

## **Group Report**

**COP 4710 Summer 2018**

**Application: College Event Website**

**Group Four: Freddy Haug, Jie Xiong, Meng Zhang**

<b>1.....</b>	<b>Introduction</b>
<b>2.....</b>	<b>ER Diagram</b>
<b>3.....</b>	<b>Create Table</b>
<b>4.....</b>	<b>Front-end Design and Back-end Development</b>
<b>5.....</b>	<b>Conclusion</b>

## **1. Introduction**

Each university usually has a website where events are posted. However, there are several limitations for those websites. First, one has to check the website in order to add each event to his/her calendar; second, not all events across the university are included; third, one cannot track weekly events.

To overcome the limitations of the university websites, we have created a web application for event scheduling. A student can register using this application to obtain a user ID and a password. There are three user levels: super admin who creates a profile for a university (name, location, description, number of students, pictures, etc.), admin who owns a registered student organization (RSO) and may host events, and student who uses the application to look up information about the various events.

Fig.1 shows our technical approaches. We first identified the problems as mentioned above, based on which a feasibility study was performed. Second, we did a logical system design using ER

diagram and relational schemes. Third, we implemented the model using different technologies. Finally, we tested and debugged the app and accordingly tuned the app.

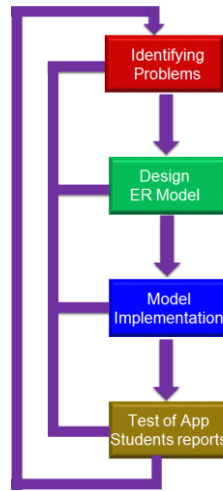


Fig. 1. Technical approaches for creating college event website.

The following is a summary of technologies used for creating the app:

**Front End:** CSS, HTML, JavaScript

**Back End:** PHP, MySQL, Apache, jQuery, AJAX

**Language:** HTML, CSS, PHP, JavaScript

**Database:** MySQL

**APIs:** Geocode, Google Maps, Axios

## 2. ER diagram and relational schemes

Fig. 2.1 shows the designed ER diagram. There are three levels: super-admin, admin, and students. Super admin can create a profile for a university including university id, name, location, description, number of students, pictures. Super-admin can create both public events and private events. Admin can manage an RSO and hold events. Admin can also create both public events and private events. Admin can create events with name, event id, event category, description, time, location, and contact email address. Each admin is affiliated with one university, and one or more RSOs. A student user can either create a new RSO or to join an existent one. A new RSO can be

created with at least 5 other students with the same email domain, and one of them should be assigned as an administrator. Students can view university events by selecting either location or the University where they can view the event they are interested in. Students users are able to see all the events from RSOs that they are following or around their locations.

There are three different types of events: each event can be public, private, or an RSO event. Public events can be seen by everyone; private events can be seen by the students of the host university; and an RSO events can only be seen by members of the RSO. Events can also be created without an RSO as long as the events are approved by the super admin. After an event has been published, users can rate and edit comments on the event.

Fig. 2.2 shows a relational schema, which can help organize and understand the structure of a database. The table relationships were created in MySQL database using phpmyadmin.

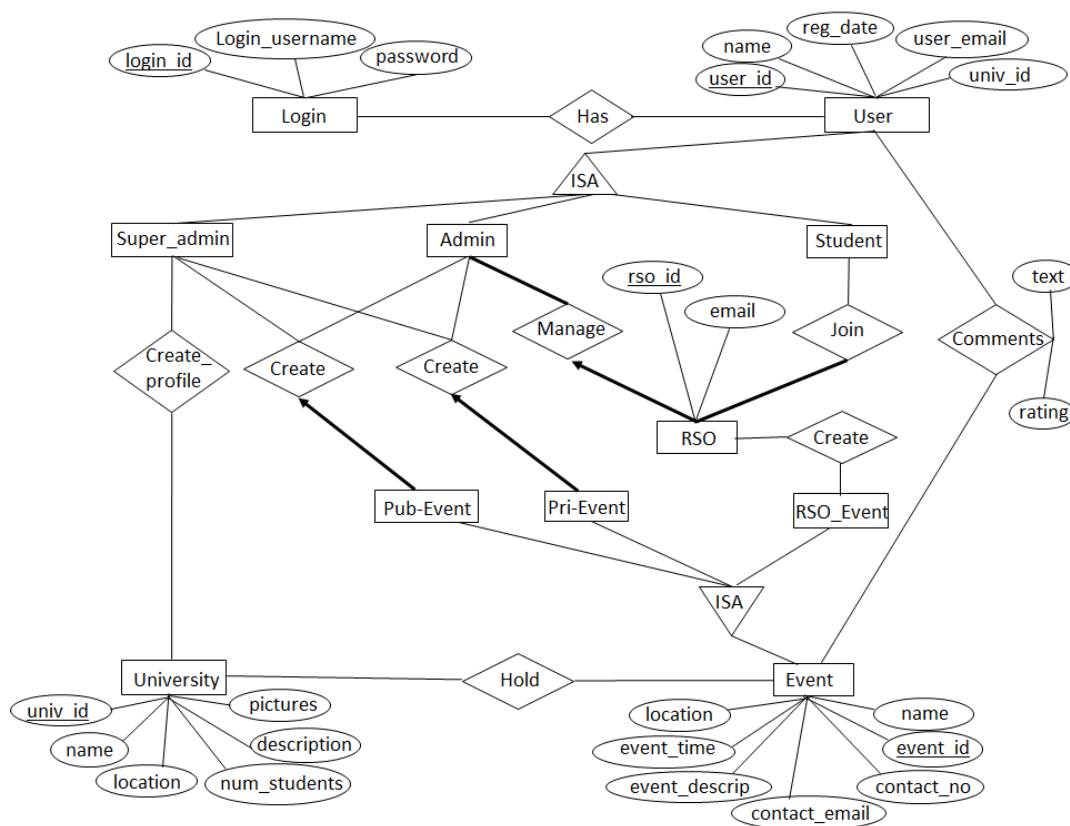


Fig. 2.1 ER diagram

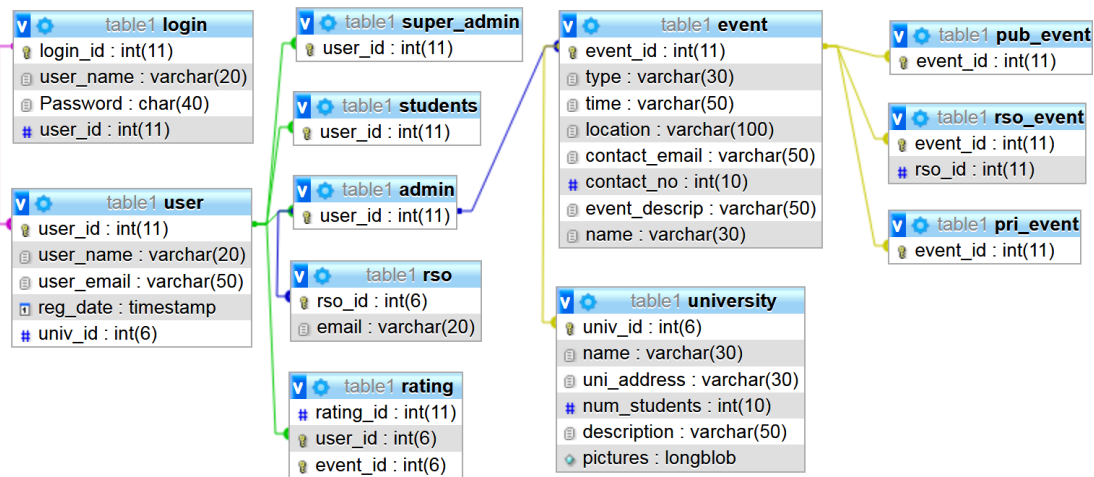


Fig. 2.2 Relational schemes.

### 3. Create Table

The platform for managing database and connecting database to interface is PHPMYADMIN. Basically, five tables were created under EVENT file. These tables are Event, Rating, RSO, University and Users. These tables directly reflect the entities and relationship between them, which we already clarified in our ER diagram.

Table	Action	Rows	Type	Collation	Size	Overhead
event	Browse Structure Search Insert Empty Drop	0	InnoDB	latin1_swedish_ci	16 KiB	-
rating	Browse Structure Search Insert Empty Drop	0	InnoDB	latin1_swedish_ci	16 KiB	-
rso	Browse Structure Search Insert Empty Drop	3	InnoDB	latin1_swedish_ci	16 KiB	-
university	Browse Structure Search Insert Empty Drop	2	InnoDB	latin1_swedish_ci	16 KiB	-
users	Browse Structure Search Insert Empty Drop	6	InnoDB	latin1_swedish_ci	16 KiB	-
<b>5 tables</b>	<b>Sum</b>	<b>11</b>	<b>InnoDB</b>	<b>latin1_swedish_ci</b>	<b>80 KiB</b>	<b>0 B</b>

Fig. 3.1 Five Tables in Database

Since our database records all the information from our interface, it is very important for us to set appropriate variables that display in our designed websites. Generally speaking, most of these information are related to two important tables: Event table and Users table.

For table of Users, it reflects all the information about three kinds of users: super admin, admin and student user. The attributes of USERS are User id, First name, Last name, Email, university

id, and image. Of all properties of table USERS, two attributes will be used repeatedly for login. They are Email and password. In addition, the primary key for table USERS is Event id.

```
CREATE TABLE `users` (
  `id` int(11) NOT NULL,
  `firstName` varchar(30) NOT NULL,
  `LastName` varchar(30) NOT NULL,
  `Email` varchar(80) NOT NULL,
  `password` text NOT NULL,
  `university_id` text NOT NULL,
  `image` text NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Figure 3.2 SQL for Creating Table EVENT

For table of Event, it displays all related information corresponding to kinds of events for different type. The attributes of Event are event id, event name, event type, event time, event date and event location. Of all properties of EVENTS, attribute of Event type is related to table RSO. And Event location is related to table University, which could be dynamically displayed by google map.

```
CREATE TABLE `event` (
  `event_id` int(10) UNSIGNED NOT NULL,
  `name` varchar(30) CHARACTER SET armSCII8 COLLATE armSCII8_bin NOT NULL,
  `type` varchar(30) CHARACTER SET armSCII8 COLLATE armSCII8_bin NOT NULL,
  `event_time` varchar(20) CHARACTER SET armSCII8 COLLATE armSCII8_bin NOT NULL,
  `event_date` date NOT NULL,
  `Location` int(100) NOT NULL,
  `university_id` varchar(3) CHARACTER SET armSCII8 COLLATE armSCII8_bin NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Fig. 3.3 SQL for Creating Table EVENT

Table University, RSO and Rating are concerned to event type, event location and university, which are three basic attributes of table Event. Therefore, more details about them will be mentioned in the chapter of Designing event pages.

## 4 Front-end Design and back-end development

### 4.1 Login and registration page design—welcome to university event website

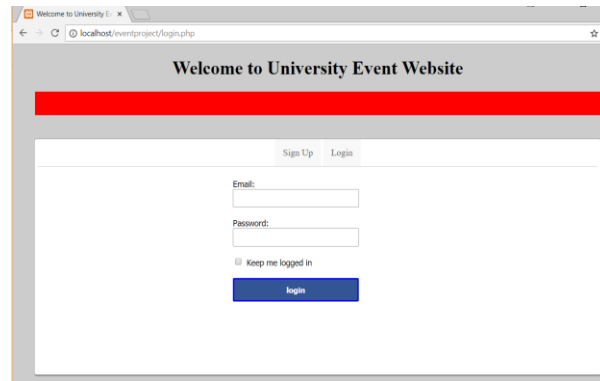


Fig. 4.1 Home Page

Frontend design involves creating the HTML,CSS, which creating experience for users to search for their interested university event. Back-end development involves using MYSQL to manage our event database. More important, PHP creates a dynamical connection between interface and our event database, convert data from graphical interface to values in tables. Generally speaking, HTML,CSS,PHP and java are popular used in interface and back-end development and also used to create our university event page.

Figure 4.1 is the event home page we designed. The layout is very simple and clear. The title in this page in the center shows this is a welcome page. And the head of white frame reflects two vital functions of this page, one is sign up and the other is login. A new user is not able to login this system until she/he has registered. Hence first step for all users is registration by pressing sign up button and fill out the blanks in registration page.

iversity Event Website

First Name:  
Jacky

Last Name:  
Green

Email:  
Jack@hotmail.com

Password:  
\*\*\*\*\*

Re-enter Password:  
\*\*\*\*\*

University\_id:  
UCF

Image:  
Choose File 三个.png

Submit

Fig. 4.2 Registration Page

Figure 4.2 is the screenshot of my registration page. A new user need input first name, last name, Email, Password, university id and image as basic individual information for registration. Once Jacky in this example submitting his information in the page, the PHP and SQL could convert his information to the values of the attributes in table USERS by this queries:

*INSERT INTO users*

*(Firstname, Lastname, Email, password, University\_id, image)*

*Values (Jacky, Green, jack@Hotmail.com, 12341234,UCF)*

At this time, a new record has been inserted into table USERS which is highlighted by dark blue in Table 4.1. Normally, there will be a remainder appear in the red frame of registration page, which is ‘ You are successfully registered.’

Table 4.1. Table USERS

id	firstName	LastName	Email	password	image
5	Caraline	Laul	caraline@gmail.com	123456789	22820181.jpg
6	Catherine	SUE	CSUE@email.com	123456789	IMG_20161105_10
1	Mike	Green	mikegreen@hotmail.com	12345678	1528660592(1).png
4	Meng	zhang	meng@hotmail.com	12345678	WeChat Image_201
2	Jie	Xiong	jiexiong4159@knights.ucf.edu	12341234	1528660592(1).png
3	jane	Gong	jane@hotmail.com	12341234	a,%a,.png
7	Jacky	Green	jack@hotmail.com	\$2y\$10\$HfipY2f8nebt	501228535648421

Successfully registration means that you have your personal account to login. To continue our searching, we skip to our home page and login with a registered Email and password.

Sign Up

Login

---

Email:

jiexiong4159@knights.ucf.edu

Password:

.....

☐ Keep me logged in

login

Fig. 4.2 Login Page

In the example of Figure 4.2, the account name (email) jiexiong4159@knights.ucf.edu logged in with correct password. PHP connect this information with database and SQL send the query to Mysql. The query is:

*SELECT password FROM users*  
*WHERE Email='jiexiong4159@knights.ucf.edu*

Jie	Xiong	jiexiong4159@knights.ucf.edu	12341234	1528660592(1).png
-----	-------	------------------------------	----------	-------------------

Fig. 4.3 The Results of Select Query

After receiving this query, mysql search the result from table USERS and the results of this query are showed in figure 4.3. If the password in the table equal to the password input in login page



which are both equal to 12341234, PHP could return the command to website and then enter in the event page.

## 4.2 Validate variables in registration page

```
if(strlen($firstName)<3)
{
    $error="First name is too short";
}
else if(strlen($LastName)<3)
{
    $error="Last name is too short";
}
else if(!filter_var($Email,FILTER_VALIDATE_EMAIL))
{
    $error="Please enter valid email address";
}
else if(strlen($password)<8)
{
    $error="password must be greater than 8 character";
}
else if($password != $passwordConfirm)
{
    $error="password does not match";
}
else if($image=="")
{
    $error="please upload your image";
}
```

Fig. 4.4 Codes for Validation Input Values

Obviously, in Table 4.1, new record has totally different password format comparing to other users with 1 to 6. That's because the hash has applied to the password by which makes much safer for users. Except the hash of password, some restrictions has been set to validate the input variables. Such as the length of first name and last name should greater than 3. The Email should in a correct format. The password should contains 8 or more characters. Details of these validation are described in the codes of this part (in figure 4.4). And the most important validation of all are all the variables are not null.

## 4.3 Event page design

The event page took up a very long time in the development cycle. In front-end development we were tasked with creating a design and also implementing all of the functionalities. Once we had connected to the database through our PHP pages, we were able to manipulate tables and personalize web template functions. In the beginning I had to find a way for logged in users to consistently communicate with the database so that events could be filed into the correct sections and so that users could find the correct events. This was before I had implemented the RSO applications in the web app. To get around this I discovered the `_SESSION()` function which

helped keep user information between pages and actions. With this in play I was able to check a user's privilege and display the correct "Create Event" for users of different status within our web app.

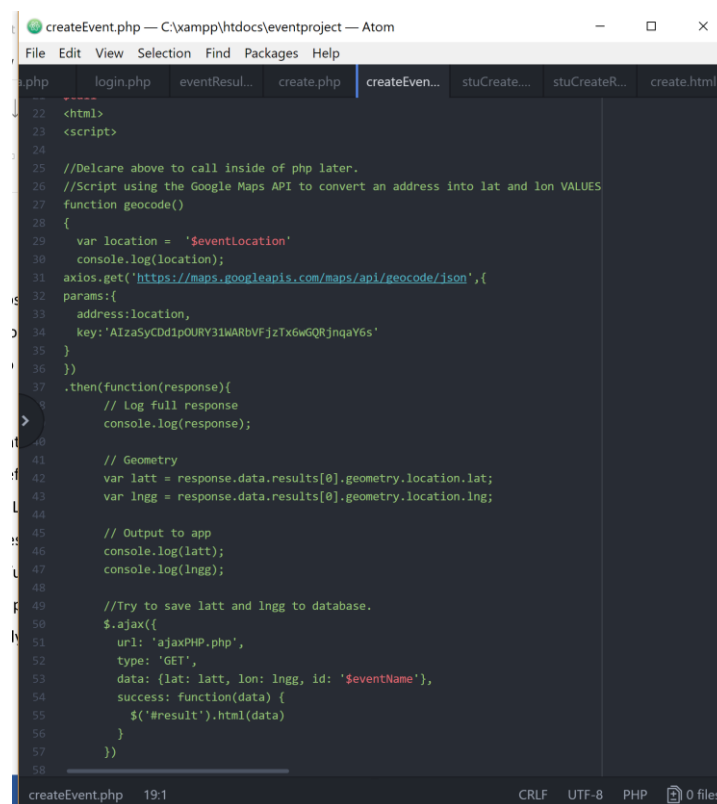
For our application, I decided to let admins and super admins create events immediately. This allowed for a more fluid event development that I believe students would be able to make use of. None admin users would be able to create events too. However, students would be notified that they were not able to immediately publish their events. Their created events would be emailed to the super admin of the user's university for further review. If the Super admin approved of this event, they could go to the create event page and create the event for the student user.

The screenshot shows a web browser window with the URL `localhost/eventproject/stuCreate.php`. The browser's address bar and tabs are visible at the top. The page has a navigation bar with links: Profile, Search Event, RSOs, Create Event, and Log out. Below the navigation bar, a message box states: "You are not the admin of any RSO. You can request to create an event. This information will be sent to your university super admin for review before being uploaded to the site." The main content area contains a form with the following fields: Event Name (text input), Type (dropdown menu with "Social" selected), Time (text input), Date (text input with a date picker icon), Scope (dropdown menu with "Private" selected), Location (text input), Contact Number (text input), and Short Description (text input). A blue "Submit" button is located at the bottom of the form.

*Figure 4.5 "Create Event Page"*

Another limitation I put in this section of our website was the "Type" field. I wanted to help make the search event function more user friendly. To do this we limited the amount of event types admin's can pick. This limitation allows users who are searching for events to be able to find every type of event offered by our application.

One of the most difficult parts of this assignment was the implementation of the geocoder and google maps API. I had never used JavaScript before and it was a difficult language to get used to. Through a mess of JavaScript inside of HTML inside of PHP echo commands, I was able to convert whatever input a user put into the address field into longitude and latitude coordinates. I inserted this information into my database for future search event use. Below I have included a picture of this code. It is the most difficult to appreciate part of our code but I am proud that I was able to overcome this challenge and correctly capture the information I intended to get.



```
createEvent.php — C:\xampp\htdocs\eventproject — Atom
File Edit View Selection Find Packages Help
login.php eventResul... create.php createEven... stuCreate... stuCreateR... create.html
22 <html>
23 <script>
24
25 //Declare above to call inside of php later.
26 //Script using the Google Maps API to convert an address into lat and lon VALUES
27 function geocode()
28 {
29     var location = '$eventlocation'
30     console.log(location);
31     axios.get('https://maps.googleapis.com/maps/api/geocode/json',{
32     params:{
33         address:location,
34         key:'AIzaSyCDdipOURY3lWARbVFjzTx6wGQRjnqaY6s'
35     }
36 })
37 .then(function(response){
38     // Log full response
39     console.log(response);
40
41     // Geometry
42     var latt = response.data.results[0].geometry.location.lat;
43     var lngg = response.data.results[0].geometry.location.lng;
44
45     // Output to app
46     console.log(latt);
47     console.log(lngg);
48
49     //Try to save latt and lngg to database.
50     $.ajax({
51         url: 'ajaxPHP.php',
52         type: 'GET',
53         data: {lat: latt, lon: lngg, id: '$eventName'},
54         success: function(data) {
55             $('#result').html(data)
56         }
57     })
58 }
```

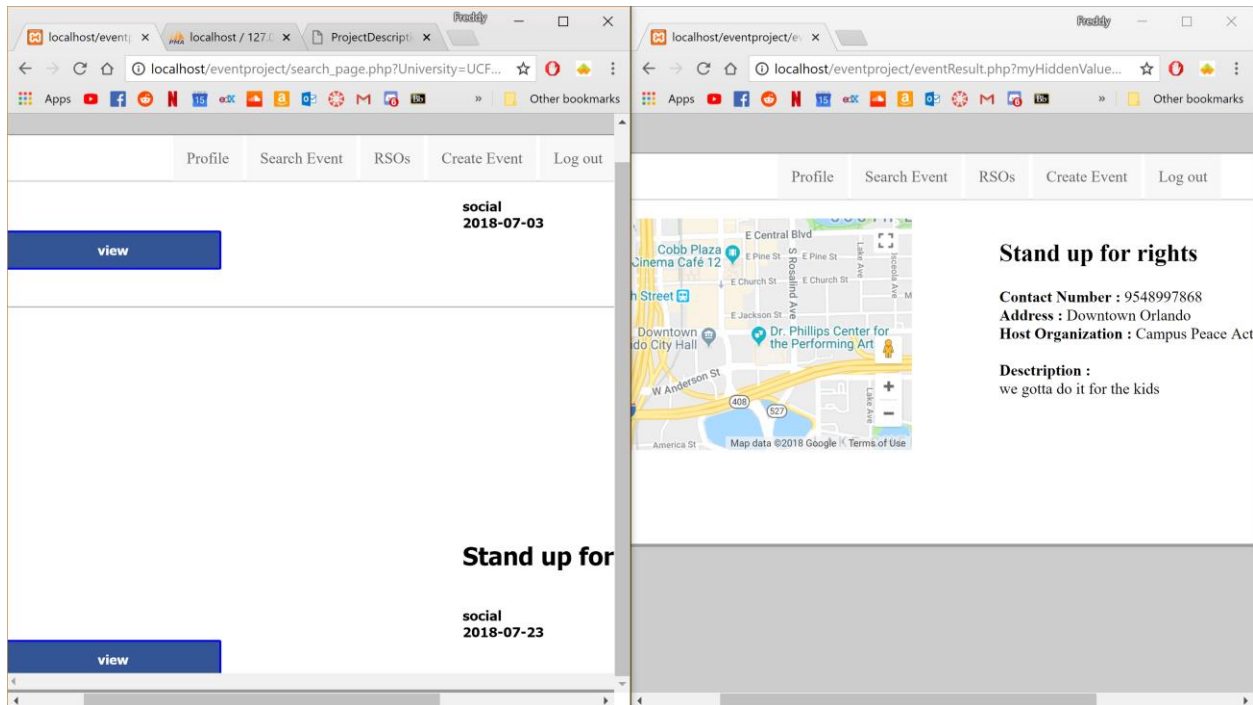
Figure 4.6 jQuery, Geocoder implementation

This brings me to the discussion of redesign. As I was the only group member with prior programming experience. I had to review the ER diagram and change attributes inside of table and remove or include some tables. A very interesting problem that comes with web application design for a front-end and back-end team is how often we had to revise our previous work. There

were many times where I would be working with an early design and discover that there was some connection or attribute I missed that was essential to future plans we had for the application. This type of work was a reminder that finished work is never truly finished. When trying to implement the create and search event pages, I must have deleted and re-added the address attribute three times.

After the finishing the event creation page, I was able to begin working on the event search engine. This came with a fair share of problems as well. I had a design in mind that would display the google maps view for each event after the events requested by the user were displayed. The problem with this was that Google's map API only allows for one call per element. This ended up being a problem I could not solve in time. I must have spent 8 hours trying everything to get around this issue but the fundamental problem stayed. I could not call a dynamic number of div containers in HTML. Here I decided that I could not keep trying to fix this problem.

My solution to this was to simply redesign my vision to work with this limitation. In the end, I made the search results a separate page, where users can add a banner photo in future updates to the app. Once users found an event they want more information on, they could open up the specific event page and take in all the information for that event. Here, a user would be able to see the location of the event on a map as well as write down the actual address and call or email the RSO leader with convenience. Below I have included a screenshot of the search results page as well as an event page.



#### 4.7 Search Result pages

This design is a scrollable and dynamic experience. I think browsing students will have a pleasant and easy time with our UX. In the pictures above, I would like to point out that the “profile” tab was switched to a “feed” tab. This is the landing page for users after they sign into the website and its function is to let the users scroll through all events in their university that are public or within the user’s scope.

#### 4.4 RSO Implementation

After completing all of the event functions, I began working on RSO functions. When I began, I was overwhelmed. I thought about all of the functions RSO brought to the web app and started to worry that all of my event work would become obsolete. At the beginning it was, but thankfully after a few hours of tweaking everything around and creating more functionality, I was able to fuse the two parts of the project together.

Our RSO page lets any user create a new RSO, and also gives the user the ability to pick any of the five email address to become the admin of that RSO. This information is immediately

included into our RSO table, and it also updates our user table. Whichever user is chosen to be the admin will have their *priv* level moved from a 1 to a 2. If they are already a Super Admin (Priv 3) their authority will stay the same.

Once an RSO is created, it is included into the RSO page. Here you will find all the other student organizations inside of the same university that the current user is registered under. Below every RSO element, a user can join an RSO. Joining an RSO will update our membership table and will let the user search and find private events hosted by this RSO.

## **5.1 Conclusion/Observations**

Before I created the membership table, I was holding so much information inside of the *user* table. I found the true application of clever database design during this project. With clever database relations and entities, your code can be reduced drastically and you can also open the door to more uses and relations within the app. Web development is a very interesting cycle of creation and destruction.

During the development of this application we ran into many road blocks. Our group had a few logistic issues as the summer semester can be riddled with home moving and planned vacations. We were able to create a fully functional application. Unfortunately, a lot of the blockages were not able to be inserted into the application. If we could continue working on the project I would include some restrictions. It would also benefit our web app greatly if we were to implement higher security features in the instance of a SQL injection attack. I assumed it would be easy to insert security restraints at the end but we see now that it is more effective and easier to implement them as we develop the application.