Group Report, College Event Website

COP4710-0V01 Spring 2021

Group 46

Alexandra French, Zefang Yao, Jialin Zheng

Table of Contents

Project Description

This project aims to solve the problem of there being many different university event systems that do not communicate with each other. These mean students signing up on many different sites or not even seeing events for certain universities that they do not sign up for. It also separates many students from public events that may be in their area but not setup as an event at their university. Thus, this project involves creating a college event management website for students of all universities to join and get involved with many different events.

The solution to the problem of project event management come in the form of a web application. This application is intended to be usable on any of the common web browsers: Firefox, Chrome, and Edge. Any student or faculty member can register with the application to create their own user ID and password. The students are able to view events, their profile, as well as join RSOs and Universities. Students can become an admin of an RSO, that is then able to create events. Finally, faculty are super admins, and they are able to create universities and approve events that need approval.

Students' exact capabilities include viewing all university events that are public, an event in an RSO they belong to, or a private event in a university they belong to. As such, students are also able to view all RSOs by name or university, and join/leave ones in their university. Similarly, students can view all universities by name and location, as well as join/leave a particular university.

Admins are students, and each admin belongs to only one university and however many RSOs they may be an admin of. Admins are able to create events. These can be either public (everyone can view), private (only users in the university can view), or RSO (only users in the RSO can view). RSOs are created by a student with at least 5 other students with the same email domain, and the admin can be specified during this creation as one of the students.

Super admins are able to create universities. They can also delete the university they have created, giving them more agency. The universities must have names, descriptions, locations, number of students, and any number of pictures the super admin wishes to add. Super admins can also approve events as needed. Any event made without being an "RSO" event will need to be approved before being shown to anyone but the super admin. These events can also be rejected.

Each event have a name, description, categorie, access scope (public, private, RSO), location, time, date, contact phone number, and contact email address. The location can be confirmed on a Google map during creation. Events can be searched according to a users' scope. The event search allows for searching by location name, latitude and longitude, or university. Any event made without being a "RSO" event will require super admin approval. Each event can be shared to social media, including Facebook and Twitter, from their detail pages.

Additional Assumptions

In terms of project instructions, a non-SQL database and ORM usage was determined to be allowed through class and email correspondence. At the instruction of the professor, it was assumed that the use of Django model was acceptable. Considering this, model-view interfaces were used in conjunction with a SQLite database. We also assumed that example SQL tables and queries may be helpful, and thus created these and stored in their own folder “SQL Code”.

In terms of the RSO member count, specifications seemed vague. Thus it was assumed that the admin counted as a member. It was also assumed that an “active” RSO includes 5 members, with the admin included as a member. And also assumed that admin must create an “active” RSO, which means the RSO must have 5 members when it is created.

In terms of university student count, it was never specified if this should increase when people join the site or just be the university’s overall student count. Thus, we assumed it was the universities’ overall student count, to be set by the super admin and unchanged by students joining/leaving university on the website.

In terms of event approval without RSO, instructions on how events need to be made with an RSO were somewhat unclear. Thus, we assumed that any event made without being an RSO event needs to be approved. This means any public event or any private event that is not also an RSO event must be approved by a super admin.

GUI

The GUI for the College Event Site exists in the form of a website that can be viewed in a web browser. For the purposes of this assignment, the server can be downloaded and run locally. The website is compatible with Edge, Chrome, and Firefox.

The GUI was created using HTML, CSS, JS, and Bootstrap. HTML/CSS/Bootstrap provided the main stylings for the website. JS was used to achieve some button functionality on certain pages. For example, the create university and create event pages both have a google map that needs to update with the users' input when pressing a button. This was handled via a javascript method. Another such example would be the capability for adding additional emails on the create RSO page and photos on the create university page.

The database management system (DBMS) used was SQLite. This was picked for its compactness, usability with Django, an ease of use. It also made sharing and populating tables throughout development simplet.

The GUI consists of starting up pages, such as the login home page and the user registration page. After that, there are 4 internal page areas. Events, which contains pages to view and create (if an Admin) events. Universities, which contains the abilities to view, join, and leave different universities. RSOs, which contains pages for creating, viewing, joining, leaving, and (if a creator) deleting registered student organizations (RSOs). Finally there is a profile page, where users can view their info and join or leave universities and RSOs. Super-admin users, also referred to as "faculty", have additional pages and functionality to approve events as well as create and delete universities.

Login and Registration GUI

Upon opening the website, users will be greeted by a login page, where they can login if they already have an account. There are two tiers of login here: The basic student login and the faculty login. Students are able to sign up so that they can view events, join universities, and join/form RSOs. Faculty are able to sign up so that they can create universities for users to be able to join. There is also a “register” link for new users to be able to get started by going to the registration form.

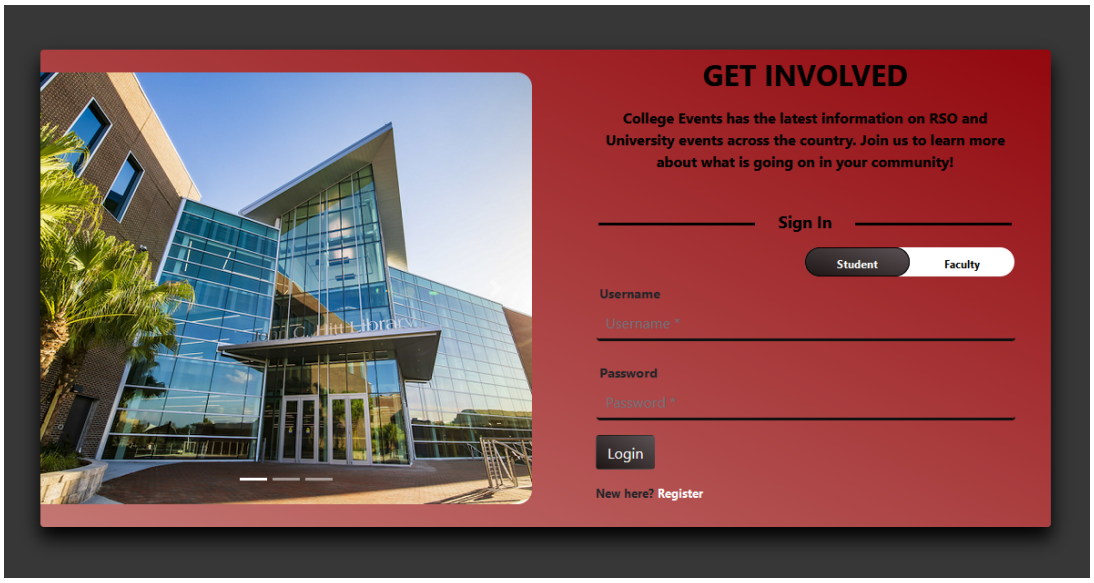


Figure 1: Login page

The user registration page allows for signing up as one of the two standard user levels: a student or a faculty member. The user can select the one they wish to go with.

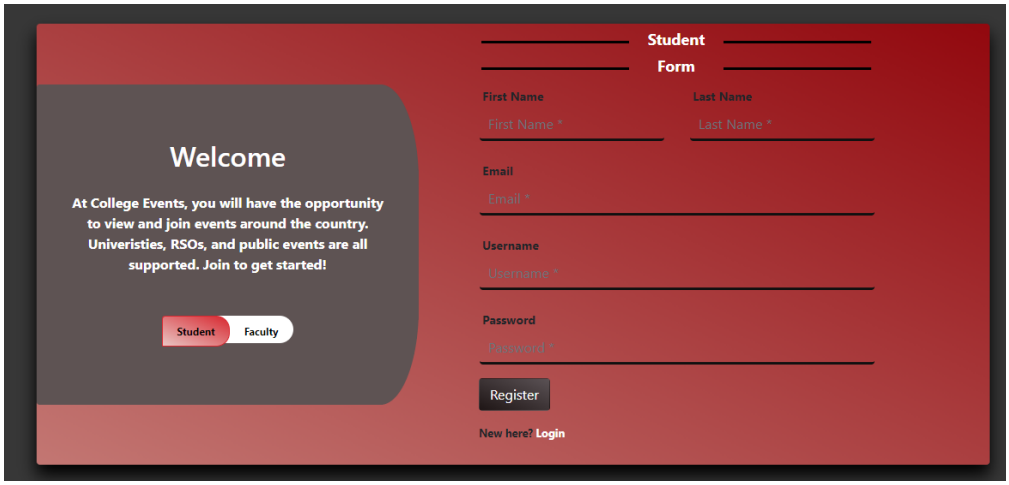


Figure 2: Registration Page

Event GUI

Users can visit a view events page. This lists all approved events that are currently in the system. Users will be able to view events by their permission levels: users can view all public events, events in their university, and events in their RSO. Users can also search for events by location using latitude and longitude, as well as by location name and either “all” or the users’ university. Users can get more details on an event by pressing the “details” link.

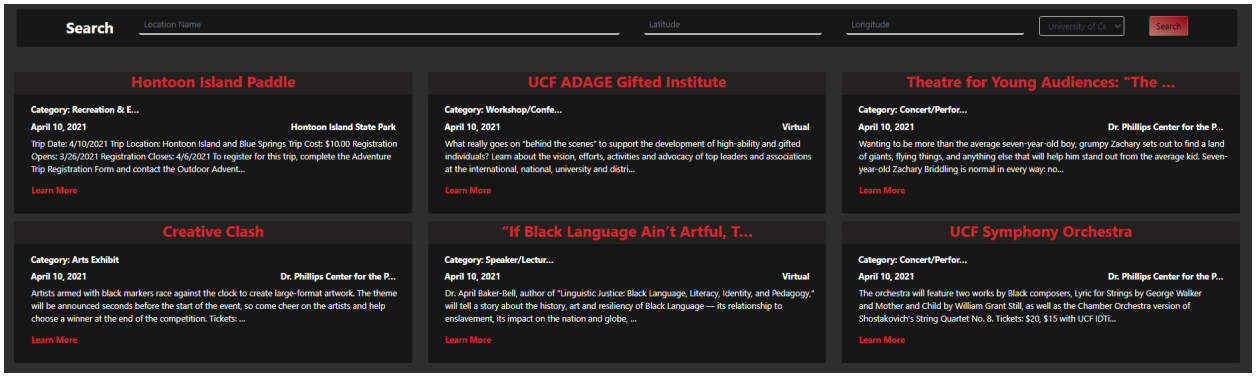


Figure 3: Viewing Events

Super admins (also called faculty) are able to approve events. When an event is made without an RSO, it must be approved by the super admin here. The super admin can view all events needing approval. They can either approve the event through the “approve” button, or deny it through the “deny” button.

Figure 4: Approve Events Page

On the details page for an event, users can view all available information on the event, including its location. They are also able to share the event to facebook or twitter via some share buttons. Users can also make, edit, and delete their own comments for the event. Finally, while commenting, users can rate the event on a system of 1-5 stars.

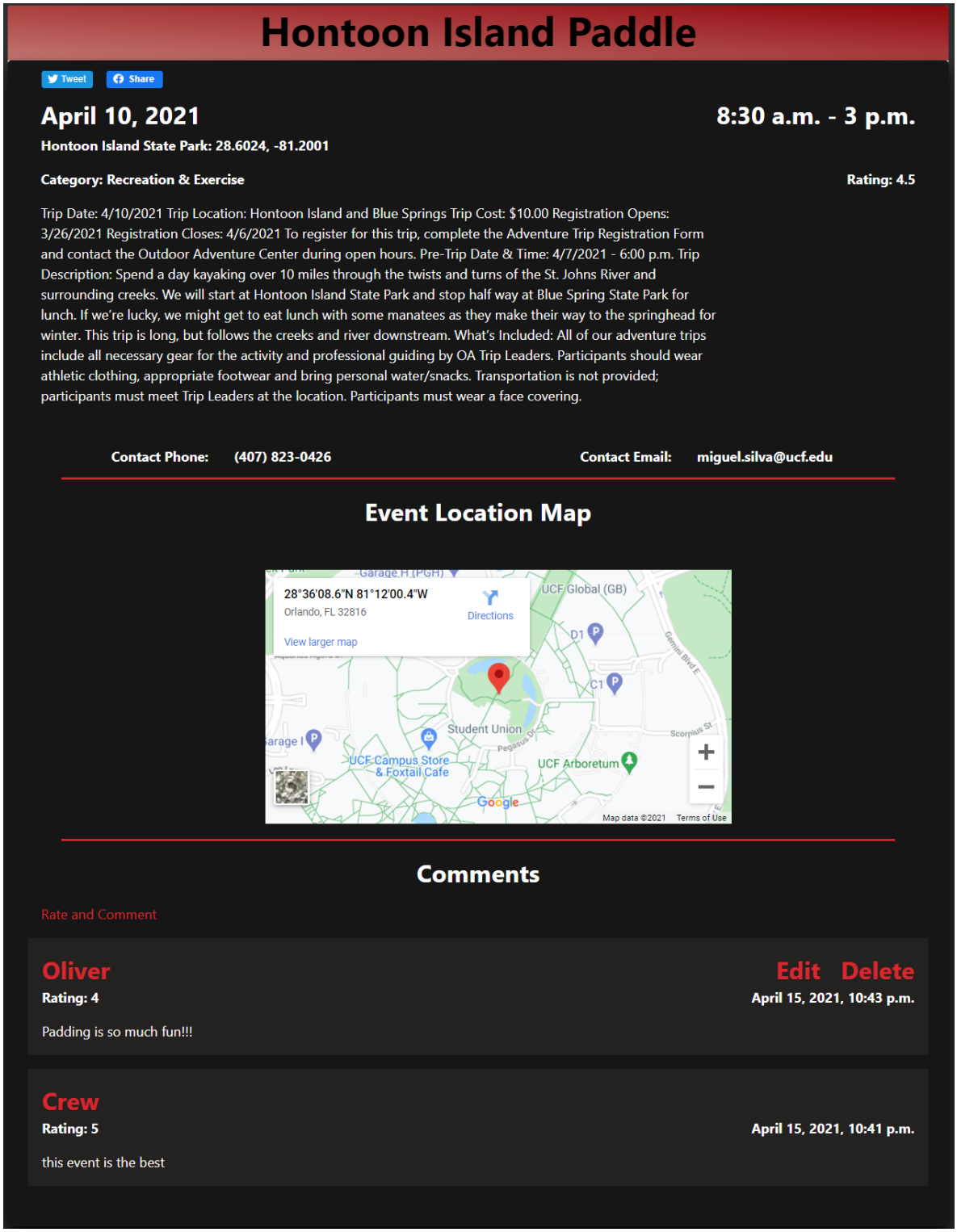


Figure 5: Viewing Event Details Page as User Oliver

RSO admins are able to create events through the create events page. These can be either public, RSO, or private (university) events. The event will need to have a name, category, description, start/end time, date, contact information in the form of a phone number and email, and an event location that can be confirmed on a google map.

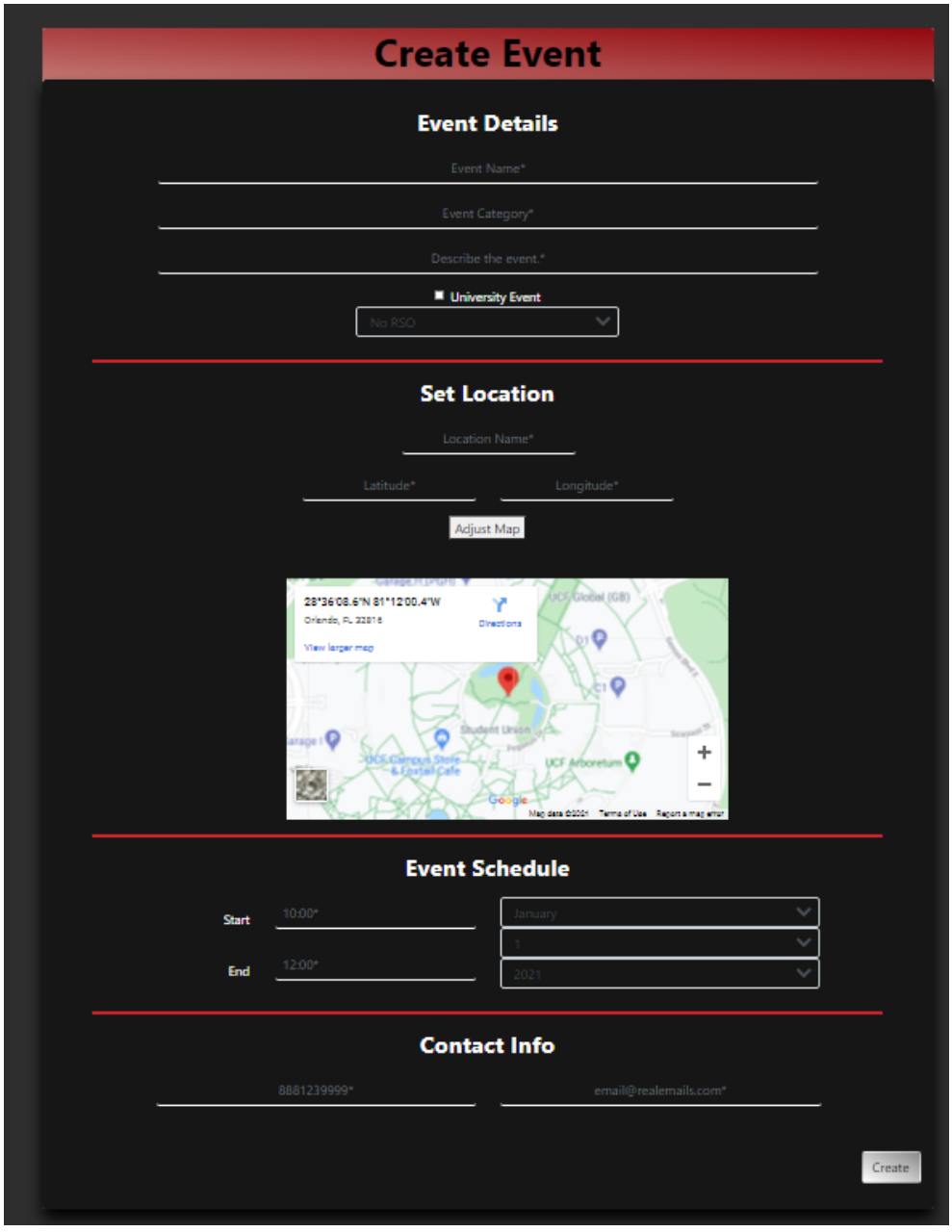


Figure 6: Create Events Page

RSO GUI

The view RSO page allows users to view registered student organizations (RSOs). They can search RSOs by name or by either all or the students' university. Users can also go to a RSO details page to view more information on the RSO.

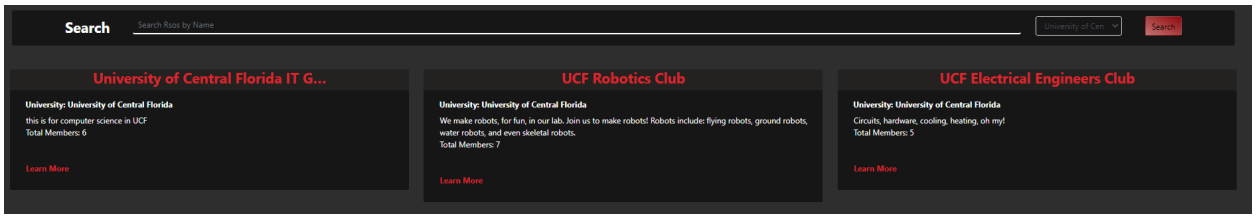


Figure 7: View RSO Page

The view inactive RSO page shows all RSOs that have turned inactive due to a lack of members. This is any RSO with <5 members outside of the admin of the RSO. It functions similarly to the view RSO page otherwise.

Figure 8: View Inactive RSO Page

The RSO details page shows all the details of an RSO, including its description, university, and name. Users can join the RSO through this screen, as well as leave an RSO. RSO admins are able to delete their RSO through this screen.

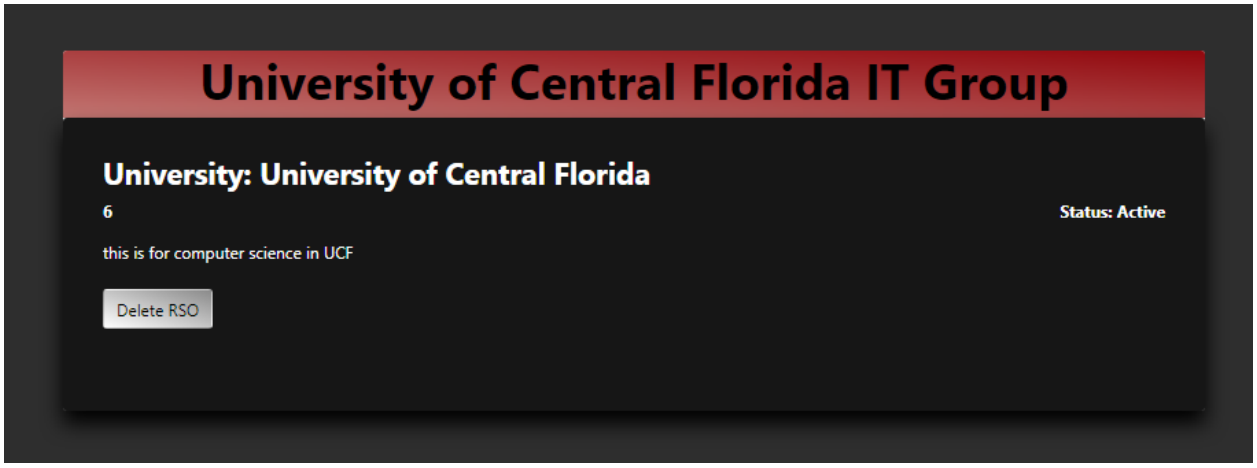


Figure 9: RSO Details Page

The create RSO page allows users to create an RSO with a name, description, university, and the needed number of member emails. Since an RSO is not active until it has a certain number of members, users must have this many members to begin one. The administrator can also be chosen here by entering the relevant email. All emails must have the same email domain as the administrator. Finally, if more users are wanted, then the user can add emails through a button.

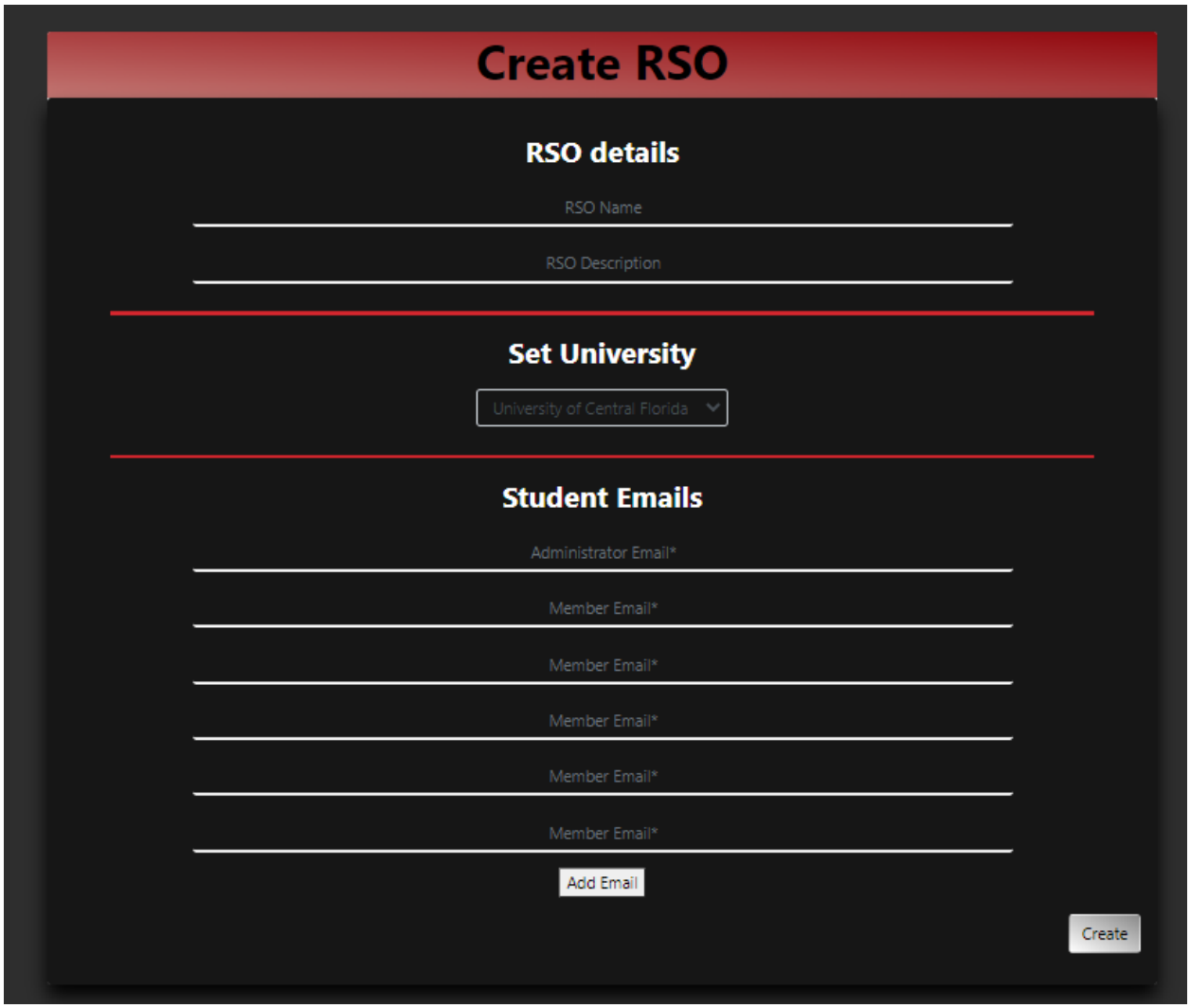


Figure 10: Create RSO Page

University GUI

Users are able to view all available universities through the “view universities” page. This will show all universities in the system. The user can also view details about the university through the “learn more” button. They are also able to search for universities by the university name, location name, or latitude/longitude combination.

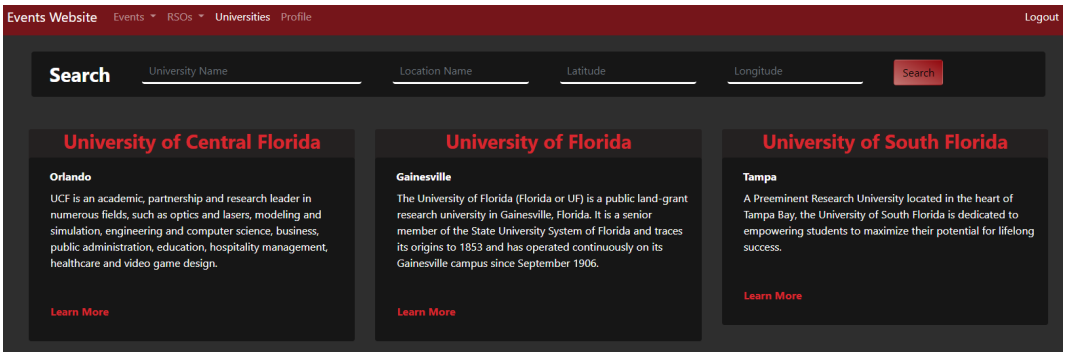


Figure 11: View Universities Page

The university details page shows the user details of the university, including student number, location, map of location, and university photos. Student/admin users can join/leave universities. Super admins that own the university can delete it.

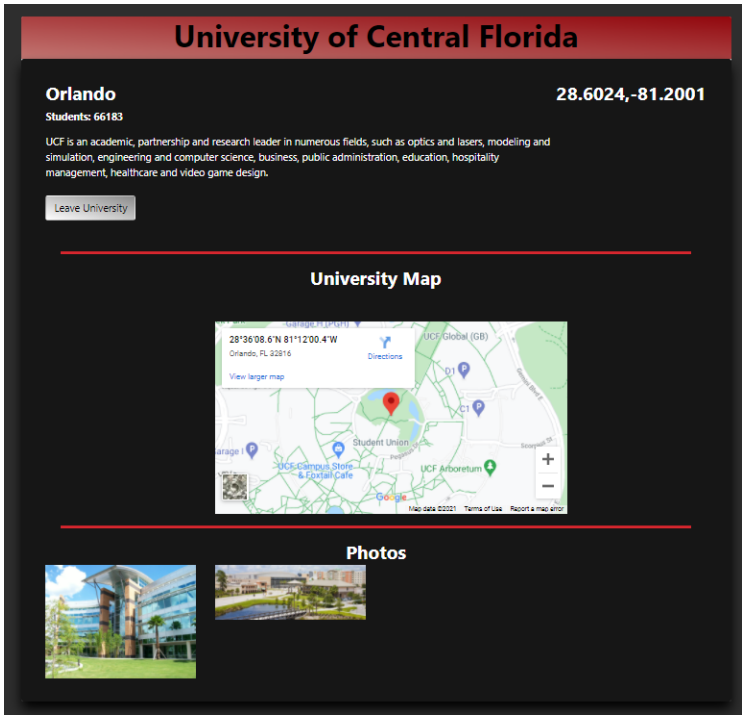


Figure 12: University Details Page

The create university page allows a super admin to create a university. It involves adding a university name, description, student body count, location that can be checked with a map, and university photos. There can be any number of photos added, with any additional photo wanted added by pressing a button.

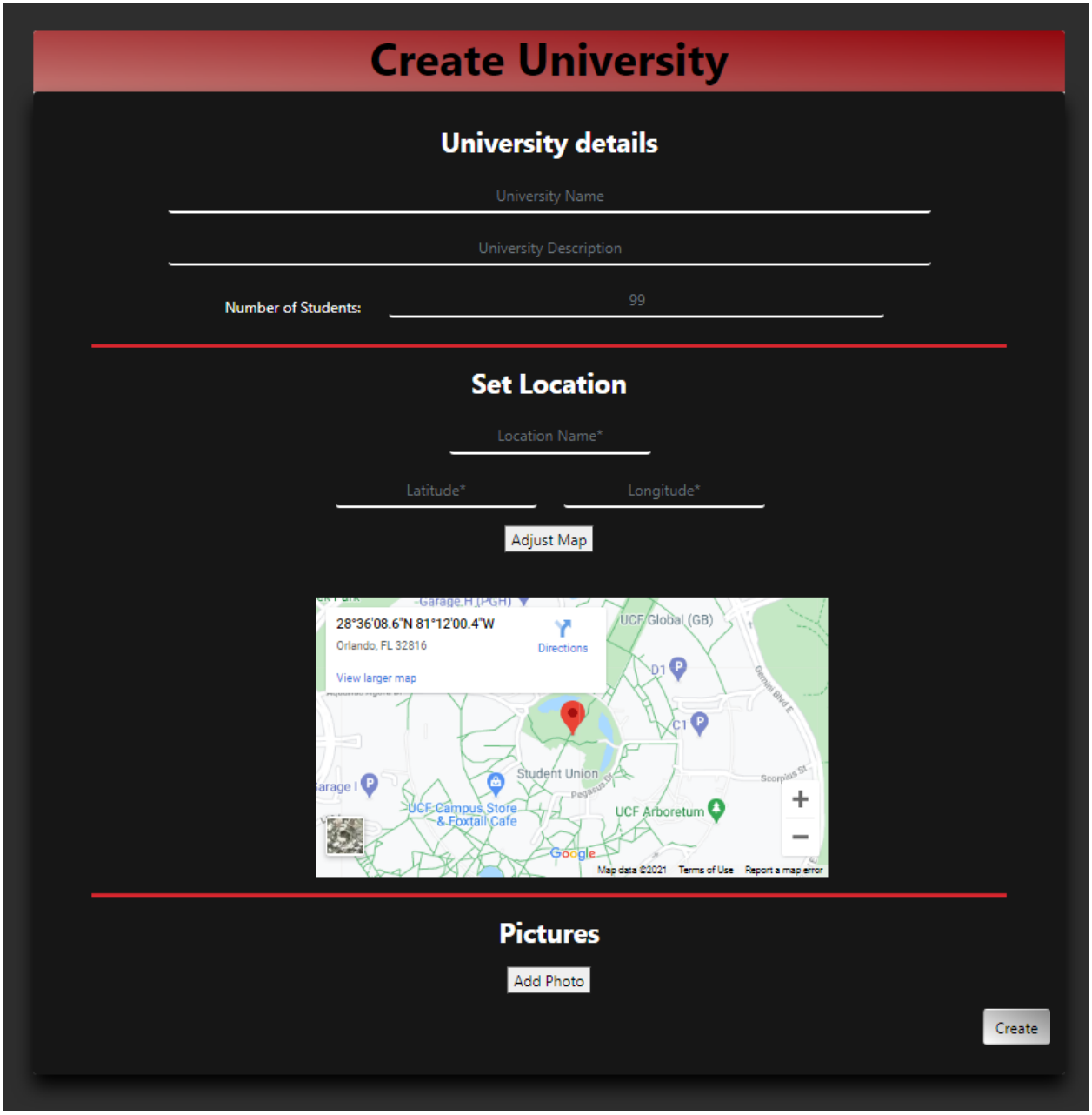


Figure 14: Create University Page

User Profile GUI

On login or registration, the user will be able to see their profile. This will have their basic account details (minus password), as well as any RSOs or universities that the user is a part of. These RSO and universities also have “detail” links so that the user can view or leave them if they want to.

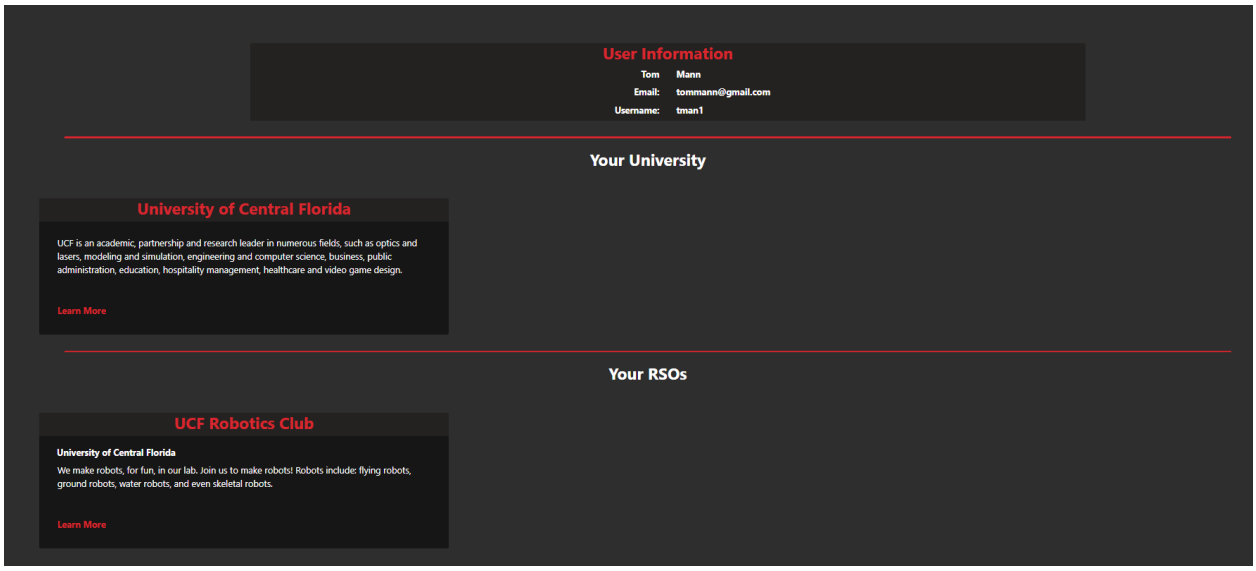


Figure 15: Profile Page

ER-Model

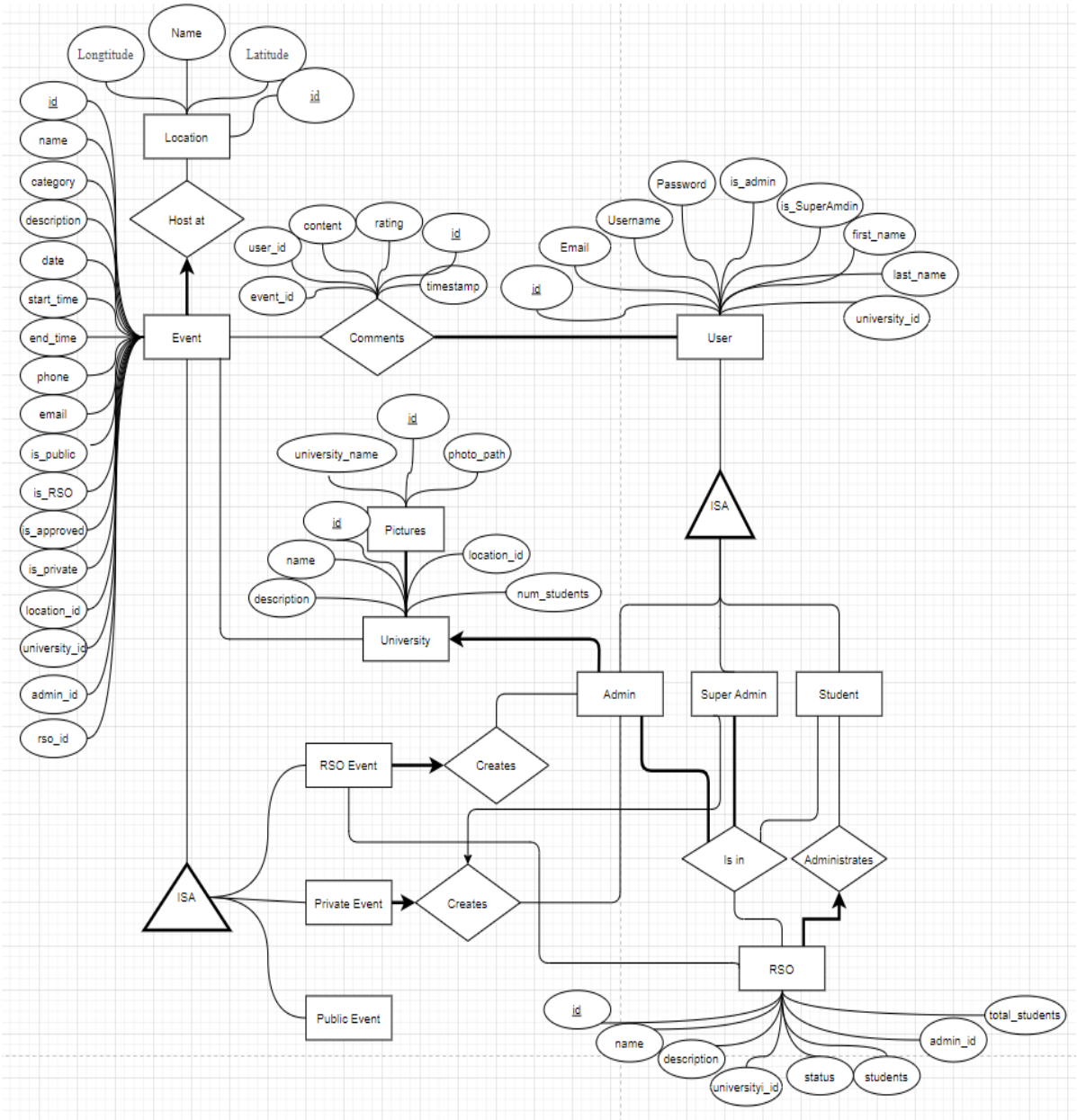


Figure 16: ER Diagram

Relational Data Model

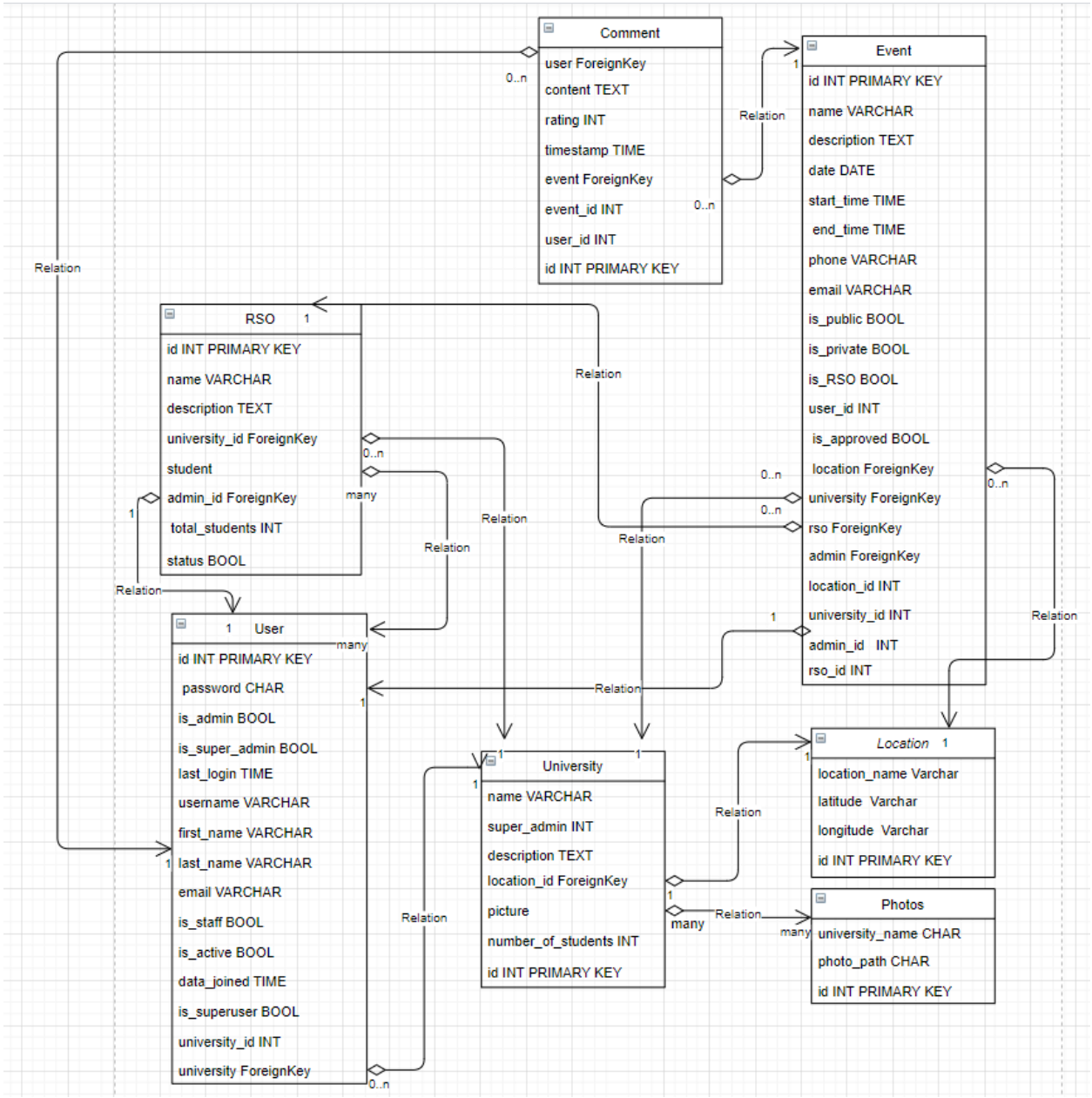


Figure 17: Relational Model Diagram

All tables and triggers were performed using Django and SQLite. Below are normal SQL examples of tables that should be equivalent to the conditions and triggers applied to our project.

Event Tables

```
CREATE TABLE Events_comment(
    id INT PRIMARY KEY NOTNULL,
    content TEXT NOTNULL,
    rating INT NOTNULL,
    event_id INT NOTNULL,
    user_id INT NOTNULL,
    timestamp TIME NOTNULL
);
```

Figure 18: Event Comment Table SQL

```
CREATE TABLE Events_locations(
    id INT PRIMARY KEY NOTNULL,
    location_name VARCHAR(100) NOTNULL,
    latitude VARCHAR(100) NOTNULL,
    longitude VARCHAR(100) NOTNULL

);
```

Figure 19: Event Location Table SQL

```
CREATE TABLE Events_event(
    id INT PRIMARY KEY NOTNULL,
    name VARCHAR(150) NOTNULL,
    description TEXT NOTNULL,
    date DATE NOTNULL,
    start_time TIME NOTNULL,
    end_time TIME NOTNULL,
    phone VARCHAR(31) NOTNULL,
    email VARCHAR(254) NOTNULL,
    is_public BOOL NOTNULL,
    is_private BOOL NOTNULL,
    is_RSO BOOL NOTNULL,
    user_id INT,
    is_approved BOOL NOTNULL,
    location_id INT,
    university_id INT,
    category VARCHAR(150)
    rso_id
);
```

Figure 20: Event Table SQL

Event Triggers

```
CREATE TRIGGER Event_check
    AFTER INNER ON  Events_event
REFERENCING NEW AS NEW Events_event
WHEN ((SELECT COUNT(*)
    FROM Events_event M
    WHERE M.start_time = NewEvent.start_time
    WHERE M.endt_time = NewEvent.end_time
    WHERE M.location_id = NewEvent.location_id
    ))


CREATE TRIGGER Event_admin_checker
    AFTER INNER ON Events_event
REFERENCING NEW AS NEW Events_event
WHEN ((SELECT COUNT(*)
    FROM Users_user M
    WHERE M.rso_id = NewEvent.admin_id.rso_id
    ))
FOR EACH ROW
UPDATE Events_event
```

Figure 21: Event Trigger SQL

RSO Tables

```
CREATE TABLE RSO_rso_students(
    id INTEGER PRIMARY KEY NOTNULL,
    rso_id INTEGER NOTNULL,
    user_id INTEGER NOTNULL
);
```

Figure 22: RSO Student Table SQL

```
CREATE TABLE RSO_rso(
    id INT PRIMARY KEY NOTNULL,
    name VARCHAR(255) NOTNULL,
    description TEXT,
    total_students INT NOTNULL,
    admin_id INT NOTNULL,
    university_id INT NOTNULL,
);
```

Figure 23: RSO Table SQL

RSO Triggers

```
CREATE TRIGGER RSOStatusUpdateA
    AFTER INNER ON RSO_rso_students
REFERENCING NEW AS NEW NewMember
WHEN ((SELECT COUNT(*)
    FROM RSO_rso_students M
    WHERE M.rso_id = NewMember.rso_id) > 4)
FOR EACH ROW
UPDATE RSOs
SET Status = 'active'
WHERE RSO_ID = NewMember.RSO_ID

CREATE TRIGGER RSOStatusUpdateP
    AFTER INNER ON RSO_rso_students
REFERENCING OLD AS ExMember
WHEN ((SELECT COUNT(*)
    FROM RSO_rso_students M
    WHERE M.rso_id = ExMember.rso_id) > 4)
FOR EACH ROW
UPDATE RSOs
SET Status = 'INactive'
WHERE RSO_ID = ExMember.RSO_ID


CREATE TRIGGER RSOStatusUpdateB
    AFTER INNER ON RSO_rso_students
REFERENCING NEW AS NEW NewMember
When((SELECT COUNT(*)
    FROM RSO_rso M
    WHERE M.admin.university_id = NewMember.university_id) > 4)
FOR EACH ROW
UPDATE RSO_rso
SET Status = 'active'
WHERE rso_id = NewMember.rso_id

CREATE TRIGGER RSOStatusUpdateC
    AFTER INNER ON RSO_rso_students
REFERENCING OLD AS ExMember
When((SELECT COUNT(*)
    FROM RSO_rso M
    WHERE M.admin.university_id = ExMember.university_id) > 4)
FOR EACH ROW
UPDATE RSO_rso
SET Status = 'active'
WHERE rso_id = ExMember.rso_id
```

Figure 24: RSO Trigger SQL

University Tables

```
CREATE TABLE Universities_locations(
    id INT PRIMARY KEY NOTNULL,
    location_name VARCHAR(100) NOTNULL,
    latitude VARCHAR(100) NOTNULL,
    longitude VARCHAR(100) NOTNULL
);
```

Figure 25: University Location Table SQL

```
CREATE TABLE Universities_photos(
    id INT PRIMARY KEY NOTNULL,
    university_name VARCHAR(100) NOTNULL,
    photo_path VARCHAR(100) NOTNULL
);
```

Figure 26: University Photo Table SQL

```
CREATE TABLE Universities_pictures(
    id INT PRIMARY KEY NOTNULL,
    university_id INT NOTNULL,
    photo_id INT NOTNULL
);
```

Figure 27: University Picture Table SQL

```
CREATE TABLE Universities_university(
    id INT PRIMARY KEY NOTNULL,
    name VARCHAR(100) NOTNULL,
    description TEXT NOTNULL,
    number_of_students INT NOT NULL,
    location_id INT NOTNULL,
    super_admin
);
```

Figure 29: University Table SQL

University Triggers

```
CREATE TRIGGER UniversityChecker
    AFTER INNER ON Universities_locations
REFERENCING NEW AS NEW Universities_university
WHEN ((SELECT COUNT(*)
    FROM RSO_rso_students M
    WHERE M.rso_id = NewMember.rso_id) > 4)
FOR EACH ROW
UPDATE RSOs
SET Status = 'active'
WHERE RSO_ID = NewMember.RSO_ID
```

Figure 30: University Trigger SQL

User Tables

```
CREATE TABLE Users_rsonumber(
    id INT PRIMARY KEY NOTNULL,
    username VARCHAR(100) NOTNULL,
    rso INT NOT NULL
);
```

Figure 31: Rsonumber Table SQL

```
CREATE TABLE Users_user (
    id INT PRIMARY KEY NOTNULL,
    password VARCHAR(128) NOTNULL,
    last_login DATATIME,
    is_superuser BOOL NOTNULL,
    username VARCHAR(150) NOTNULL,
    first_name VARCHar(150) NOTNULL,
    last_name VARCHAR(150) NOTNULL,
    email VARCHAR(254) NOTNULL,
    is_staff BOOL NOTNULL,
    is_active BOOL NOTNULL,
    data_joined DATETIME,
    is_admin BOOL NOTNULL,
    is_super_admin BOOL NOTNULL,
    university_id INT
);
```

Figure 32: User Table SQL

Sample Data

Below you can view sample data without our SQLite database: 3 universities, 20 events, 10 comments, 5 RSOs, and 20 users.

3 Universities

SQLite

SQL ▾ 1 / 1 1 - 3 of 3

id	name	
1	University of Central Florida	UCF is an academic, partnership and research leader in numerous fields, such as optics and lasers, modeling and simulation, enginee
2	University of Florida	The University of Florida (Florida or UF) is a public land-grant research university in Gainesville, Florida. It is a senior member
3	University of South Florida	A Preeminent Research University located in the heart of Tampa Bay, the University of South Florida is dedicated to empowering stud

Figure 33: All the universities in Database

5 RSOs

SQL ▾ 1 / 1 1 - 5 of 5

id	name	description
2	University of Central Florida IT Group	this is for computer science in UCF
3	UCF Robotics Club	We make robots, for fun, in our lab. Join us to make robots! Robots include: flying robots, ground robots, water robots, a
4	UCF Computer Engineers Club	We love computer hardware and embedded systems.
5	UCF Electrical Engineers Club	Circuits, hardware, cooling, heating, oh my!
6	Python Kindom	Python is the best, C is trash

Figure 34: All the Rsos in Database

20 Events

SQL ▾ 1 / 1 1 - 20 of 20

id	name	
1	Hontoon Island Paddle	Trip Date: 4/10/2021 Trip Location: Hontoon Island and Blue Springs Trip Cost: $10.00 Registration Opens: 3/26/2021 Registration Closes: 4/6/2021 To register for this trip, complete the Adventure Trip Registration Form and cc Pre-Trip Date & Time: 4/7/2021 - 6:00 p.m. Trip Description: Spend a day kayaking over 10 miles through the twists and tur What's Included: All of our adventure trips include all necessary gear for the Transportation is not provided; participants must meet Trip Leaders at the loca
2	UCF ADAGE Gifted Institute	What really goes on "behind the scenes" to support the development of high-abil At this virtual event, we will ask the big questions and examine the impact of We will also be sharing the cumulative results of Project ELEVATE, a collaborat
3	Theatre for Young Audiences: "The Grumpiest Boy in the World"	Wanting to be more than the average seven-year-old boy, grumpy Zachary sets out Seven-year-old Zachary Briddling is normal in every way: normal height, normal Tickets: $10, $5 with UCF IDTickets available now. Learn more and reserve yours UCF Celebrates the Arts, a festival showcasing creativity, innovation and colla
4	Creative Clash	Artists armed with black markers race against the clock to create large-format Tickets: FreeTickets available now. Learn more and reserve yours here: https:// UCF Celebrates the Arts, a festival showcasing creativity, innovation and colla
5	"If Black Language Ain't Artful, Then Tell Me, What Is?" By Dr. April Baker-Bell	Dr. April Baker-Bell, author of "Linguistic Justice: Black Language, Literacy, Tickets: Pay what you can (suggested $10)Learn more: https://arts.cah.ucf.edu/c UCF Celebrates the Arts, a festival showcasing creativity, innovation and colla
6	UCF Symphony Orchestra	The orchestra will feature two works by Black composers, Lyric for Strings by G Tickets: $20, $15 with UCF IDTickets available now. Learn more and reserve your UCF Celebrates the Arts, a festival showcasing creativity, innovation and colla
7	University of Central Florida IT Group Event	this is for computer science in UCF
8	University of Central Florida IT Group Event 2	this is for computer science in UCF
13	Kayaking with Cats	Come destress from finals by kayaking with cats!
14	Python Kick start	python only
15	Python Kick start	this is a public event for those who loves python
16	Python is the best	coding together, shape the future
17	Next Programming Language	discuss which language should we learn after python
18	Ruby vs. Python	Ruby and Python Only, it is War!!!!!1
19	C vs. C++	which one is faster, which one is better
20	Ice Event	eat ice, play ice
21	Minecraft Event	Minecraft is the best, play Minecraft
22	Game 9	play game, play game, play game!!!!!!!!
23	Movie 9	watch Marvel, watch DC, watch both
24	Star War 9	talk about star wars

Figure 35: All the events in Database

10 Comments

SQL ▾ 1 / 1 1 - 10 of 10

id	content	rating	event_id	user_id	timestamp
1	this event is the best	5	1	8	2021-04-15 22:41:46.850756
2	Padding is so much fun!!!	4	1	2	2021-04-15 22:43:47.826649
3	I bet C++ is better	4	19	6	2021-04-15 23:54:18.625308
4	Ruby, Omg, it is 100% better than Python	5	18	6	2021-04-15 23:54:59.539708
5	haha, Ruby is better? Bro, Python is the God	5	18	7	2021-04-15 23:55:51.704688
6	you guys know what, C is the foundation, it is the fastest	4	18	9	2021-04-15 23:57:10.103737
7	Can't wait to join	4	19	9	2021-04-15 23:57:32.148365
8	can I use C? lol	1	14	9	2021-04-15 23:58:20.705785
9	this is the worst event ever	1	3	11	2021-04-15 23:59:09.147240
10	you guys are right, this event is not bad	5	1	11	2021-04-16 00:00:06.184642

Figure 36: All the comments in Database

20 Users

id	last_login	is_superuser	username	first_name	last_name	email	is_staff	is_active
1	2021-04-15 16:46:21.398589	0	tman1	Tom	Mann	tommann@gmail.com	0	1
2	2021-04-16 01:09:35.089309	0	Oliver	Oliver	Ward	Oliver.Ward@gmail.com	0	1
3	2021-04-15 16:59:55.064933	0	Ryan	Ryan	Kelly	Ryan.Kelly@gmail.com	0	1
4	2021-04-15 23:17:30.139888	0	Luke	Luke	Patel	Luke.Patel@gmail.com	0	1
5	2021-04-13 20:42:36.510360	0	Hayden	Hayden	Cole	Hayden.Cole@gmail.com	0	1
6	2021-04-15 23:53:26.979789	0	Frederick	Frederick	Booth	Frederick.Booth@gmail.com	0	1
7	2021-04-15 23:55:14.699045	0	Mason	Mason	Hendrix	Mason.Hendrix@gmail.com	0	1
8	2021-04-15 23:24:28.113987	0	Crew	Crew	Bell	Crew.Bell@gmail.com	0	1
9	2021-04-15 23:56:19.328441	0	Zander	Zander	Gilbert	Zander.Gilbert@gmail.com	0	1
10	2021-04-15 23:21:45.422063	0	Camdyn	Camdyn	Daniel	Camdyn.Daniel@gmail.com	0	1
11	2021-04-15 23:58:42.733236	0	Maurice	Maurice	Haynes	Maurice.Haynes@gmail.com	0	1
12	2021-04-18 01:04:49.229244	0	Zefeng	Zefeng	Yao	Zefeng.Yao@gmail.com	0	1
13	2021-04-15 23:12:20.795267	0	Alexandra	Alexandra	French	Alexandra.French@gmail.com	0	1
14	2021-04-15 23:13:46.918282	0	Jialin	Jialin	Zheng	Jialin.Zheng@gmail.com	0	1
15	2021-04-18 00:18:17.684621	0	TestUser1	TestingUser	TesterSon	testuser@gmail.com	0	1
16	2021-04-18 01:05:41.340501	0	017564	John	Smith	John.smith@Knights.ucf.edu	0	1
17	2021-04-18 00:54:19.271732	0	111234	Joe	Adams	JoeAdams@Knights.ucf.edu	0	1
18	2021-04-18 00:45:26.086042	0	John1	John	Smith	johnSmith@Knights.ucf.edu	0	1
19	2021-04-18 00:46:55.747760	0	155684	Jane	Smith	JaneSmith@Knights.ucf.edu	0	1
20	2021-04-18 00:47:48.074835	0	293020	Tony	Stark	Tony.Stark@Knights.ucf.edu	0	1
21	2021-04-18 00:49:35.500345	0	928554	Steve	Rogers	Steve.Rogers@Knights.ucf.edu	0	1

Figure 37: All the users in Database

SQL Examples and Results

Our inserts and other such database access were performed using SQLite with Django. Below are equivalent SQL examples and their results into our database.

Event Insert

```
INSERT INTO Events_event (
    id,
    name,
    description,
    date,
    start_time,
    end_time,
    phone,
    email,
    is_public,
    is_private,
    is_RSO,
    is_approved,
    location_ID,
    university_id,
    category
    admin_id,
    rso_id
    )
VALUES(
    24,
    "Star War 9",
    "talk about star wars",
    "2025-12-15",
    "10:20:00",
    "12:00:00",
    "1234567890",
    "starwar@gmail.com”,
    1,
    0,
    0,
    1,
    7,
    NULL,
    "movie,game",
    2,
    NULL
)
```

Figure 38: Event Insert SQL Code

id	name
24	Star War 9

description
talk about star wars

date	start_time	end_time	phone	email	is_public	is_private	is_RSO
2025-12-15	10:20:00	12:00:00	1234567890	starwar@gmail.com	1	0	0

is_approved	location_id	university_id	category	admin_id	rso_id
1	7	NULL	movie, game	2	NULL

Figure 39: Event Insert Database Result

Event Comment Insert

```
INSERT INTO Events_comment(
    id,
    content,
    rating,
    event_id,
    user_id,
    timestamp
)
VALUES (
    10,
    "you guys are right, this event is not bad",
    5,
    1,
    11,
    "2021-04-16 00:00:06"
)
```

Figure 40: Comment Insert SQL Code

id	content	rating	event_id	user_id	timestamp
10	you guys are right, this event is not bad	5	1	11	2021-04-16 00:00:06.184642

Figure 41: Comment Insert Database Result

Event Comment Modify

```
SELECT* FROM 'Events_comment c'  WHERE c.event_id = 1

UPDATE Events_comment
    SET content = 'Hello'
    WHERE id = 1
```

Figure 42: Comment Modify SQL Code

id	content	rating	event_id	user_id	timestamp
1	this event is the best	5	1	8	2021-04-15 22:41:46.850756

Figure 43: Comment Modify Database Before Result

id	content	rating	event_id	user_id	timestamp
2	Padding is so much fun!!!	4	1	2	2021-04-15 22:43:47.826649
17	Hello	5	1	8	2021-04-18 15:39:34.627128

Figure 44: Comment Modify Database After Result - Comment Modified to Id 17

Event Comment Delete

```
SELECT* FROM 'Events_comment c'  WHERE c.event_id = 1

DELETE FROM Events_comment
    WHERE id = 1
```

Figure 45: Comment Delete SQL Code

id	content	rating	event_id	user_id	timestamp
1	this event is the best	5	1	8	2021-04-15 22:41:46.850756

Figure 46: Comment Delete Database Before Result

id	content	rating	event_id	user_id	timestamp
2	Padding is so much fun!!!	4	1	2	2021-04-15 22:43:47.826649
3	I bet C++ is better	4	19	6	2021-04-15 23:54:18.625308
4	Ruby, Omg, it is 100% better than Python	5	18	6	2021-04-15 23:54:59.539708
5	haha, Ruby is better? Bro, Python is the God	5	18	7	2021-04-15 23:55:51.704688

Figure 47: Comment Delete Database After Result - No More Id 1

Event Location Insert

```
INSERT INTO Events_locations(
    id,
    location_name,
    latitude,
    longitude
)
VALUES (
    9,
    "Tampa",
    "28.0587",
    "-82.4139"
)
```

Figure 48: Event Location Insert SQL Code

id	location_name	latitude	longitude
9	Tampa	28.0587	-82.4139

Figure 49: Event Location Insert Database Result

RSO Insert

```
INSERT INTO RSO_rso (
    id,
    name,
    description,
    total_students,
    admin_id,
    status,
    university_id
)
VALUES(
    6,
    "Python Kindom",
    "Python is the best, C is trash",
    5,
    8,
    1,
    1
)
```

Figure 50: RSO Insert SQL Code

id	name	description
6	Python Kindom	Python is the best, C is trash

total_students	admin_id	status	university_id
5	8	1	1

Figure 51: RSO Insert Database Result

RSO Student Insert

```
INSERT INTO RSO_rso_students(
    id,
    rso_id,
    user_id
)
VALUES(
    1,
    1,
    3
)
```

Figure 52: Rso Student Insert SQL example

id	rso_id	user_id
1	2	3

Figure 53: RSO Student Insert Database Result

University Insert

```
INSERT INTO Universities_university(
    id,
    name,
    description,
    number_of_students,
    location_id,
    super_admin
)
VALUES (
    1,
    "University of Central Florida",
    "UCF is an academic, partnership and research leader in numerous
    fields, such as optics and lasers, modeling and simulation, engineering
    and computer science, business, public administration, education,
    hospitality management, healthcare and video game design.",
    66183,
    1,
    12
)
```

Figure 54: University Insert SQL Code

id	name	
1	University of Central Florida	UCF is an academic, partnership and research leader in numerous

	number_of_students	location_id	super_admin
	66183	1	12

Figure 55: University Insert Database Result

University Location Insert

```
INSERT INTO Universities_locations(
    id,
    location_name,
    latitude,
    longitude
)
VALUES(
    1,
    Orlando,
    "28.6024",
    "-81.2001"
)
```

Figure 56: University Location Insert SQL Code

id	location_name	latitude	longitude
1	Orlando	28.6024	-81.2001

Figure 57: University Location Insert Database Result

University Photo Insert

```
INSERT INTO Universities_photos(
    id,
    university_name,
    photo_path
)
VALUES(
    1,
    "University of Central Florida",
    "University-Central-Florida-940x338.jpg"
)
```

Figure 58: University Photo Insert SQL Code

id	university_name	photo_path
1	University of Central Florida	UCF_HCEC.jpg
2	University of Central Florida	University-Central-Florida-940x338.jpg

Figure 59: University Photo Insert Database Result

University Picture Insert

```sql
INSERT INTO Universities_pictures(
    id,
    university_id,
    photo_id
)
VALUES(
    1,
    1,
    1
)
```

Figure 60: University Picture Insert SQL Code

SQL ▾

id	university_id	photos_id
1	1	1
2	1	2
3	2	3
4	2	4
5	3	5
6	3	6

Figure 61: University Picture Insert Database Result

User Rsonumber Insert

```
INSERT INTO Users_rsonumber(
    id,
    username,
    rso
)
VALUES(
    1,
    "Oliver",
    2
)
```

Figure 62: Rsonumber Insert SQL Code

id	username	rso
1	Oliver	2

Figure 63: Rsonumber Insert Database Result

User Insert

```
INSERT INTO Users_user (
    id,
    last_login,
    is_superuser,
    username,
    first_name,
    last_name,
    email,
    is_staff,
    is_active,
    date_joined,
    is_admin,
    is_super_admin,
    university_id,
    password
)
VALUES(
    1,
    "2021-04-15 16:46:21.398589",
    0,
    "tman1",
    "Tom",
    "Mann",
    "tommann@gmail.com",
    0,
    1,
    "2021-04-10 17:18:50.190694,
    1,
    0,
    1,
    "gAAAAABgcd36t4EBlJJ7jZ6BA9lpJVrBWAe4jHNgDVlIqui85oB_
    7JBwLvT4flYfC6Rqp9jDSmDpepurZSv-fvzaa5ZuYGlfoQ=="
)
```

Figure 64: User Insert SQL Code

id	last_login	is_superuser	username	first_name	last_name	email
1	2021-04-15 16:46:21.398589	0	tman1	Tom	Mann	tommann@gmail.com

is_staff	is_active	date_joined	is_admin	is_super_admin	university_id
0	1	2021-04-10 17:18:50.190694	1	0	1

password
gAAAAABgcd36t4EBlJJ7jZ6BA9lpJVrBWAe4jHNgDVlIqui85oB_7JBwLvT4flYfC6Rqp9jDSmDpepurZSv-fvzaa5ZuYGlfoQ==

Figure 65: User Insert Database Result

Display Events Queries

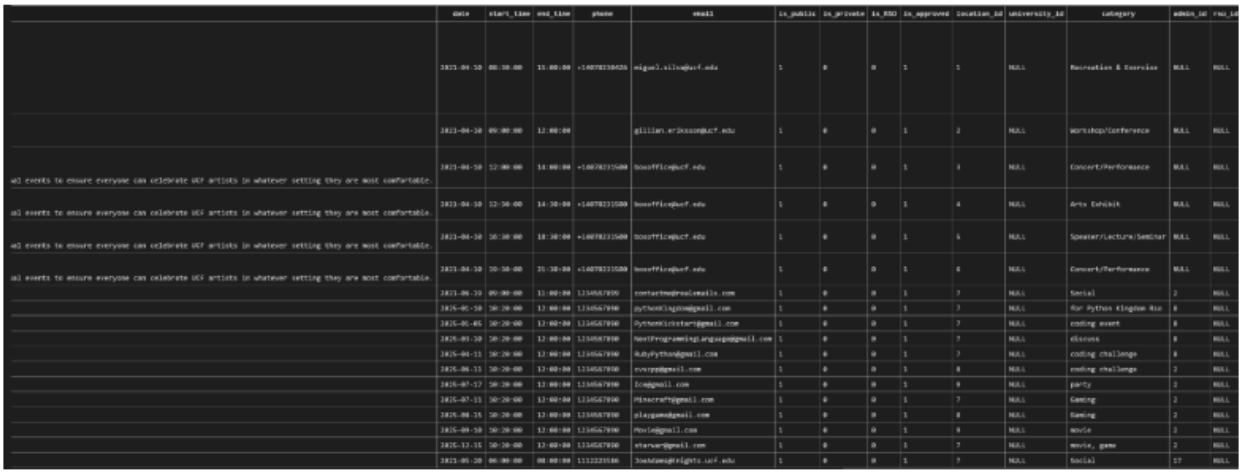
```
SELECT DISTINCT*FROM 'Events_event' e
    WHERE e.is_public = 1


SELECT DISTINCT*FROM 'Events_event' e, 'Users_user' m
    WHERE e.is_private = 1 And m.university_id = e.university_id


SELECT DISTINCT*FROM 'Events_event' e, 'Users_rsonumber' u 'Users_user' m
    WHERE e.is_RSO = 1 And u.rso_id = e.rso_id AND m.username = u.username
```

Figure 66: View Events: Public, Private, and RSO SQL Query Code

id	name	
1	Hontoon Island Paddle	Trip Date: 4/10/2021 Trip Location: Hontoon Island and Blue Springs Trip Cost: $10.00 Registration Opens: 3/26/2021 Registration Closes: 4/6/2021 To register for this trip, complete the Adventure Trip Registration Form and contact the Outdoor Adventure Center during open hours. Pre-Trip Date & Time: 4/7/2021 - 6:00 p.m. Trip Description: Spend a day kayaking over 10 miles through the twists and turns of the St. Johns River and surrounding creeks. We will start at Hontoon Island State Park and stop half way at Blue Spring St What's Included: All of our adventure trips include all necessary gear for the activity and professional guiding by UA Trip Leaders. Participants should wear athletic clothing, appropriate footwear and bring Transportation is not provided; participants must meet Trip Leaders at the location. Participants must wear a face covering.
2	UCF ADAGE Gifted Institute	What really goes on "behind the scenes" to support the development of high-ability and gifted individuals? Learn about the vision, efforts, activities and advocacy of top leaders and associations at the inte At this virtual event, we will ask the big questions and examine the impact of the pandemic on evolving trends and practices, the future of gifted education in an era of social justice and innovation, and ho we will also be sharing the cumulative results of Project ELEVATE, a collaboration with 12 Title I schools to bridge the equity gap, and provide access and a relevant curriculum for underserved, diverse lear
3	Theatre for Young Audiences: "The Grumpiest Boy In the World"	Wanting to be more than the average seven-year-old boy, grumpy Zachary sets out to find a land of giants, flying things, and anything else that will help him stand out from the average kid. Seven-year-old Zachary Grimbling is normal in every way: normal height, normal hair, and normal dreams. And this makes him so grumpy. Trying to find a way to stand out, The Grumpiest Boy In the World follows Tickets: $10, $5 with UCF IDTickets available now. Learn more and reserve yours here: https://arts.cah.ucf.edu/celebrates/ UCF Celebrates the Arts, a festival showcasing creativity, innovation and collaboration from Orlando's hometown university, occurs annually at Dr. Phillips Center for the Performing Arts in April. The event
4	Creative Clash	Artists armed with black markers race against the clock to create large-format artwork. The theme will be announced seconds before the start of the event, so come cheer on the artists and help choose a winn Tickets: Free!Tickets available now. Learn more and reserve yours here: https://arts.cah.ucf.edu/celebrates/ UCF Celebrates the Arts, a festival showcasing creativity, innovation and collaboration from Orlando's hometown university, occurs annually at Dr. Phillips Center for the Performing Arts in April. The event
5	"If Black Language Ain't Artful, Then Tell Me, What Is?" By Dr. April Baker-Bell	Dr. April Baker-Bell, author of "Linguistic Justice: Black Language, Literacy, Identity, and Pedagogy," will tell a story about the history, art and resiliency of Black Language — its relationship to enslave Tickets: Pay what you can (suggested $10)Learn more: https://arts.cah.ucf.edu/celebrates/ UCF Celebrates the Arts, a festival showcasing creativity, innovation and collaboration from Orlando's hometown university, occurs annually at Dr. Phillips Center for the Performing Arts in April. The event
6	UCF Symphony Orchestra	The orchestra will feature two works by Black composers, Lyric for Strings by George Walker and Mother and Child by William Grant Still, as well as the Chamber Orchestra version of Shostakovich's String Quar Tickets: $30, $15 with UCF IDTickets available now. Learn more and reserve yours here: https://arts.cah.ucf.edu/celebrates/ UCF Celebrates the Arts, a festival showcasing creativity, innovation and collaboration from Orlando's hometown university, occurs annually at Dr. Phillips Center for the Performing Arts in April. The event
13	Kayaking with Cats	Come destress from finals by kayaking with cats!
14	Python Kick start	python only
15	Python Kick start	this is a public event for those who loves python
17	Next Programming Language	discuss which language should we learn after python
18	Ruby vs. Python	Ruby and Python Only, it is War!!!!!!
19	C vs. C++	which one is faster, which one is better
20	Ice Event	eat ice, play ice
21	Minecraft Event	Minecraft is the best, play Minecraft
22	Game 9	play game, play game, play game!!!!!!!!
23	Movie 9	watch Marvel, watch DC, watch both
24	Star War 9	talk about star wars
25	Bird Watching	watching birds early in the morning to relax for final exam week.



	date	start_time	end_time	phone	email	is_public	is_private	is_RSO	is_approved	location_id	university_id	category	admin_id	rso_id
	2021-04-10	08:30:00	15:00:00	+14078230426	miguel.silva@ucf.edu	1	0	0	1	1	NULL	Recreation & Exercise	NULL	NULL
	2021-04-10	09:00:00	12:00:00		gillian.erickson@ucf.edu	1	0	0	1	2	NULL	Workshop/Conference	NULL	NULL
al events to ensure everyone can celebrate UCF artists in whatever setting they are most comfortable.	2021-04-10	12:00:00	14:00:00	+14078231500	boxoffice@ucf.edu	1	0	0	1	3	NULL	Concert/Performance	NULL	NULL
al events to ensure everyone can celebrate UCF artists in whatever setting they are most comfortable.	2021-04-10	12:30:00	14:30:00	+14078231500	boxoffice@ucf.edu	1	0	0	1	4	NULL	Arts Exhibit	NULL	NULL
al events to ensure everyone can celebrate UCF artists in whatever setting they are most comfortable.	2021-04-10	16:30:00	18:30:00	+14078231500	boxoffice@ucf.edu	1	0	0	1	5	NULL	Speaker/Lecture/Seminar	NULL	NULL
al events to ensure everyone can celebrate UCF artists in whatever setting they are most comfortable.	2021-04-10	19:30:00	21:30:00	+14078231500	boxoffice@ucf.edu	1	0	0	1	6	NULL	Concert/Performance	NULL	NULL
	2021-04-10	09:00:00	11:00:00	1234567890	contactme@realemails.com	1	0	0	1	7	NULL	Social	2	NULL
	2025-01-10	10:20:00	12:00:00	1234567890	pythonKingdom@gmail.com	1	0	0	1	7	NULL	for Python Kingdom Ric	8	NULL
	2025-01-05	10:20:00	12:00:00	1234567890	PythonKickstart@gmail.com	1	0	0	1	7	NULL	coding event	8	NULL
	2025-01-10	10:20:00	12:00:00	1234567890	NextProgramingLanguage@gmail.com	1	0	0	1	7	NULL	discuss	8	NULL
	2025-04-11	10:20:00	12:00:00	1234567890	RubyPython@gmail.com	1	0	0	1	7	NULL	coding challenge	8	NULL
	2025-06-11	10:20:00	12:00:00	1234567890	cscpp@gmail.com	1	0	0	1	8	NULL	coding challenge	2	NULL
	2025-07-17	10:20:00	12:00:00	1234567890	lcm@gmail.com	1	0	0	1	8	NULL	party	2	NULL
	2025-07-11	10:20:00	12:00:00	1234567890	Minecraft@gmail.com	1	0	0	1	7	NULL	Gaming	2	NULL
	2025-08-16	10:20:00	12:00:00	1234567890	playgame@gmail.com	1	0	0	1	8	NULL	Gaming	2	NULL
	2025-09-10	10:20:00	12:00:00	1234567890	Movie@gmail.com	1	0	0	1	8	NULL	movie	2	NULL
	2025-12-15	10:20:00	12:00:00	1234567890	starwar@gmail.com	1	0	0	1	7	NULL	movie, game	2	NULL
	2021-05-20	06:00:00	08:00:00	1112223336	bookdawg@rights.ucf.edu	1	0	0	1	7	NULL	Social	17	NULL

Figure 67: View Events - Public Results

id	name	description	date	start_time	end_time	phone	email	is_public	is_private
16	Python is the best	coding together, shape the future	2025-02-10	10:20:00	12:00:00	1234567890	Pythonisthebest@gmail.com	0	1
26	UCF Horse Admiring	Academically admiring different horses.	2021-11-26	08:00:00	12:00:00	+19995551111	JoeAdams@Knights.ucf.edu	0	1

email	is_public	is_private	is_RSO	is_approved	location_id	university_id	category	admin_id	rso_id
Pythonisthebest@gmail.com	0	1	1	1	7	1	coding bus	8	6
JoeAdams@Knights.ucf.edu	0	1	0	1	10	1	Academic	17	NULL

Figure 68: View Events - Private Results

id	name	description	date	start_time	end_time	phone
8	University of Central Florida IT Group Event 2	this is for computer science in UCF	2027-05-14	10:20:00	12:00:00	+14078235117
16	Python is the best	coding together, shape the future	2025-02-10	10:20:00	12:00:00	1234567890
27	Cat Cafe Enjoyment	Make a mock cat cafe at UCF and enjoy it.	2021-06-27	07:00:00	12:00:00	1112223333

is_private	is_RSO	is_approved	location_id	university_id	category	admin_id	rso_id
0	1	1	7	NULL	This is for University of Central Florida IT Group	2	2
1	1	1	7	1	coding bus	8	6
0	1	1	7	NULL	Business	17	10

Figure 69: View Events - RSO Results

Constraints and Enforcement

Upon adding an event, it is checked that the event is not at the same location and time with an existing event. If it is, an error shows notes about the conflicting event, time, and location.

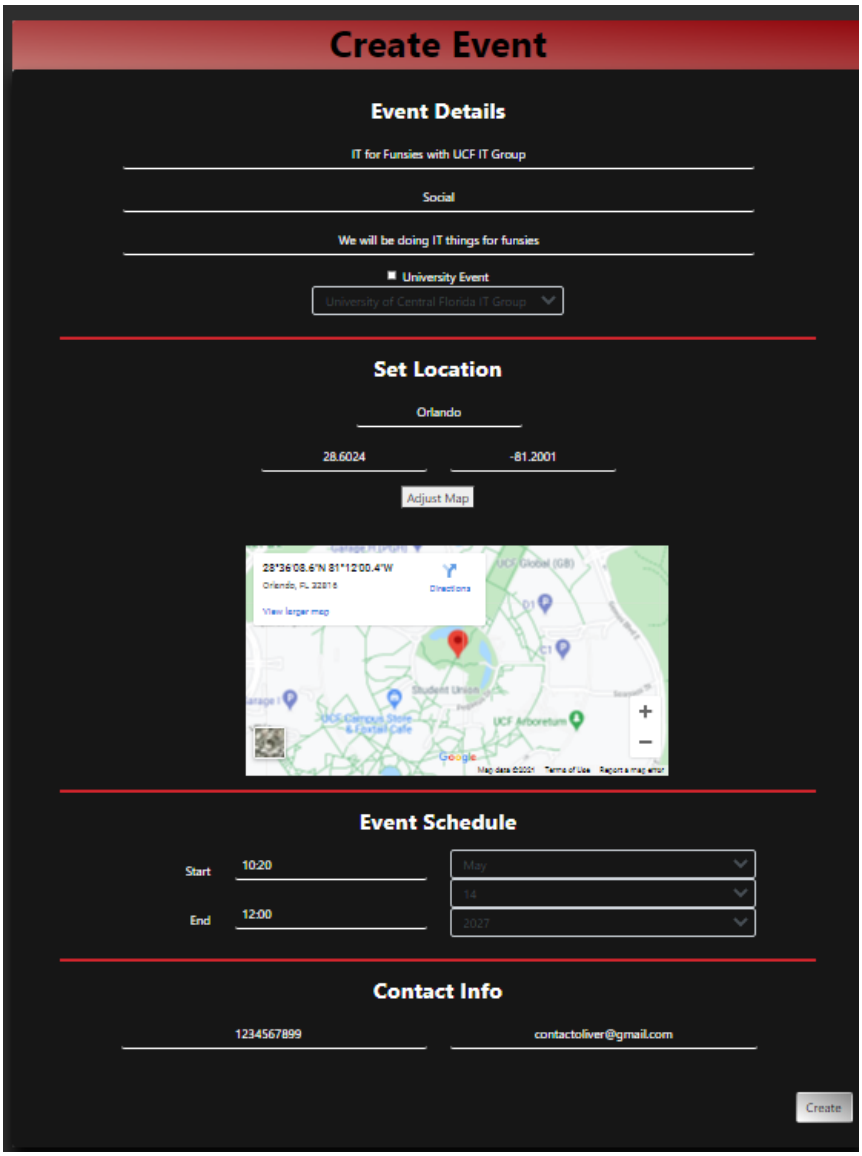


Figure 70: Attempting to Add an Event at the Same Time/Location as Another

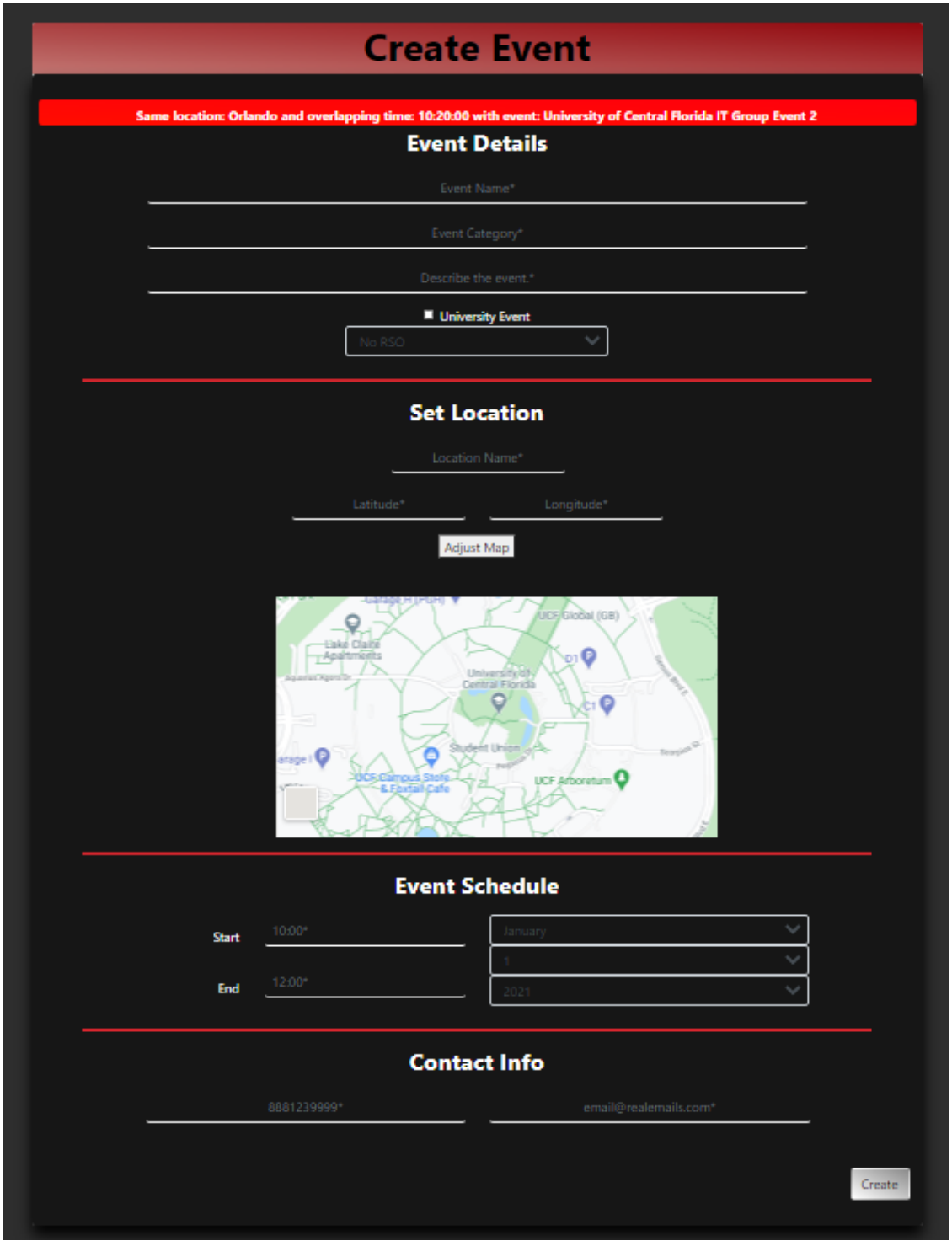


Figure 71: Resultant Error of Trying to Add an Event at the Same Time/Location as Another

Adding an event for an RSO that the Admin is not the Admin of will result in a constraint error. This is because they do not control that RSO. The error will simply show that it is not their RSO.

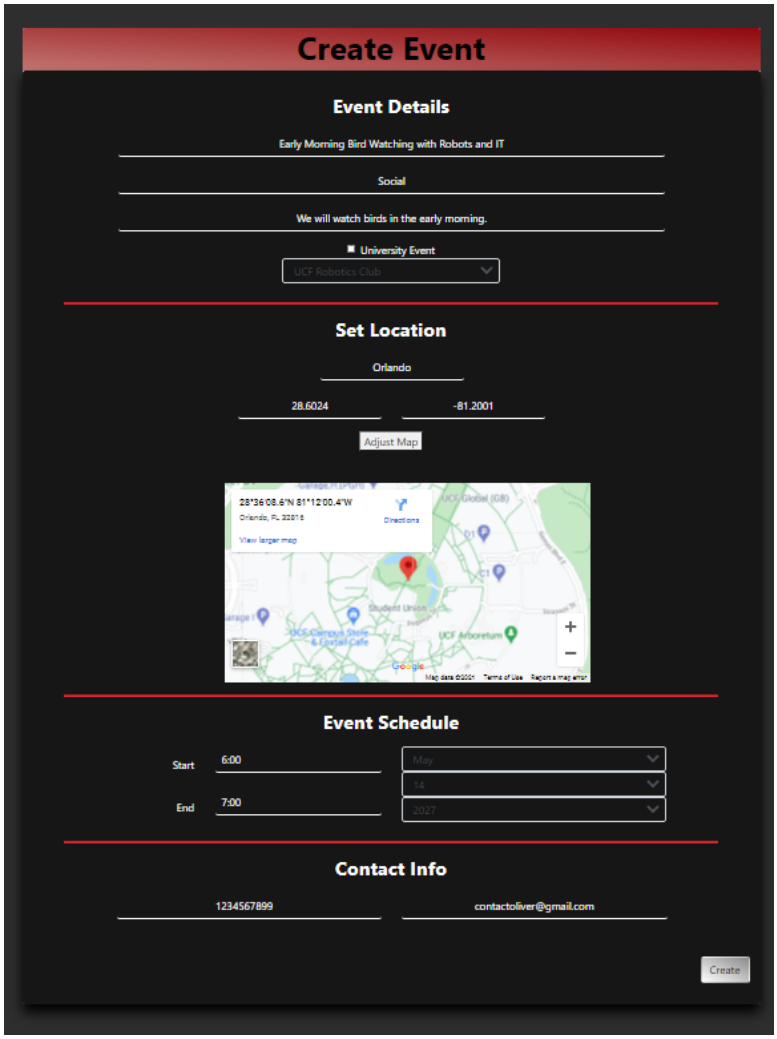


Figure 72: Attempting to Add an Event for an RSO User is Not Admin for

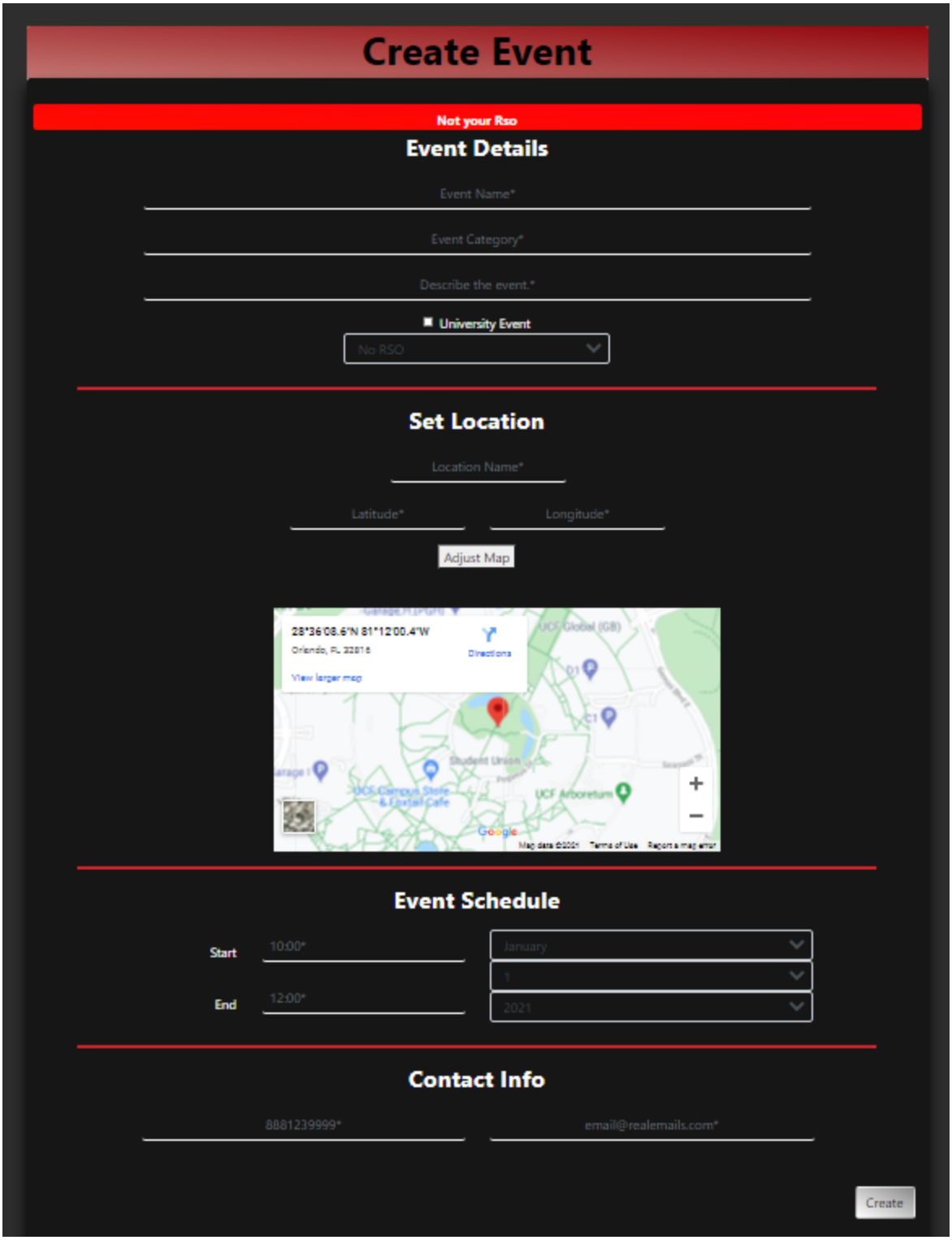


Figure 73: Resulting Error of Attempting to Add an Event for an RSO User is Not Admin for

When a member is added to an inactive RSO with 4 members, that RSO will become an active RSO. Similarly, when a member is removed from an active RSO with 5 members, that RSO will become an inactive RSO. This is because there must be 5 members total for the RSO to be considered active.

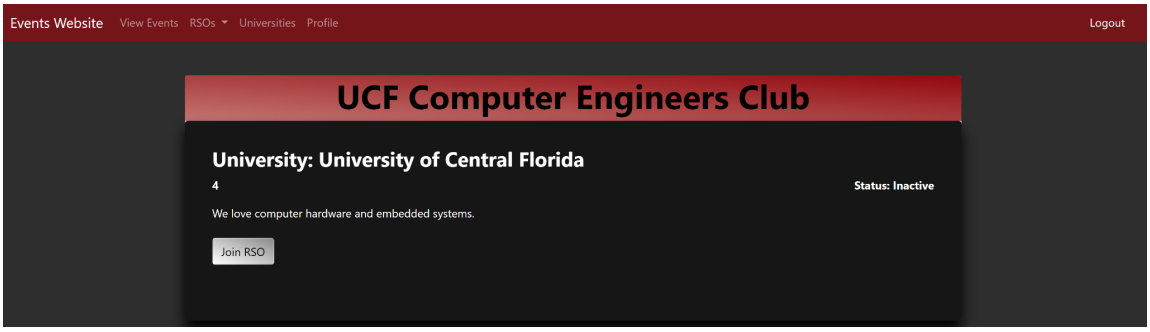


Figure 74: RSO with Only 4 Members (Has Inactive Status)

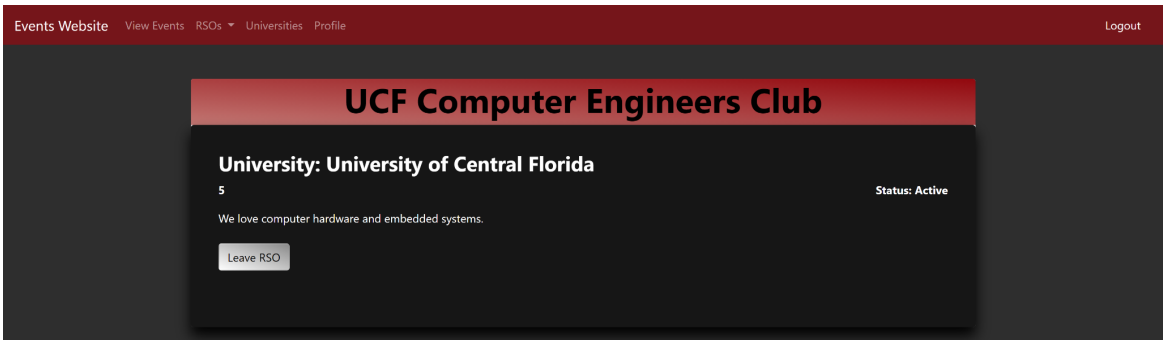


Figure 75: RSO after New Member Joins Now has Active Status

Figure 76: RSO with Only 5 Members (Has Active Status)

Figure 77: RSO After Current Member Leaves Now has Inactive Status

Demo

The demonstration is included in the same CEW46.zip folder. It includes a demonstration of the college event website application running through some common tasks. These include creating some users, joining universities with those users, creating an RSO as one of the users, creating some public, private, and RSO events, and adding comments to various events. It is titled CEW46DemoVideo.mp4.

Conclusion/Observations

The website runs smoothly and achieves all desired goals of event management. There is the ability to view events, universities, RSOs and the additional functionality of a user profile. These achieve the base functionality of an event website - the ability to have events for all the agents needed. Searches have been added for all of those mentioned, which allows for an easier user experience for finding things on the website. Creation of events, RSOs, and universities is all achieved according to level of scope, allowing for a self-sustaining system for students, RSO admins, and faculty without needed developer intervention.. Deletion of RSOs and universities is allowed for the admins/creators of such, allowing for more user agency regarding their own creations Finally, the automation of RSO inactivity and event approval allows for the website to run smoothly for the student view. Overall, all of the features expected of a website of this caliber are present.

For security purposes, encryption was used. Login and registration passwords are secured through encryption - a symmetric (secret key) method was used for both its security and convenience. Passwords were encrypted before going into the database on registration, then decrypted from the database for login purposes. This achieved a well working security system regarding user passwords. In the future, this could potentially be improved through layering or using more complicated encryption algorithms, as well as requiring the user to have passwords of certain lengths and character types. Overall password security was sufficient but given more time could be improved.

Social network integration was added for the various events so that users could share events to Facebook and Twitter. This integration works well for sharing events, but in the future could be expanded to also add a feed from the universities' social media accounts. For example, RSO detail pages could allow for a Twitter or Facebook feed for that RSO's page. Similar things could be done for university detail pages. Overall, the social media component achieved sufficient integration for events but could offer more personalized components to universities and RSOs.

University event feeds from the University of Central Florida (UCF) were used to help populate the database and provide real events for users to view. This gave, generally, about ~6 events at a given time, as UCF only releases feeds for a given time interval through its json feed. Given more time, feeds from other universities could be added, or an optional form for universities to input their json feed from their university when they make the university. This would give more agency to the universities and allow for feeds without needing developer intervention. Overall, feeds for UCF were successful.

The database query response time is fast. The database that we used is SQLite, it is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. Therefore, we don't have to use firebase.

Throughout the project, we applied database knowledge from our class to the project such as attribute, domain, entity, relationship,entity set, relationship set, one-to-many relationship, many-to-many relationship, participation constraint, overlap constraint, covering constraint, weak entity set, aggregation, and role indicator. We also learned how to code SQL, which helped us to understand data structure and extract data information from huge data sets. The solid foundation of database knowledge led us to better organize and create our own ER diagram and relational data diagram to be more readable and understandable for both ourselves and the people who view our project.

During the planning section of this project, we chose ASP.NET Core as our backend framework at first, because it supports a useful and famous design pattern "Model–View–Controller (usually known as MVC)". But we ended up using Django, because we couldn't connect .NET Core with our database for some reason and besides that we believed that a lot of companies want their employees to have a python background before applying to the job, and we only have little experience with python. Therefore, we chose Django as our backend framework. It is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. And by using the Django framework, we learned a lot of the python knowledge that's never been taught before, like response and request parameters. We also learned how to use python on html files, which makes the frontend easier to code. And the last thing we learned is the relationship between frontend, backend, and database.