

University Events

Course: COP 4710 M/W 6:00pm Summer 2016

Group 7:

Robert Beyhl

Ryan Fonzi

Justin Graham

Table of Contents:


Description.....	1
GUI.....	2
ER diagram.....	4
Relational Model.....	5
Sample data in table form.....	14
SQL examples.....	16
Instructions to Install.....	21
Conclusion.....	22

Project Description:

We were tasked with creating a website that allowed the creation, viewing, and interaction with events affiliated with Universities. This was made possible by allowing creation/management of a University, the creation/management of RSOs, and the creation/management of individual accounts all within our website and stored on an implemented database. These accounts interact with the event functionality and management of the website to control and view the database information.

GUI: Platform, languages, DBMS, screen shots of 'Create RSO,' 'Create Event,' 'View Event,' etc.

Languages used to build the GUI include HTML, CSS, PHP, Javascript, and JQuery.
Languages used to implement the functionality and database include PHP and SQL.



The screenshot shows a web form titled "Create RSO" set against a background of a blue sky with white clouds. The form includes the following elements:

- Title:** "Create RSO" in a stylized, bold font.
- RSO Name:** A text input field.
- University Name:** A text input field.
- Description:** A large text area for the description.
- Email:** Five stacked text input fields, each containing the placeholder "email@website.com".
- Submit Button:** A button labeled "CREATE RSO" at the bottom.

Create Events

time: HHMMSS

date: 20YYMMDD

Event Name

Category

Public

Description

Contact Phone

Event Type

Location Name

Latitude

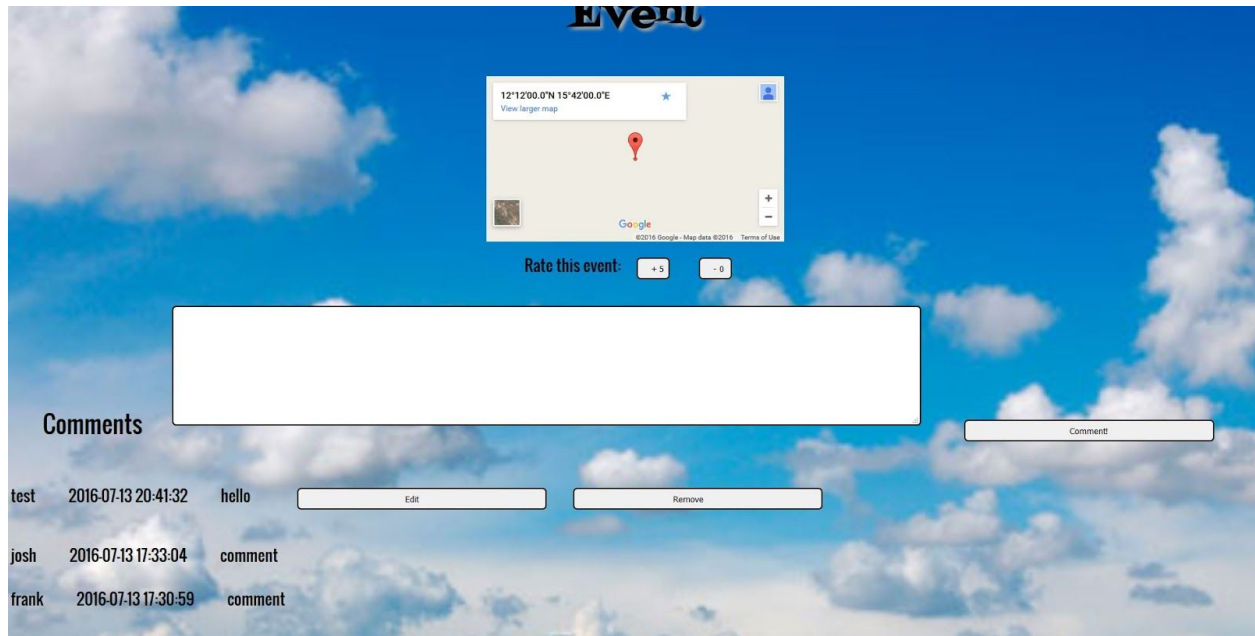
Longitude

CREATE EVENT

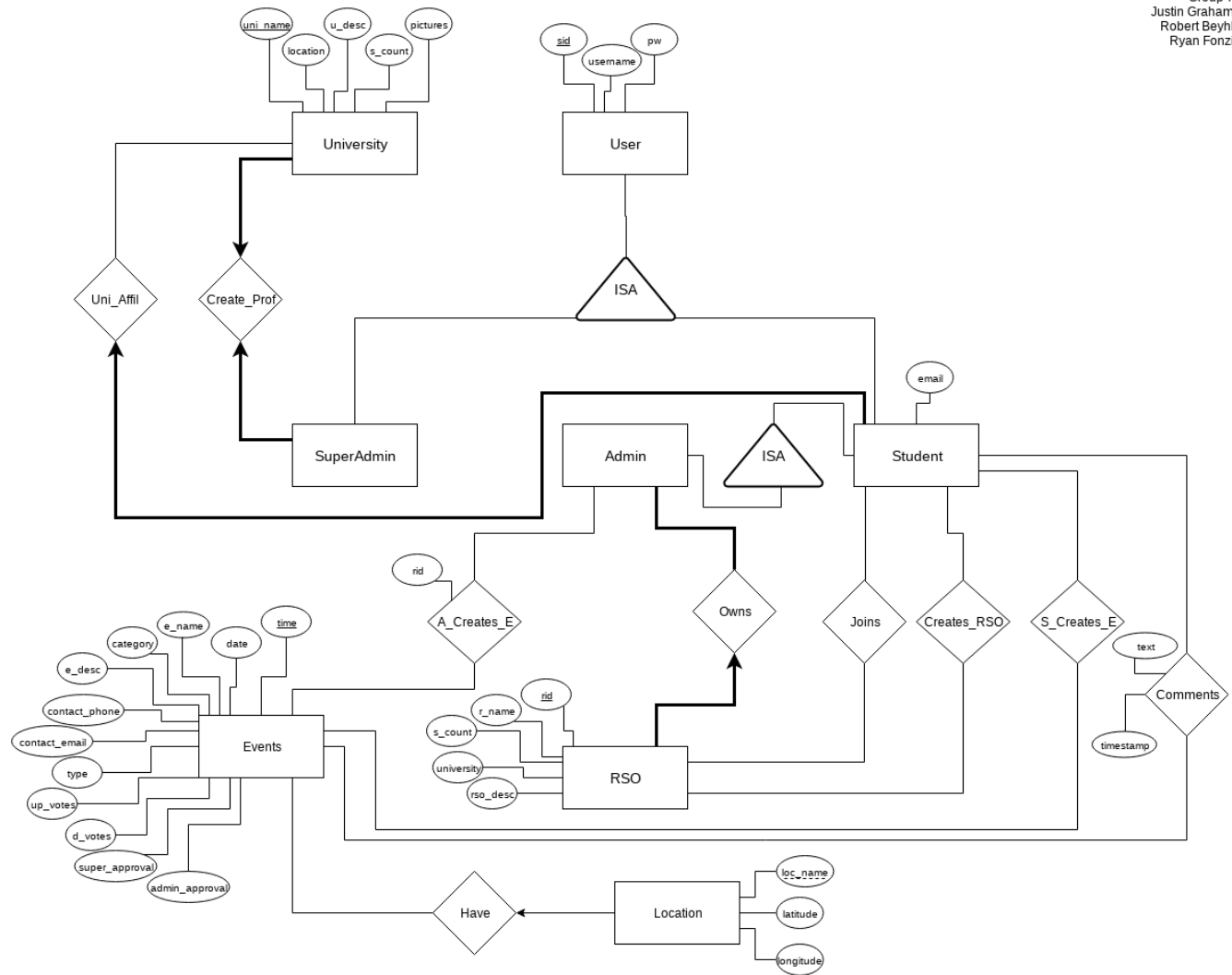
Public Events

Event Name	Date	Event Time
HAPPY	2016-07-24	11:21:06
Event	2016-12-05	12:03:43

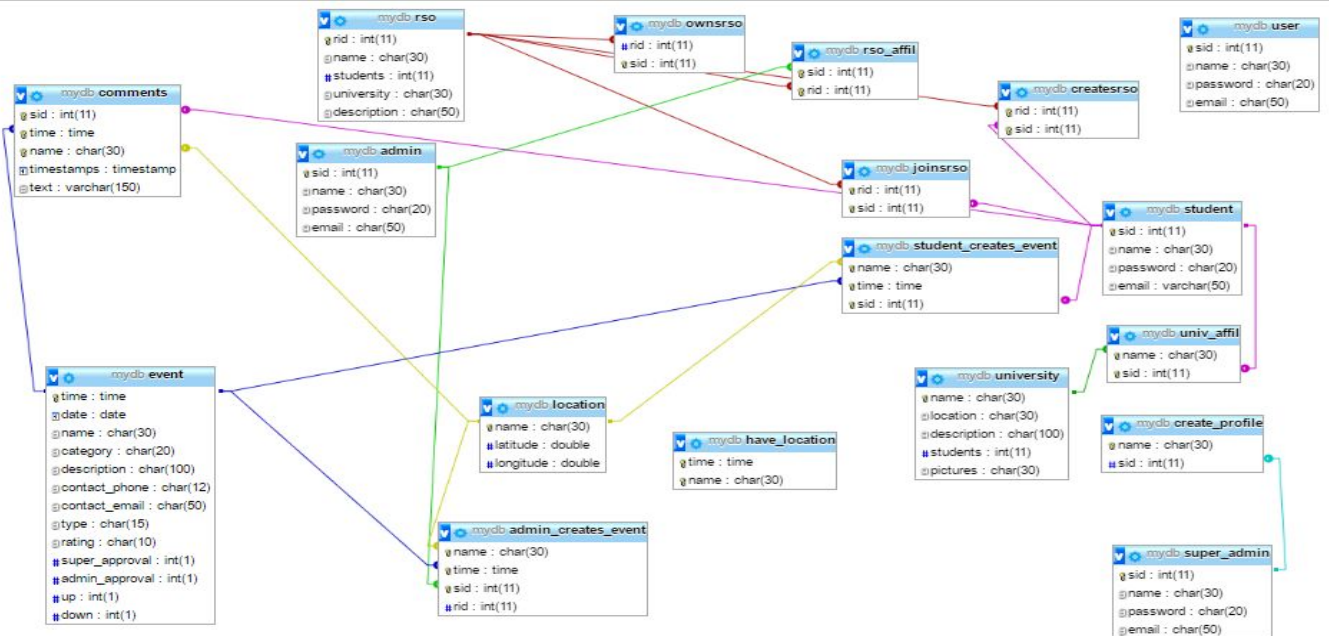
RETURN TO DASH



ER-Model: ER diagram, constraints captured in the ER-model and other general constraints to be enforced by other means, such as assertions, triggers... 3NF decomposition (if required)



The relational data model: SQL 'CREATE TABLE,' 'CHECK,' 'CREATE ASSERTION,' 'CREATE TRIGGER,' as required.



```
--
-- Database: `mydb`
--
--
--
-- Table structure for table `admin`
--
CREATE TABLE IF NOT EXISTS `admin` (
  `sid` int(11) NOT NULL,
  `name` char(30) DEFAULT NULL,
  `password` char(20) DEFAULT NULL,
  `email` char(50) DEFAULT NULL,
  PRIMARY KEY (`sid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
--
--
-- Table structure for table `admin_creates_event`
```

```
--

CREATE TABLE IF NOT EXISTS `admin_creates_event` (
  `name` char(30) NOT NULL DEFAULT "",
  `time` time NOT NULL DEFAULT '00:00:00',
  `sid` int(11) NOT NULL DEFAULT '0',
  `rid` int(11) NOT NULL,
  PRIMARY KEY (`sid`,`time`,`name`),
  KEY `time` (`time`),
  KEY `name` (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
-- -----
```

```
--
-- Table structure for table `comments`
--
```

```
CREATE TABLE IF NOT EXISTS `comments` (
  `sid` int(11) NOT NULL DEFAULT '0',
  `time` time NOT NULL DEFAULT '00:00:00',
  `name` char(30) NOT NULL DEFAULT "",
  `timestamps` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`sid`,`time`,`name`),
  KEY `time` (`time`),
  KEY `name` (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
-- -----
```

```
--
-- Table structure for table `createsrso`
--
```

```
CREATE TABLE IF NOT EXISTS `createsrso` (
  `rid` int(11) NOT NULL DEFAULT '0',
  `sid` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`sid`,`rid`),
  KEY `rid` (`rid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
-- -----
```



```
--
-- Table structure for table `create_profile`
--

CREATE TABLE IF NOT EXISTS `create_profile` (
  `name` char(30) NOT NULL DEFAULT "",
  `sid` int(11) DEFAULT NULL,
  PRIMARY KEY (`name`),
  KEY `sid` (`sid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
-- -----
```

```
--
-- Table structure for table `event`
--
```

```
CREATE TABLE IF NOT EXISTS `event` (
  `time` time NOT NULL DEFAULT '00:00:00',
  `date` date DEFAULT NULL,
  `name` char(30) DEFAULT NULL,
  `category` char(20) DEFAULT NULL,
  `description` char(100) DEFAULT NULL,
  `contact_phone` char(12) DEFAULT NULL,
  `contact_email` char(50) DEFAULT NULL,
  `type` char(15) DEFAULT NULL,
  `rating` char(10) DEFAULT NULL,
  `super_approval` int(1) DEFAULT NULL,
  `admin_approval` int(1) DEFAULT NULL,
  `up` int(1) DEFAULT NULL,
  `down` int(1) DEFAULT NULL,
  PRIMARY KEY (`time`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
-- -----
```

```
--
-- Table structure for table `have_location`
--
```

```
CREATE TABLE IF NOT EXISTS `have_location` (
  `time` time NOT NULL DEFAULT '00:00:00',
  `name` char(30) NOT NULL DEFAULT "",
```

```
PRIMARY KEY (`time`,`name`),  
KEY `time` (`time`),  
KEY `name` (`name`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
-- -----
```

```
--
```

```
-- Table structure for table `joinsrso`
```

```
--
```

```
CREATE TABLE IF NOT EXISTS `joinsrso` (  
  `rid` int(11) NOT NULL DEFAULT '0',  
  `sid` int(11) NOT NULL DEFAULT '0',  
  PRIMARY KEY (`sid`,`rid`),  
  KEY `rid` (`rid`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
-- -----
```

```
--
```

```
-- Table structure for table `location`
```

```
--
```

```
CREATE TABLE IF NOT EXISTS `location` (  
  `name` char(30) NOT NULL DEFAULT "",  
  `latitude` double DEFAULT NULL,  
  `longitude` double DEFAULT NULL,  
  PRIMARY KEY (`name`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
-- -----
```

```
--
```

```
-- Table structure for table `ownsrso`
```

```
--
```

```
CREATE TABLE IF NOT EXISTS `ownsrso` (  
  `rid` int(11) NOT NULL DEFAULT '0',  
  `sid` int(11) NOT NULL DEFAULT '0',  
  PRIMARY KEY (`sid`),  
  KEY `rid` (`rid`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```

-----

--
-- Table structure for table `rso`
--

CREATE TABLE IF NOT EXISTS `rso` (
  `rid` int(11) NOT NULL AUTO_INCREMENT,
  `name` char(30) DEFAULT NULL,
  `students` int(11) DEFAULT NULL,
  `university` char(30) DEFAULT NULL,
  `description` char(50) DEFAULT NULL,
  PRIMARY KEY (`rid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;

-----

--
-- Table structure for table `rso_affil`
--

CREATE TABLE IF NOT EXISTS `rso_affil` (
  `sid` int(11) NOT NULL DEFAULT '0',
  `rid` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`sid`,`rid`),
  KEY `rid` (`rid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-----

--
-- Table structure for table `student`
--

CREATE TABLE IF NOT EXISTS `student` (
  `sid` int(11) NOT NULL AUTO_INCREMENT,
  `name` char(30) DEFAULT NULL,
  `password` char(20) DEFAULT NULL,
  `email` varchar(50) NOT NULL,
  PRIMARY KEY (`sid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;

```

```

-----

--
-- Table structure for table `student_creates_event`
--

CREATE TABLE IF NOT EXISTS `student_creates_event` (
  `name` char(30) NOT NULL DEFAULT "",
  `time` time NOT NULL DEFAULT '00:00:00',
  `sid` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`sid`,`time`,`name`),
  KEY `time` (`time`),
  KEY `name` (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

```

-----

--
-- Table structure for table `super_admin`
--

CREATE TABLE IF NOT EXISTS `super_admin` (
  `sid` int(11) NOT NULL,
  `name` char(30) DEFAULT NULL,
  `password` char(20) DEFAULT NULL,
  `email` char(50) DEFAULT NULL,
  PRIMARY KEY (`sid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

```

-----

--
-- Table structure for table `university`
--

CREATE TABLE IF NOT EXISTS `university` (
  `name` char(30) NOT NULL,
  `location` char(30) DEFAULT NULL,
  `description` char(100) DEFAULT NULL,
  `students` int(11) DEFAULT NULL,
  `pictures` char(30) DEFAULT NULL,
  PRIMARY KEY (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

```

-- -----

--
-- Table structure for table `univ_affil`
--

CREATE TABLE IF NOT EXISTS `univ_affil` (
  `name` char(30) NOT NULL DEFAULT "",
  `sid` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`name`,`sid`),
  KEY `sid` (`sid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- -----

--
-- Table structure for table `user`
--

CREATE TABLE IF NOT EXISTS `user` (
  `sid` int(11) NOT NULL AUTO_INCREMENT,
  `name` char(30) DEFAULT NULL,
  `password` char(20) DEFAULT NULL,
  `email` char(50) DEFAULT NULL,
  PRIMARY KEY (`sid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=2 ;

--
-- Dumping data for table `user`
--

INSERT INTO `user` (`sid`, `name`, `password`, `email`) VALUES
(1, 'test', 'test', 'testemail');

--
-- Constraints for dumped tables
--

--
-- Constraints for table `admin_creates_event`
--
ALTER TABLE `admin_creates_event`

```

```

    ADD CONSTRAINT `admin_creates_event_ibfk_1` FOREIGN KEY (`sid`) REFERENCES
`admin` (`sid`),
    ADD CONSTRAINT `admin_creates_event_ibfk_2` FOREIGN KEY (`time`) REFERENCES
`event` (`time`),
    ADD CONSTRAINT `admin_creates_event_ibfk_3` FOREIGN KEY (`name`) REFERENCES
`location` (`name`);

```

```

--
-- Constraints for table `comments`
--

```

```

ALTER TABLE `comments`
    ADD CONSTRAINT `comments_ibfk_1` FOREIGN KEY (`sid`) REFERENCES `student`
(`sid`),
    ADD CONSTRAINT `comments_ibfk_2` FOREIGN KEY (`time`) REFERENCES `event`
(`time`),
    ADD CONSTRAINT `comments_ibfk_3` FOREIGN KEY (`name`) REFERENCES `location`
(`name`);

```

```

--
-- Constraints for table `createsrso`
--

```

```

ALTER TABLE `createsrso`
    ADD CONSTRAINT `createsrso_ibfk_1` FOREIGN KEY (`sid`) REFERENCES `student`
(`sid`),
    ADD CONSTRAINT `createsrso_ibfk_2` FOREIGN KEY (`rid`) REFERENCES `rso` (`rid`);

```

```

--
-- Constraints for table `create_profile`
--

```

```

ALTER TABLE `create_profile`
    ADD CONSTRAINT `create_profile_ibfk_1` FOREIGN KEY (`sid`) REFERENCES
`super_admin` (`sid`);

```

```

--
-- Constraints for table `joinsrso`
--

```

```

ALTER TABLE `joinsrso`
    ADD CONSTRAINT `joinsrso_ibfk_1` FOREIGN KEY (`sid`) REFERENCES `student` (`sid`),
    ADD CONSTRAINT `joinsrso_ibfk_2` FOREIGN KEY (`rid`) REFERENCES `rso` (`rid`);

```

```

--
-- Constraints for table `ownsrso`
--

```

```

ALTER TABLE `ownsrso`
  ADD CONSTRAINT `ownsrso_ibfk_2` FOREIGN KEY (`rid`) REFERENCES `rso` (`rid`);

--
-- Constraints for table `rso_affil`
--
ALTER TABLE `rso_affil`
  ADD CONSTRAINT `rso_affil_ibfk_1` FOREIGN KEY (`sid`) REFERENCES `admin` (`sid`),
  ADD CONSTRAINT `rso_affil_ibfk_2` FOREIGN KEY (`rid`) REFERENCES `rso` (`rid`);

--
-- Constraints for table `student_creates_event`
--
ALTER TABLE `student_creates_event`
  ADD CONSTRAINT `student_creates_event_ibfk_1` FOREIGN KEY (`sid`) REFERENCES
`student` (`sid`),
  ADD CONSTRAINT `student_creates_event_ibfk_2` FOREIGN KEY (`time`) REFERENCES
`event` (`time`),
  ADD CONSTRAINT `student_creates_event_ibfk_3` FOREIGN KEY (`name`) REFERENCES
`location` (`name`);

--
-- Constraints for table `univ_affil`
--
ALTER TABLE `univ_affil`
  ADD CONSTRAINT `univ_affil_ibfk_1` FOREIGN KEY (`name`) REFERENCES `university`
(`name`),
  ADD CONSTRAINT `univ_affil_ibfk_2` FOREIGN KEY (`sid`) REFERENCES `student` (`sid`);

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;

```

Populating tables with sample data:

Table with added student info

SQL Buddy interface showing a table with student information.

Navigation: > Home, > Users, > Query, > Import, > Export

DATABASES: information_schema, mydb

mydb tables: admin (0), admin_creates_event (0), comments (4), createsrso (0), create_profile (0), event (2), have_location (2), joinsrso (0), location (2), ownsrso (1), rso (0), rso_affil (0), student (6), student_creates_event (0), super_admin (0), university (3), univ_affil (3), user (5)

Select: All None With selected: Edit Delete Refresh

	sid	name	password	email
<input type="checkbox"/> i	4	test	test	test@email.com
<input type="checkbox"/> i	6	frank	pass	franky@mail.com
<input type="checkbox"/> i	8	josh	pass	email@email
<input type="checkbox"/> i	9	Johnny Cage	123kungfume	JCage@outworld.net
<input type="checkbox"/> i	10	Ash Ketchum	ichooseyou	Pikapika@PalletTown.com
<input type="checkbox"/> i	11	Eren Yeager	Titan'sMe	attackthistitan@yahoo.com

Table with added university info

SQL Buddy interface showing a table with university information.

Navigation: > Home, > Users, > Query, > Import, > Export

DATABASES: information_schema, mydb

mydb tables: admin (0), admin_creates_event (0), comments (4), createsrso (0), create_profile (0), event (2), have_location (2), joinsrso (0), location (2), ownsrso (1), rso (0), rso_affil (0), student (6), student_creates_event (0), super_admin (0), university (3), univ_affil (3), user (5)

Select: All None With selected: Edit Delete Refresh

	name	location	description	students	pictures
<input type="checkbox"/> i	Great Scott University	1985 Outatime way	Youre gunna see some serious		
<input type="checkbox"/> i	UCF	UCF	UCF	3	
<input type="checkbox"/> i	University of College Florida	1000 address ln	a great way to schooling!		

Table with RSO info added

SQL Buddy Browse (4) Structure Insert Query Import Export

Select: All None With selected: Edit Delete Refresh

	rid	name	students	university	description
<input type="checkbox"/> i	2	rso	1	univ	desc
<input type="checkbox"/> i	3	Doctor's Orders	6	Great Scott University	Ready to go back in time!
<input type="checkbox"/> i	4	Gators R us	6	University of College Florida	Who wants to swim with these
<input type="checkbox"/> i	5	Hackersive Writing	6	UCF	Teaches you breaking the interwebs

DATABASES

- information_schema
- mydb
 - admin (0)
 - admin_creates_event (0)
 - comments (4)
 - createsrso (0)
 - create_profile (0)
 - event (2)
 - have_location (2)
 - joinsrso (0)
 - location (2)
 - ownsrsro (1)
 - rso (4)
 - rso_affil (0)
 - student (6)
 - student_creates_event (0)
 - super_admin (0)
 - university (3)
 - univ_affil (3)
 - user (5)

Table with added event info

SQL Buddy Browse (2) Structure Insert Query Import Export Logout

Select: All None With selected: Edit Delete Refresh

	time	date	name	category	description	contact_phone	contact_email	type
<input type="checkbox"/> i	11:21:06	2016-07-24	HAPPY	public	yeppee	4072678312	franky@gmail.com	FUN
<input type="checkbox"/> i	12:03:45	2016-12-03	Event	Public		4072678312	test@email.com	Xmas

DATABASES

- information_schema
- mydb
 - admin (0)
 - admin_creates_event (0)
 - comments (4)
 - createsrso (0)
 - create_profile (0)
 - event (2)
 - have_location (2)
 - joinsrso (0)
 - location (2)
 - ownsrsro (1)
 - rso (4)
 - rso_affil (0)
 - student (6)
 - student_creates_event (0)
 - super_admin (0)
 - university (3)
 - univ_affil (3)
 - user (5)

Table with added comment info

SQL Buddy Browse (5) Structure Insert Query Import Export Logout

Select: All None With selected: Edit Delete Refresh

	sid	time	name	timestamps	text
<input type="checkbox"/> i	4	12:03:45	UCF	2016-07-13 20:41:32	hello
<input type="checkbox"/> i	6	11:21:06	Arkansas	2016-07-13 17:38:49	comment
<input type="checkbox"/> i	6	12:03:45	UCF	2016-07-13 17:30:59	comment
<input type="checkbox"/> i	8	11:21:06	Arkansas	2016-07-13 17:39:37	frank is a bonehead
<input type="checkbox"/> i	8	12:03:45	UCF	2016-07-13 17:33:04	comment

DATABASES

- information_schema
- mydb
 - admin (0)
 - admin_creates_event (0)
 - comments (5)
 - createsrso (0)
 - create_profile (0)
 - event (2)
 - have_location (2)
 - joinsrso (0)
 - location (2)
 - ownsrsro (1)
 - rso (4)
 - rso_affil (0)
 - student (6)
 - student_creates_event (0)
 - super_admin (0)
 - university (3)
 - univ_affil (3)
 - user (5)

SQL examples and results:

SQL statement to insert a new RSO (part of the processing of the 'Create RSO' form), show results

```
INSERT INTO rso (name, students, university, description)
VALUES('$rname', 1, '$uni_name', '$rso_desc')
```

Returns true or false based on if the query was completed successfully or not.

SQL statement to insert a new student to an existing RSO (part of the processing of the 'Join RSO' form), show results

```
INSERT INTO joinsrso (rid, sid)
VALUES('$rid', '$user->sid')
```

Inserts user into relational table for joining the RSO. Returns true if query was completed successfully.

```
UPDATE rso set students = students + 1 WHERE rid = '$rid'
```

Increments the number of students for the RSO the student joined. Returns true if query was completed successfully.

SQL statement to insert a new event (part of the processing of the 'Create Event' form), show results

```
INSERT INTO event (time, date, name, category, description, contact_phone, contact_email,
type, super_approval, admin_approval, up, down)
VALUES('$time', '$date', '$e_name', '$category', '$e_desc', '$contact_phone',
'$contact_email', '$type', 1, 1, 0, 0)
```

Inserts event into event table. For demonstration purposes they are pre approved. Returns true if query was completed successfully.

```
INSERT INTO location (name, latitude, longitude)
VALUES('$loc_name', '$latitude', '$longitude')
```

Inserts into location table. Returns true if query was completed successfully.

```
INSERT INTO have_location (time, name)
VALUES('$time', '$loc_name')
```

Inserts into have_location relational table. Returns true if query was completed successfully.

```
INSERT INTO student_creates_event (name, time, sid)
VALUES('$loc_name', '$time', '$user->sid')
```

Inserts into student_creates_event relational table. Returns true if query was completed successfully.

For RSO events, queries are the same except instead of inserting into the student_creates_event table we insert into the admin_creates_event table

SQL statement to insert/update a (new) comment (part of the processing of the 'Create/Add/Modify Comment' form), show results

```
INSERT INTO comments (sid, time, name, timestamps, text)
VALUES ('$sid', '$time', '$name', now(), '$text')
```

Inserts into the comments relational table. Returns true if query was completed successfully. The now() function created a timestamp with the current time and date.

```
UPDATE comments SET text = '$text' WHERE sid = '$sid' AND name = '$name' AND time = '$time'
```

Updates the comments relational table. Returns true if query was completed successfully.

Several SQL queries to display events—public, private, and RSO-- (part of the processing of the 'View Event' request by a user with a specific role), show results

PUBLIC EVENTS

```
SELECT *
FROM event WHERE category = 'public'
```

We then match up place name with time in the relational table, returns true if query was completed successfully.

```
SELECT *
FROM event WHERE time = '$timehelper'
```

And put the data into objects, returns true if query was completed successfully.

```

$final[] = new Event_Location ($row["time"], $row["date"], $row["name"], $row["category"],
$row["description"], $row["contact_phone"],
    $row["contact_email"], $row["type"], $row["up"], $row["down"],
$row["super_approval"], $row["admin_approval"], $row2["name"],
    $row2["latitude"], $row2["longitude"]);

```

PRIVATE EVENTS

```

SELECT *
    FROM event WHERE category = 'private'

```

Gets all private events.

```

SELECT *
    FROM have_location

```

Gets all entries in the relational table.

```

for($i = 0; $i < count($locations); $i++)
{
    $timehelper2 = $locations[$i]->time;
    $namehelper2 = $locations[$i]->name;
    $sql = "SELECT sid
    FROM student_creates_event WHERE name = '$namehelper2' AND time =
' $timehelper2'";
    $result = $conn->query($sql);
    $row3 = $result->fetch_assoc();

    $searchsid = $row3["sid"];
    $result->close();

    $sql = "SELECT name
    FROM univ_affil WHERE sid = '$searchsid'";
    $result = $conn->query($sql);
    $row3 = $result->fetch_assoc();
    //return $row3["name"];
    if($row3["name"] != $student_uni)
    {
        unset($locations[$i]);
    }
}

```

```
}
```

Filters out events from other universities.

Events are then returned as objects.

```
for($i = 0; $i < count($locations); $i++)
{
    //events
    $timehelper = $locations[$i]->time;
    $sql = "SELECT *
    FROM event WHERE time = '$timehelper'";
    $result = $conn->query($sql);

    //locations
    $namehelper = $locations[$i]->name;
    $sql = "SELECT *
    FROM location WHERE name = '$namehelper'";
    $result2 = $conn->query($sql);

    $row = $result->fetch_assoc();
    $row2 = $result2->fetch_assoc();

    $final[] = new Event_Location ($row["time"], $row["date"], $row["name"],
    $row["category"], $row["description"], $row["contact_phone"],
    $row["contact_email"], $row["type"], $row["up"], $row["down"],
    $row["super_approval"], $row["admin_approval"], $row2["name"],
    $row2["latitude"], $row2["longitude"]);

    $result->close();
    $result2->close();
}
```

RSO

```
SELECT *
    FROM event WHERE category = 'RSO'
```

Gets all RSO events

```
$sql = "SELECT *
    FROM have_location";
$result = $conn->query($sql);
```

```

//get all place names
while($row = $result->fetch_assoc())
{
    for($i = 0; $i < count($times); $i++)
    {
        if($row["time"] == $times[$i])
        {
            $locations[] = new EventHelper($row["name"], $times[$i]);
        }
    }
}

```

Gets all place names

```

//filter
for($i = 0; $i < count($locations); $i++)
{
    $timehelper2 = $locations[$i]->time;
    $namehelper2 = $locations[$i]->name;
    $sql = "SELECT rid
    FROM admin_creates_event WHERE name = '$namehelper2' AND time =
'$timehelper2'";
    $result = $conn->query($sql);
    $row3 = $result->fetch_assoc();

    //$searchrid = $row3["rid"];
    $result->close();

    for($j = 0; $j < count($studentrso); $j++)
    {
        if($row3["rid"] == $studentrso[$j])
        {
            $rso_loc[] = $locations[$j];
        }
    }
}

```

Filters out data from RSOs not joined by the user.

Software installation: one folder (directory) contains source code and required libraries to run the app, and installation instructions. A '*.sql' file containing SQL statements to set up the environment (database name, user, password, etc.) and create the database (CREATE TABLE, but no sample data needed, i.e., no INSERT statements) is to be included here.

1. Install and set up (if necessary) your *AMP stack (WAMP for Windows, LAMP for Linux).
 - a. Make sure Apache and MySQL database services are running
2. Move all files pertaining to the app to C:\path\to\wamp\www\ in WAMP, or /path/to/lamp/apache2/htdocs/ in LAMP.
3. Open up a browser and go to localhost/phpmyadmin if using WAMP, or localhost:8080/phpmyadmin if using LAMP.
4. Log in as root without a password if on WAMP. On LAMP, you should have set up a password during installation (or manually disabled needing to log in with one), so use that instead.
5. Click "new" on the lefthand panel and create a new database.
6. Click on the new database you created on the lefthand panel, and navigate to the import tab.
7. Click on "Choose File" and navigate to the "mydb.sql" file which was included in our project files. Click on "Go"
8. You're done with phpmyadmin for now. Click on the address bar on your browser and navigate to localhost/index.php if using WAMP, or localhost:8080/index.php if using LAMP.
9. The website should be displaying now.

Conclusion/Observation: o Database performance: query response time, suggested indexes o Desired features/functionality: security (login), event feed from university websites (e.g., XML feeds from <http://events.ucf.edu/>), social network integration (Facebook, Twitter, etc.) o Problems encountered, things that have been learned from the project, more things/skills needed to master to build a more advanced database app

Our database queries were quick and responsive for the number of entries we tested. In order to scale our database up for thousands of users, we would be required to make more intricate measurements pertaining to response time at different server loads with large numbers of entries, which is outside the scope of this project.

Our login system has basic security, but more care would need to be given before this app could be deployed. Relying on session tokens and encryption instead of keeping track of the user object between pages would need to be implemented. Event feeds from university websites was a planned feature, but we did not have enough experience to implement such a

feature in the timeframe we were given. With enough time though, an event feed could be implemented. More social network integration in general was planned, such as the ability to log in via Facebook or Google, and posting events to your wall on Facebook. Other social media features such as posting to twitter could also be implemented.

Most of the problems we encountered stemmed from our inexperience with web development before this project. There are several things we now know we can implement to make development simpler, such as integrating frameworks like Bootstrap for the UI, taking advantage of AJAX for running PHP scripts for SQL queries via Javascript, and implementing AngularJS for more dynamic webpages.