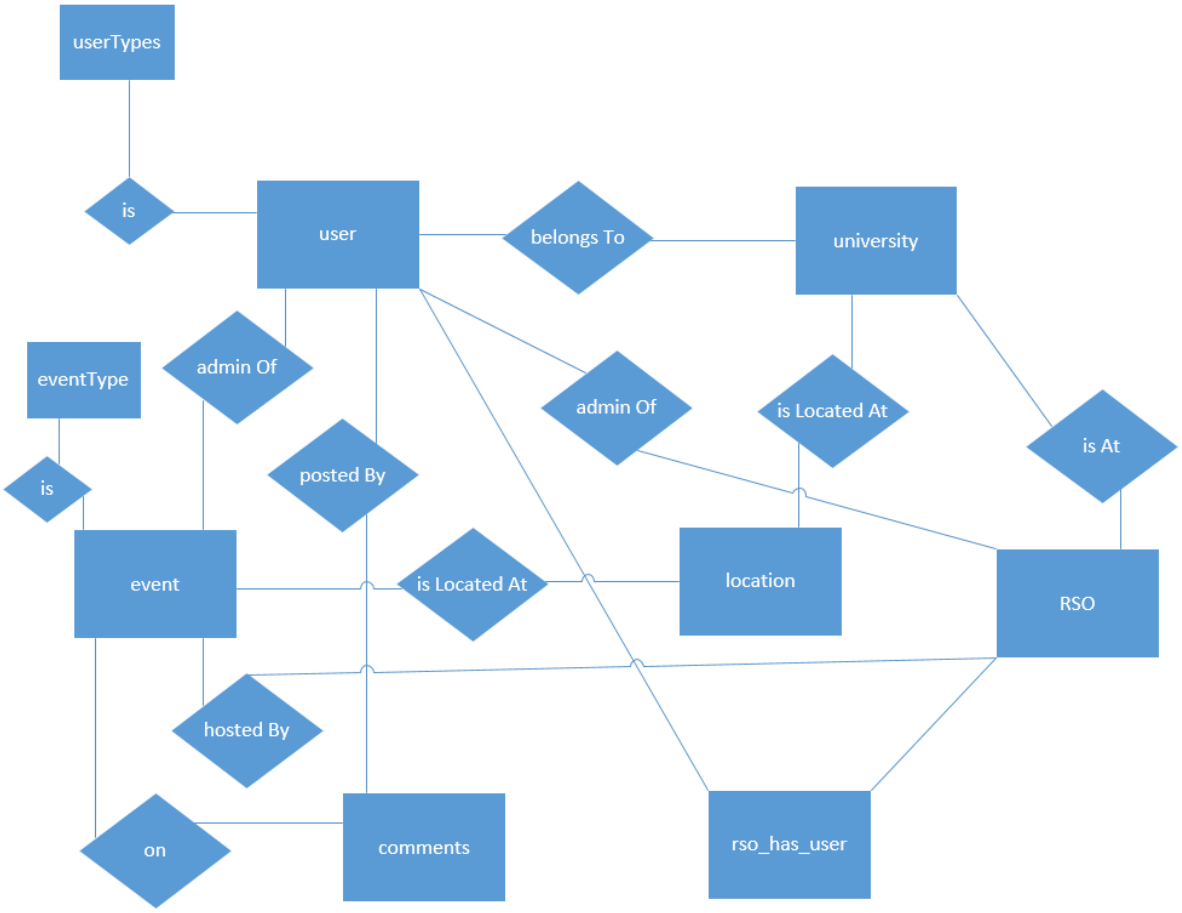


Viewing an Event:

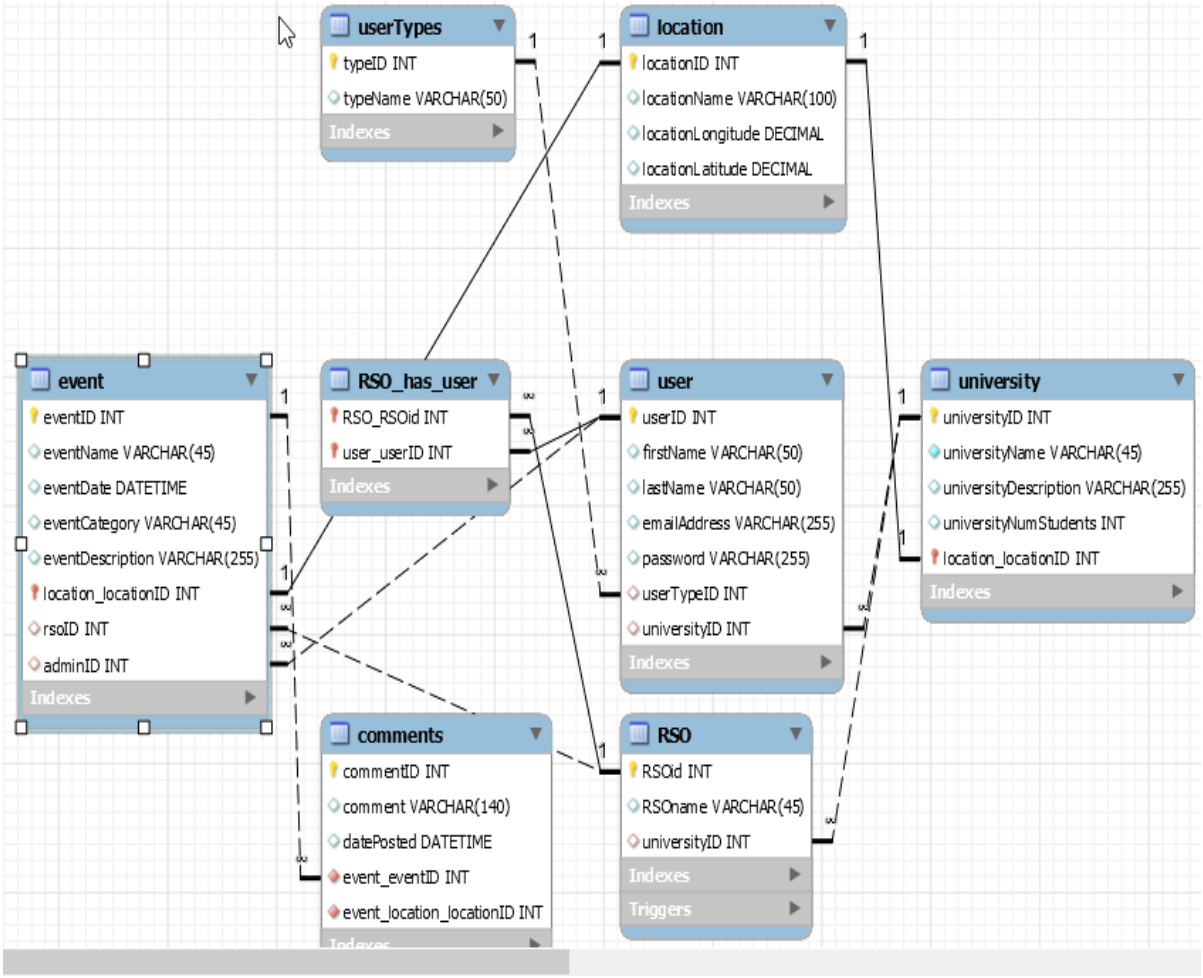


Notice the ability to delete comments, but only for the current user's comments.

ER Diagram:

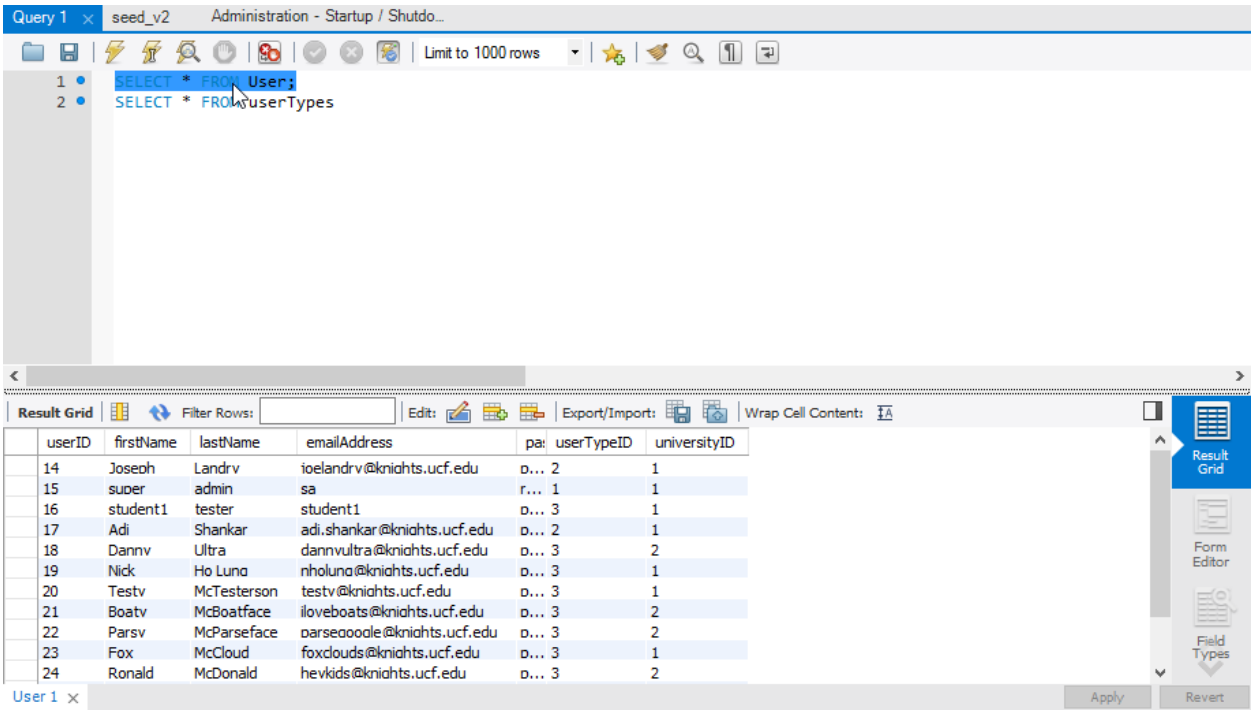

userTypes
is
user
belongs To
university
eventType
admin Of
admin Of
is Located At
is At
is
posted By
event
is Located At
location
RSO
hosted By
on
comments
rso_has_user

Relationship Model with attributes:



Database filled with sample data:

Sample User Table:



userID	firstName	lastName	emailAddress	pa	userTypeID	universityID
14	Joseph	Landry	joelandry@knights.ucf.edu	p...	2	1
15	super	admin	sa	r...	1	1
16	student1	tester	student1	p...	3	1
17	Adi	Shankar	adi.shankar@knights.ucf.edu	p...	2	1
18	Danny	Ultra	dannyultra@knights.ucf.edu	p...	3	2
19	Nick	Ho Lung	nholung@knights.ucf.edu	p...	3	1
20	Testy	McTesterson	testy@knights.ucf.edu	p...	3	1
21	Boaty	McBoatface	iloveboats@knights.ucf.edu	p...	3	2
22	Parsy	McParseface	parseagoogle@knights.ucf.edu	p...	3	2
23	Fox	McCloud	foxclouds@knights.ucf.edu	p...	3	1
24	Ronald	McDonald	heykids@knights.ucf.edu	p...	3	2

Sample User Types Table:

typeID	typeName
1	superAdmin
2	admin
3	student
NULL	NULL

Sample RSO table:

RSOid	RSOname	universityID
1	Habitat for Humanity	1
2	HackUCF	1
3	Tech Knights	1
4	Student Government Association	2
5	How to be Cena	2
6	Men's Football	1
7	Women's Volleyball	2
NULL	NULL	NULL

Sample University table:

Query 1 | seed_v2 | Administration - Startup / Shutdo...

Limit to 1000 rows

```
SELECT * FROM User;
SELECT * FROM userTypes;
SELECT * FROM University;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

universityID	universityName	universityDescription	universityNumStudents	location_locationID
1	University of Central Florida	Home of the UCF Knights!	64000	1
2	Cena U	Should i say more	1000	2
NULL	NULL	NULL	NULL	NULL

Result Grid | Form Editor | Field Types

University 2 | Apply | Revert

Sample Event Table:

eventID	eventName	eventDate	eventCategory	eventDescription	location_locationID	rsoID	adminID
1	Test Event1	2017-03-04 00:00:00	Test@contact.edu	testing event creation with DB	3	1	1
2	event2	2017-03-04 00:00:00	pop@google.com	More db testing	4	1	1
3	Event at HEC	2017-04-04 00:00:00	testing@hec.com	Hec event for testing	5	1	1
4	Another event	2017-03-04 00:00:00	contact@ucf.edu	this seems to be working	6	1	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Installation Information:

Prerequisites for installation:

1. Node.js (latest recommended version) https://nodejs.org/en/
2. MySQL Workbench (or other MySQL database management system): https://dev.mysql.com/downloads/installer/

How to Install

Setup DB:

1. In MySQL Workbench or other tool, run the sql script “database/createdb_v2.sql” (this is in the database folder).
2. This will setup the database. There is an option seed_v2.sql script to seed the database with some random information, but this is not necessary.

Setup and Run web server

1. Open up a command prompt or terminal to the directory our website was downloaded to.
2. cd into the “eventapp” folder (`cd eventapp`)
3. Run the command: `npm install`
4. After this completes, run the command: `npm start`
5. This begins the web server on the local machine.
6. Navigate a web browser (recommended: Google Chrome) to http://localhost:3000/ to access the website

SQL Queries in Use for the site:

In all of the Queries below, a question mark is used where the values would be (safely) filled in in the application

```
#To get all the public events
SELECT * FROM event WHERE eventTypeID = (
   SELECT eventTypeID FROM eventType
   WHERE eventTypeName LIKE 'public'
);

#To get the information on comments (date, comment, and poster name)
SELECT c.comment, c.datePosted, concat(u.firstName,' ',u.lastName) as name,
u.userID, c.commentID
   FROM comments c
   INNER JOIN user u ON c.userID = u.userID
   WHERE eventID = ?
   ORDER BY datePosted DESC;

#To Search for events. Uses either event name, or description
```

```
SELECT eventID, eventDescription, eventName
   FROM event
   INNER JOIN user ON user.userID = event.adminID
   WHERE (eventName LIKE ?) OR ( eventDescription LIKE ?) AND (eventTypeID = 1 OR
user.universityID = ?);

#This will get a location based on a locationID
SELECT * FROM location WHERE locationID = ?;

#Our login query.
`SELECT * FROM user WHERE emailAddress = ? AND password = ?;`

#Get Rsos that a user are a part of.
SELECT r.RSOname,ru.RSOid FROM rso_has_user ru INNER JOIN rso r ON r.RSOid =
ru.RSOid WHERE userID = ?;

#This is used to create a new Event. The ? gets populated with all of the fields
#and their SQL Query safe values.
INSERT INTO event SET ?;

#To insert a new comment.
INSERT INTO comments SET ?;

#This gets the longitude/latitude of a university (for showing on map)
SELECT l.locationLongitude, l.locationLatitude FROM university u
   INNER JOIN location l ON u.locationID = l.locationID
   WHERE universityID = ?;`

#This is to add a user to an RSO. the ? gets replaced with the userId of the user
#and the RSOid that the user is joining.
INSERT INTO rso_has_user SET ?;
```

Relational Data Model:

```
-- MySQL Workbench Forward Engineering
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';
-- -----------------------------------------------------
-- Schema eventWebsiteDatabase
-- -----------------------------------------------------
-- -----------------------------------------------------
-- Schema eventWebsiteDatabase
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `eventWebsiteDatabase` DEFAULT CHARACTER SET utf8 ;
USE `eventWebsiteDatabase` ;
-- -----------------------------------------------------
-- Table `eventWebsiteDatabase`.`location`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `eventWebsiteDatabase`.`location` ;
CREATE TABLE IF NOT EXISTS `eventWebsiteDatabase`.`location` (
 `locationID` INT NOT NULL AUTO_INCREMENT,
```

```sql
 `locationName` VARCHAR(100) NULL,
 `locationLongitude` DECIMAL(12,8) NULL,
 `locationLatitude` DECIMAL(12,8) NULL,
 PRIMARY KEY (`locationID`),
 UNIQUE INDEX `locationID_UNIQUE` (`locationID` ASC))
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `eventWebsiteDatabase`.`university`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `eventWebsiteDatabase`.`university` ;
CREATE TABLE IF NOT EXISTS `eventWebsiteDatabase`.`university` (
 `universityID` INT NOT NULL AUTO_INCREMENT,
 `universityName` VARCHAR(45) NOT NULL,
 `universityDescription` VARCHAR(255) NULL,
 `universityNumStudents` INT NULL,
 `locationID` INT NULL,
 `universityPicture` VARCHAR(500) NULL,
 PRIMARY KEY (`universityID`),
 INDEX `fk_university_location_idx` (`locationID` ASC),
 CONSTRAINT `fk_university_location`
   FOREIGN KEY (`locationID`)
   REFERENCES `eventWebsiteDatabase`.`location` (`locationID`)
   ON DELETE CASCADE
   ON UPDATE NO ACTION)
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `eventWebsiteDatabase`.`userTypes`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `eventWebsiteDatabase`.`userTypes` ;
CREATE TABLE IF NOT EXISTS `eventWebsiteDatabase`.`userTypes` (
 `typeID` INT NOT NULL AUTO_INCREMENT,
 `typeName` VARCHAR(50) NULL,
 PRIMARY KEY (`typeID`))
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `eventWebsiteDatabase`.`user`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `eventWebsiteDatabase`.`user` ;
CREATE TABLE IF NOT EXISTS `eventWebsiteDatabase`.`user` (
 `userID` INT NOT NULL AUTO_INCREMENT,
 `firstName` VARCHAR(50) NULL,
 `lastName` VARCHAR(50) NULL,
 `emailAddress` VARCHAR(255) NULL,
 `password` VARCHAR(255) NULL,
 `userTypeID` INT NULL,
 `universityID` INT NULL,
 PRIMARY KEY (`userID`),
 INDEX `fk_usertype_idx` (`userTypeID` ASC),
 UNIQUE INDEX `userID_UNIQUE` (`userID` ASC),
 INDEX `fk_userUniversity_idx` (`universityID` ASC),
 CONSTRAINT `fk_usertype`
   FOREIGN KEY (`userTypeID`)
   REFERENCES `eventWebsiteDatabase`.`userTypes` (`typeID`)
   ON DELETE NO ACTION
   ON UPDATE NO ACTION,
```

```sql
 CONSTRAINT `fk_userUniversity`
   FOREIGN KEY (`universityID`)
   REFERENCES `eventWebsiteDatabase`.`university` (`universityID`)
   ON DELETE CASCADE
   ON UPDATE NO ACTION)
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `eventWebsiteDatabase`.`RSO`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `eventWebsiteDatabase`.`RSO` ;
CREATE TABLE IF NOT EXISTS `eventWebsiteDatabase`.`RSO` (
 `RSOid` INT NOT NULL AUTO_INCREMENT,
 `RSOname` VARCHAR(45) NULL,
 `RSOdescription` VARCHAR(255) NULL,
 `universityID` INT NULL,
 `adminID` INT NULL,
 PRIMARY KEY (`RSOid`),
 INDEX `fk_rso_university_idx` (`universityID` ASC),
 INDEX `fk_rso_admin_idx` (`adminID` ASC),
 CONSTRAINT `fk_rso_university`
   FOREIGN KEY (`universityID`)
   REFERENCES `eventWebsiteDatabase`.`university` (`universityID`)
   ON DELETE CASCADE
   ON UPDATE NO ACTION,
 CONSTRAINT `fk_rso_admin`
   FOREIGN KEY (`adminID`)
   REFERENCES `eventWebsiteDatabase`.`user` (`userID`)
   ON DELETE NO ACTION
   ON UPDATE NO ACTION)
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `eventWebsiteDatabase`.`eventType`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `eventWebsiteDatabase`.`eventType` ;
CREATE TABLE IF NOT EXISTS `eventWebsiteDatabase`.`eventType` (
 `eventTypeID` INT NOT NULL AUTO_INCREMENT,
 `eventTypeName` VARCHAR(50) NULL,
 PRIMARY KEY (`eventTypeID`))
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `eventWebsiteDatabase`.`event`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `eventWebsiteDatabase`.`event` ;
CREATE TABLE IF NOT EXISTS `eventWebsiteDatabase`.`event` (
 `eventID` INT NOT NULL AUTO_INCREMENT,
 `eventName` VARCHAR(45) NULL,
 `eventStartDate` DATETIME NULL,
 `eventEndDate` DATETIME NULL,
 `eventDescription` VARCHAR(255) NULL,
 `rsoID` INT NULL,
 `adminID` INT NULL,
 `locationID` INT NULL,
 `eventTypeID` INT NULL,
 PRIMARY KEY (`eventID`),
 INDEX `fk_event_rso_idx` (`rsoID` ASC),
```

```sql
 INDEX `fk_event_admin_idx` (`adminID` ASC),
 INDEX `fk_event_location_idx` (`locationID` ASC),
 INDEX `fk_event_type_idx` (`eventTypeID` ASC),
 CONSTRAINT `fk_event_rso`
   FOREIGN KEY (`rsoID`)
   REFERENCES `eventWebsiteDatabase`.`RSO` (`RSOid`)
   ON DELETE CASCADE
   ON UPDATE CASCADE,
 CONSTRAINT `fk_event_admin`
   FOREIGN KEY (`adminID`)
   REFERENCES `eventWebsiteDatabase`.`user` (`userID`)
   ON DELETE CASCADE
   ON UPDATE CASCADE,
 CONSTRAINT `fk_event_location`
   FOREIGN KEY (`locationID`)
   REFERENCES `eventWebsiteDatabase`.`location` (`locationID`)
   ON DELETE NO ACTION
   ON UPDATE NO ACTION,
 CONSTRAINT `fk_event_type`
   FOREIGN KEY (`eventTypeID`)
   REFERENCES `eventWebsiteDatabase`.`eventType` (`eventTypeID`)
   ON DELETE NO ACTION
   ON UPDATE NO ACTION)
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `eventWebsiteDatabase`.`comments`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `eventWebsiteDatabase`.`comments` ;
CREATE TABLE IF NOT EXISTS `eventWebsiteDatabase`.`comments` (
 `commentID` INT NOT NULL AUTO_INCREMENT,
 `comment` VARCHAR(140) NULL,
 `datePosted` DATETIME NULL,
 `eventID` INT NULL,
 `userID` INT NULL,
 PRIMARY KEY (`commentID`),
 INDEX `fk_comment_event_idx` (`eventID` ASC),
 INDEX `fk_comment_user_idx` (`userID` ASC),
 CONSTRAINT `fk_comment_event`
   FOREIGN KEY (`eventID`)
   REFERENCES `eventWebsiteDatabase`.`event` (`eventID`)
   ON DELETE CASCADE
   ON UPDATE NO ACTION,
 CONSTRAINT `fk_comment_user`
   FOREIGN KEY (`userID`)
   REFERENCES `eventWebsiteDatabase`.`user` (`userID`)
   ON DELETE NO ACTION
   ON UPDATE NO ACTION)
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `eventWebsiteDatabase`.`RSO_has_user`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `eventWebsiteDatabase`.`RSO_has_user` ;
CREATE TABLE IF NOT EXISTS `eventWebsiteDatabase`.`RSO_has_user` (
 `RSOid` INT NOT NULL,
 `userID` INT NOT NULL,
```

```sql
 PRIMARY KEY (`RSOid`, `userID`),
 INDEX `fk_RSO_has_user_user1_idx` (`userID` ASC),
 INDEX `fk_RSO_has_user_RSO1_idx` (`RSOid` ASC),
 CONSTRAINT `fk_RSO_has_user_RSO1`
   FOREIGN KEY (`RSOid`)
   REFERENCES `eventWebsiteDatabase`.`RSO` (`RSOid`)
   ON DELETE CASCADE
   ON UPDATE CASCADE,
 CONSTRAINT `fk_RSO_has_user_user1`
   FOREIGN KEY (`userID`)
   REFERENCES `eventWebsiteDatabase`.`user` (`userID`)
   ON DELETE CASCADE
   ON UPDATE CASCADE)
ENGINE = InnoDB;
SET SQL_MODE = '';
GRANT USAGE ON *.* TO dbuser;
DROP USER dbuser;
SET SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';
CREATE USER 'dbuser' IDENTIFIED BY '&z47JGdzgrT*^uG';
GRANT SELECT, INSERT, TRIGGER ON TABLE `eventWebsiteDatabase`.* TO 'dbuser';
GRANT SELECT, INSERT, TRIGGER, UPDATE, DELETE ON TABLE `eventWebsiteDatabase`.* TO
'dbuser';
GRANT ALL ON `eventWebsiteDatabase`.* TO 'dbuser';
#GRANT EXECUTE ON ROUTINE `eventWebsiteDatabase`.* TO 'dbuser';
SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
#Required for proper operation/permissions system;
INSERT INTO userTypes (typeName) Values ('superAdmin'),('admin'),('student');
INSERT INTO eventType (eventTypeName) Values ('public'), ('university'), ('private');
```

Conclusion/Observation:

Our choice to build the app with Nodejs and Express was a good decision because of the large amount of available support and documentation for the framework and library. It was simple to create the endpoints and routes and we were easily able to set up the API. AngularJS was also a good choice for the front end since it allowed us to easily make API calls and control our views. Our database, created with MySQL was again easy to build and integrate. We were able to implement maps and location with the Google Maps API. This allows users to view all of the events at the university they are interested in. We were also able to add and sign in as various users. Another thing that our app is capable of doing is being able to create RSO's along with

events. In terms of problems, we had a few minor hiccups with integration but nothing that was too debilitating. Another small issue that we ran into was the learning curve of the technologies that we used.