College Event Website Group Report

COP 4710 - Spring 2025 - Section 001

Group 7: Andy Van

Table of Contents

Project Description

This is a full stack LAMP project that implements the guidelines set out for the College Event Website project for COP 4710. There are three different levels of access, User (Student), Admin, and SuperAdmin, that change the view of the application and enable/disable features based on this role. Each University can have Registered Student Organizations (RSOs) and users can join these. Events can be created that have details such as name, time, location, etc. There are three different types of events implemented: RSO, Private, Public. RSO Events can only be seen by those in the respective RSO, Private events can only be seen by those in the same University, and Public events can be seen by every user. There is also functionality to add/edit/delete comments that are associated with each event. This report will go into detail about the implementation of it and certain highlights.

I will clarify some assumptions I've made to have a working implementation of the project. Based on the given base ER Diagram and the technical requirements set by the demo, it seems that there is no request or approval system between the different levels for creation of events or RSOs. For the superuser, there is no direct way to register as one from the web application; I assume we do this in the back and assign it to a University entity. However, there is functionality to login as the superuser and be able to edit the associated University profile.

GUI

Platform

I am using a Docker container and hosting the technology stack locally on a MacBook Air M1.

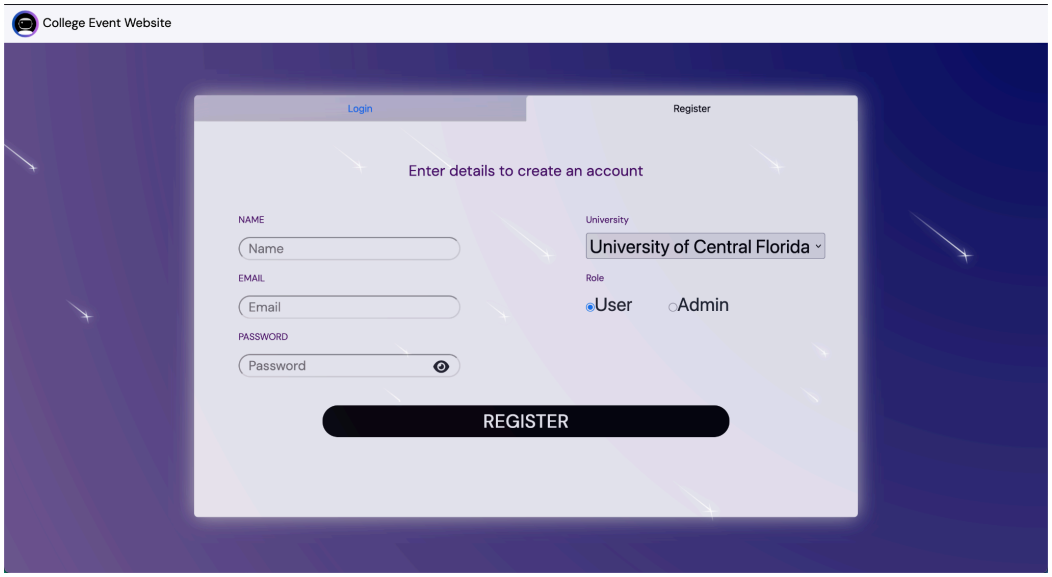
Languages

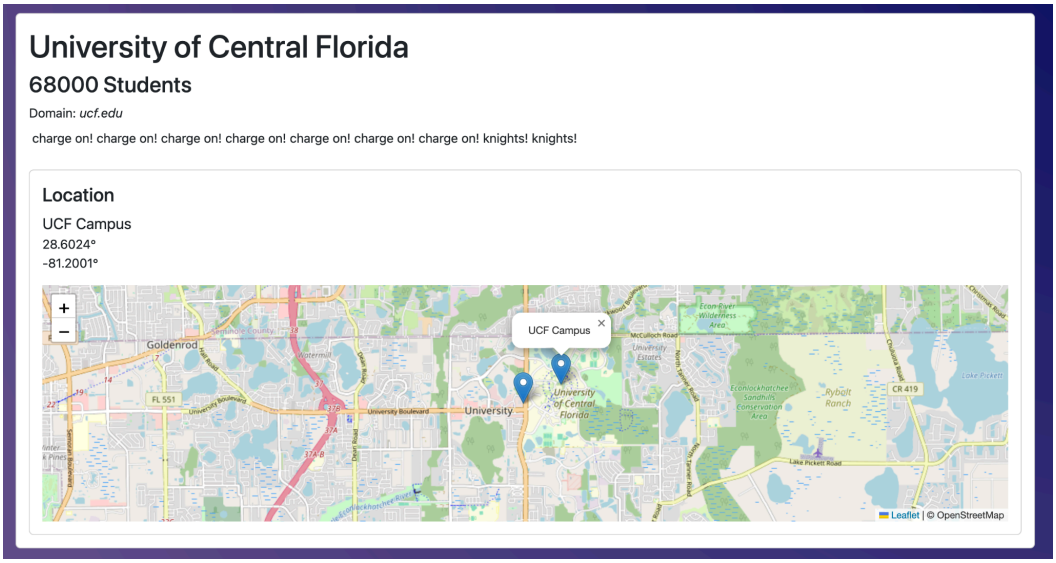
The stack used is a LAMP stack which includes Linux, Apache Web Server, MySQL, and PHP. Apache Web Server handles web requests and shows web content for the frontend. MySQL is relational database management. I am using PHP to make SQL queries to the database for the API endpoints. The frontend consists of HTML, CSS, and Javascript languages, and is also using the Bootstrap framework.

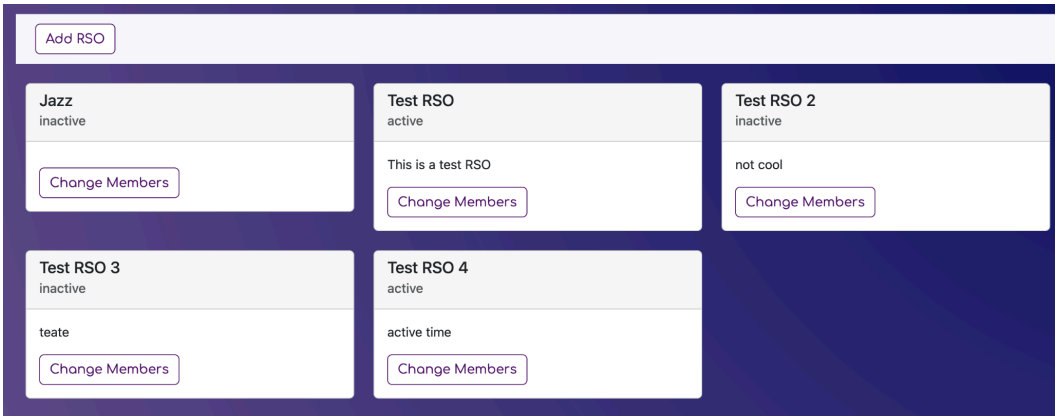
DBMS

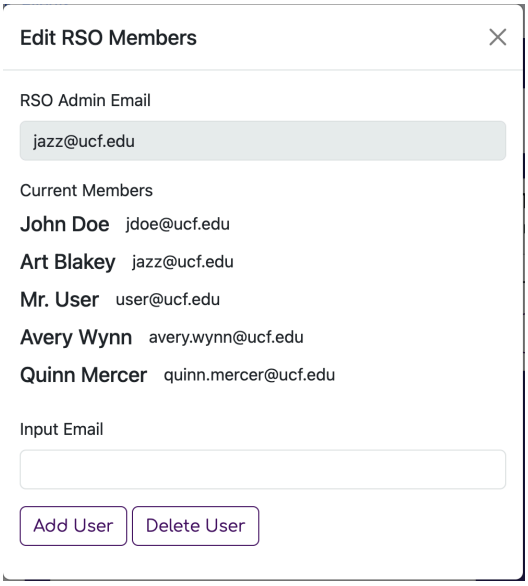
MySQL is the database management system used for this project. The GUI tool, phpMyAdmin, allows for managing the MySQL database. This was very useful for implementing the schema and creating the initial data.

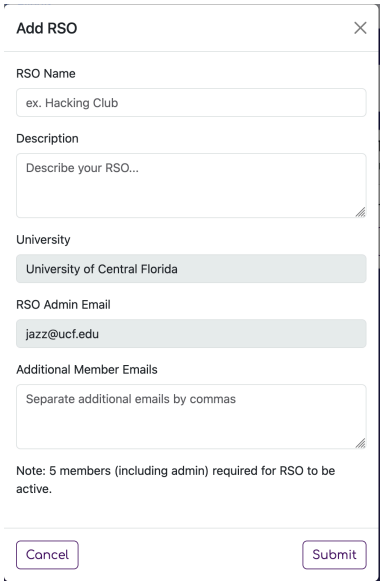
Screenshots

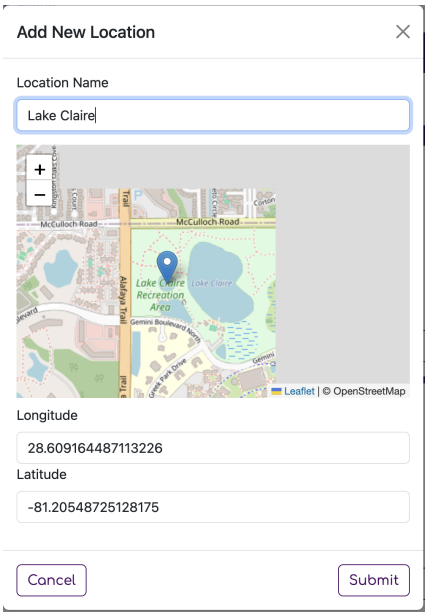


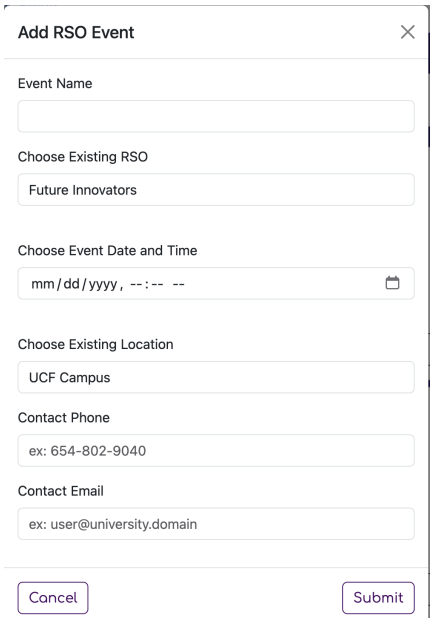


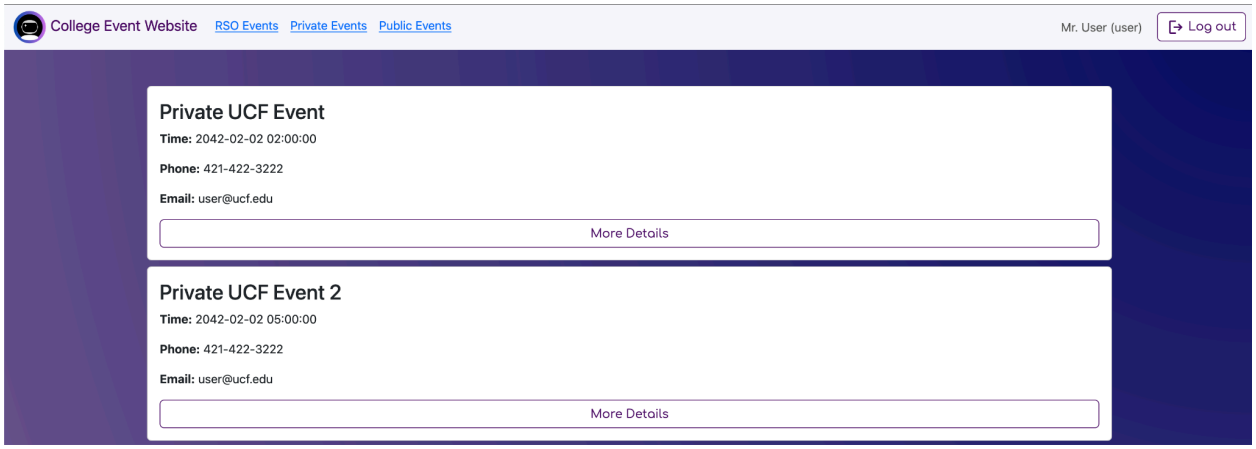

Add RSO
Jazz
inactive
Change Members
Test RSO
active
This is a test RSO
Change Members
Test RSO 2
inactive
not cool
Change Members
Test RSO 3
inactive
teate
Change Members
Test RSO 4
active
active time
Change Members

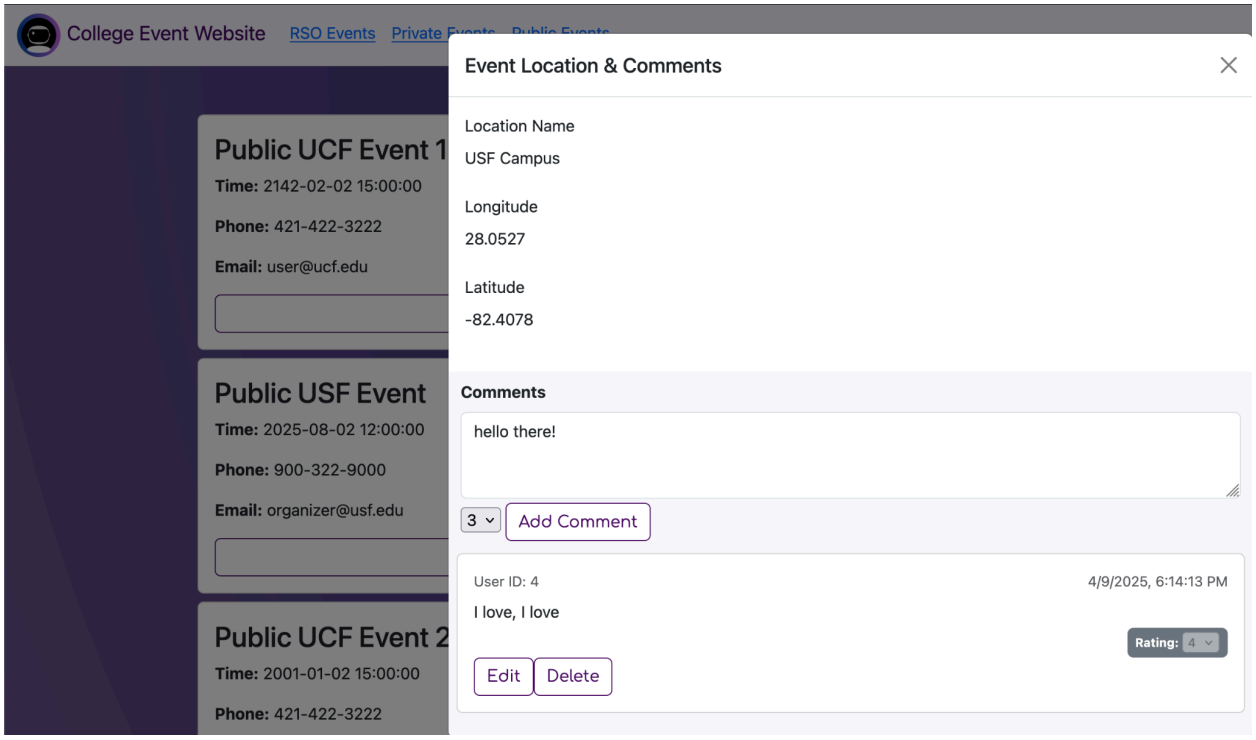

Edit RSO Members
RSO Admin Email
jazz@ucf.edu
Current Members
John Doe jdoe@ucf.edu
Art Blakey jazz@ucf.edu
Mr. User user@ucf.edu
Avery Wynn avery.wynn@ucf.edu
Quinn Mercer quinn.mercer@ucf.edu
Input Email
Add User
Delete User

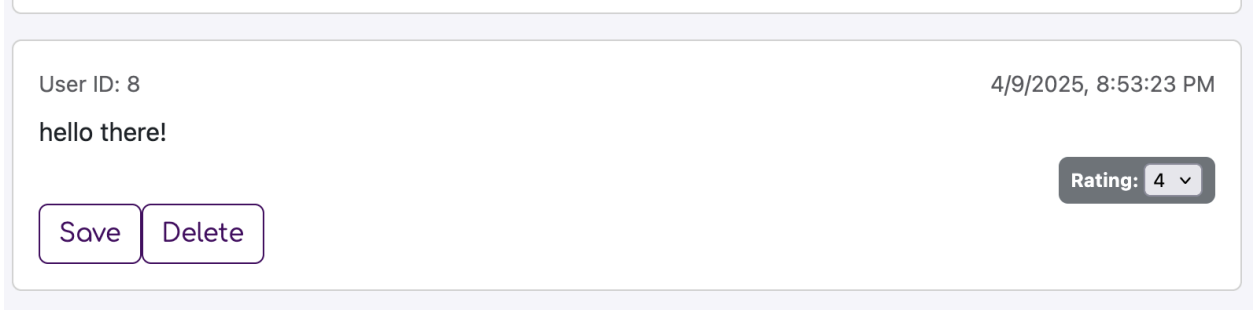

Add RSO
RSO Name
ex. Hacking Club
Description
Describe your RSO...
University
University of Central Florida
RSO Admin Email
jazz@ucf.edu
Additional Member Emails
Separate additional emails by commas
Note: 5 members (including admin) required for RSO to be active.
Cancel
Submit

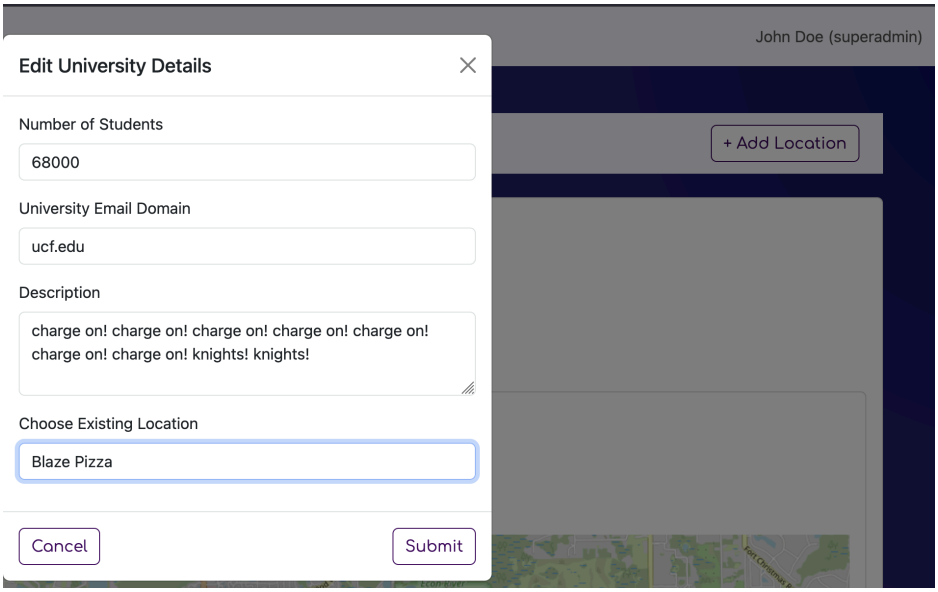

Add New Location
Location Name
Lake Claire
McCulloch Road
Lake Claire Recreation Area
Leaflet | © OpenStreetMap
Longitude
28.609164487113226
Latitude
-81.20548725128175
Cancel
Submit


Add RSO Event
Event Name
Choose Existing RSO
Future Innovators
Choose Event Date and Time
mm / dd / yyyy , --:-- --
Choose Existing Location
UCF Campus
Contact Phone
ex: 654-802-9040
Contact Email
ex: user@university.domain
Cancel
Submit


College Event Website
RSO Events
Private Events
Public Events
Mr. User (user)
Log out
Private UCF Event
Time: 2042-02-02 02:00:00
Phone: 421-422-3222
Email: user@ucf.edu
More Details
Private UCF Event 2
Time: 2042-02-02 05:00:00
Phone: 421-422-3222
Email: user@ucf.edu
More Details

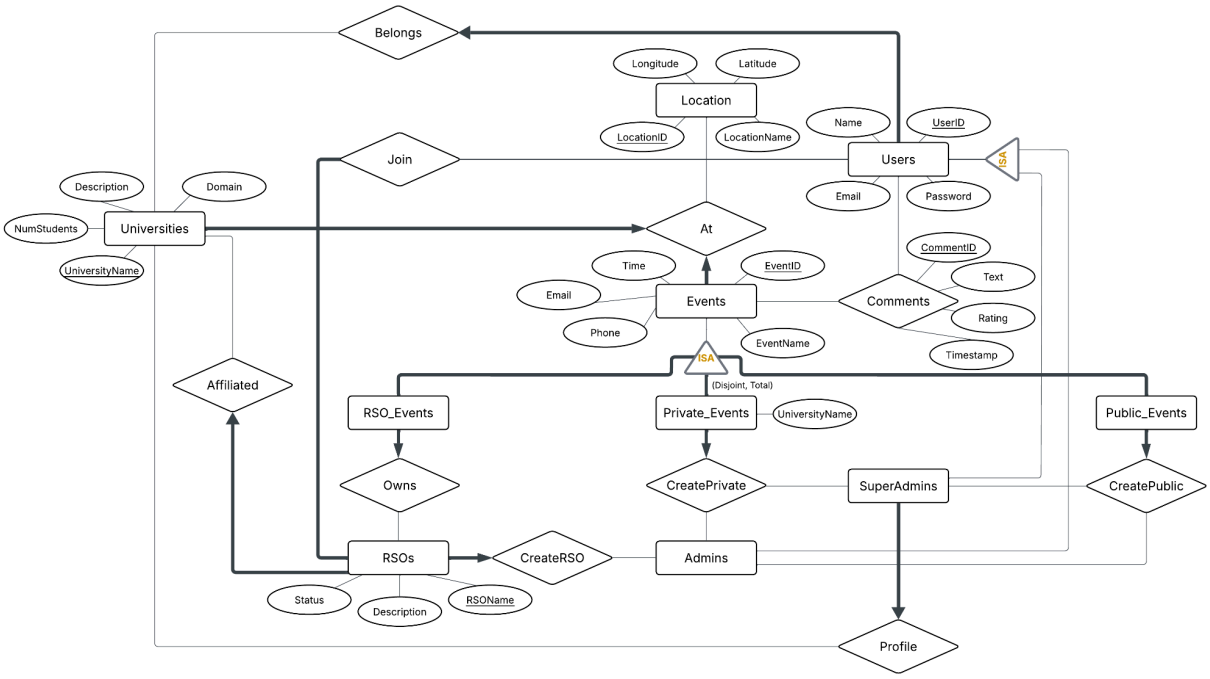

College Event Website
RSO Events
Public UCF Event 1
Time: 2142-02-02 15:00:00
Phone: 421-422-3222
Email: user@ucf.edu
Public USF Event
Time: 2025-08-02 12:00:00
Phone: 900-322-9000
Email: organizer@usf.edu
Public UCF Event 2
Time: 2001-01-02 15:00:00
Phone: 421-422-3222
Event Location & Comments
Location Name
USF Campus
Longitude
28.0527
Latitude
-82.4078
Comments
hello there!
3
Add Comment
User ID: 4
4/9/2025, 6:14:13 PM
I love, I love
Rating: 4
Edit
Delete


User ID: 8
4/9/2025, 8:53:23 PM
hello there!
Rating: 4
Save
Delete



ER-Model

ER Diagram



This is the ER Model which acts as the design structure for the database. This uses the given ER model as a base and adds appropriate attributes, relationships, and entities to support the program's necessities.

Constraints in ER Model

Many-to-One Total Participation

Many entities are marked to have many-to-one total participation. Relational schema has foreign keys be NOT NULL and don't use joint tables to represent the relationships as they only correspond to one entity.

Total Participation of RSOs

Application checks that all RSOs have at least one Admin joined as the model shows that RSOs require total participation.

ISA

Use Assertions of the form RSO_Events $\subseteq$ Events, stating that the class RSO_Events is a subclass of the class Events; taken care of in the relational schema

Disjointness

Use Assertions of the form RSO_Events $\cap$ Private_Events = $\emptyset$, stating disjointness between the two classes: RSO_Events and Private_Events; taken care of in the relational schema

Covering

Use the corresponding Assertion to state that each member of a class must be contained in (at least) one of the covering classes. That is, the superclass is covered by the union of all subclass: RSO_Events $\cup$ Private_Events $\cup$ Public_Events = Events; taken care of in the relational schema

Other Constraints

No Overlapping Events

I made all events one hour long, and used the unique constraint on time and location ID to guard against overlap. Chosen times from the application all start at the top of the hour.

RSO Activity Status

Triggers were implemented that activate after insertion and deletion on the Students_RSO table that will change the status of the associated RSOs.

Shared University Email Domain

Ensure that every user of a university has the same email domain, implemented at the application level. It checks for the email domain which is an attribute of each university when registering, which inserts into the User table and possibly either Admin or SuperAdmins.

I created a trigger that activates after Universities are updated, checking if the domain was changed; if it was modified, then it will change the emails of Users associated with the affected University to have the new domain.

Relational Data Model in SQL

This schema be found in db.sql file for importing

CREATE TABLE

```
CREATE TABLE Locations (
        LocationID INT AUTO_INCREMENT,
        Name CHAR(50),
        Longitude REAL,
        Latitude REAL,
        PRIMARY KEY (LocationID));

CREATE TABLE Universities (
        UniversityName CHAR(50),
        NumStudents INT,
        Domain CHAR(30),
        Description TEXT,
        LocationID INT NOT NULL,
        PRIMARY KEY (UniversityName),
        FOREIGN KEY (LocationID) REFERENCES Locations(LocationID));

CREATE TABLE Users (
        UserID INT AUTO_INCREMENT,
        UniversityName CHAR(50) NOT NULL,
        Name CHAR(50),
        Email CHAR(50),
        Password CHAR(32),
        UNIQUE (Email),
        PRIMARY KEY (UserID),
        FOREIGN KEY (UniversityName) REFERENCES Universities(UniversityName));

CREATE TABLE Admins (
```

```
        UserID INT,
        PRIMARY KEY (UserID),
        FOREIGN KEY (UserID) REFERENCES Users(UserID) ON DELETE CASCADE);

CREATE TABLE SuperAdmins (
        UserID INT,
        PRIMARY KEY (UserID),
        FOREIGN KEY (UserID) REFERENCES Users(UserID) ON DELETE CASCADE);

CREATE TABLE Events (
        EventID INT AUTO_INCREMENT,
        LocationID INT NOT NULL,
        EventName CHAR(50),
        Time DATETIME,
        Phone CHAR(15),
        Email CHAR(50),
        PRIMARY KEY (EventID),
        UNIQUE (LocationID, Time),
        FOREIGN KEY (LocationID) REFERENCES Locations(LocationID));

CREATE TABLE RSO_Events (
        EventID INT,
        RSOName CHAR(50) NOT NULL,
        PRIMARY KEY (EventID),
        FOREIGN KEY (RSOName) REFERENCES RSOs(RSOName) ON DELETE CASCADE,
        FOREIGN KEY (EventID) REFERENCES Events(EventID) ON DELETE CASCADE);

CREATE TABLE Private_Events (
        EventID INT,
        AdminID INT,
        SuperAdminID INT,
        UniversityName CHAR(50),
        PRIMARY KEY (EventID),
        FOREIGN KEY (AdminID) REFERENCES Admins(UserID),
        FOREIGN KEY (SuperAdminID) REFERENCES SuperAdmins(UserID),
        FOREIGN KEY (EventID) REFERENCES Events(EventID) ON DELETE CASCADE,
        CHECK (
                (AdminID IS NOT NULL AND SuperAdminID IS NULL) OR
                (AdminID IS NULL AND SuperAdminID IS NOT NULL)
        ));

CREATE TABLE Public_Events (
        EventID INT,
        AdminID INT,
```

```
        SuperAdminID INT,
        PRIMARY KEY (EventID),
        FOREIGN KEY (AdminID) REFERENCES Admins(UserID),
        FOREIGN KEY (SuperAdminID) REFERENCES SuperAdmins(UserID),
        FOREIGN KEY (EventID) REFERENCES Events(EventID) ON DELETE CASCADE),
        CHECK (
                (AdminID IS NOT NULL AND SuperAdminID IS NULL) OR
                (AdminID IS NULL AND SuperAdminID IS NOT NULL)
        ));

CREATE TABLE Comments (
        CommentID INT AUTO_INCREMENT,
        EventID INT,
        UserID INT,
        Timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
        Text TEXT,
        Rating INT,
        PRIMARY KEY (CommentID),
        FOREIGN KEY (EventID) REFERENCES Events(EventID) ON DELETE CASCADE,
        FOREIGN KEY (UserID) REFERENCES Users(UserID) ON DELETE CASCADE,
        CHECK (Rating BETWEEN 1 AND 5));

CREATE TABLE RSOs (
        RSOName CHAR(50),
        UniversityName CHAR(50) NOT NULL,
        AdminID INT NOT NULL,
        Status   CHAR(10),
        Description TEXT,
        PRIMARY KEY (RSOName),
        FOREIGN KEY (UniversityName) REFERENCES Universities(UniversityName),
        FOREIGN KEY (AdminID) REFERENCES Admins(UserID));

CREATE TABLE Students_RSOs (
        RSOName CHAR(50),
        UserID INT,
        PRIMARY KEY (RSOName, UserID),
        FOREIGN KEY (RSOName) REFERENCES RSOs(RSOName) ON DELETE CASCADE,
        FOREIGN KEY (UserID) REFERENCES Users(UserID) ON DELETE CASCADE);
```

CREATE TRIGGER

```
DELIMITER $$
```

```
	CREATE TRIGGER RSOStatusActive
	AFTER INSERT ON Students_RSOs
	FOR EACH ROW BEGIN
		IF((SELECT COUNT(*) FROM Students_RSOs M where M.RSOName = NEW.RSOName) > 4)
			THEN
				UPDATE RSOs
				SET Status = 'active'
				WHERE RSOName = NEW.RSOName;
		END IF;
	END$$

	CREATE TRIGGER RSOStatusInactive
	AFTER DELETE ON Students_RSOs
	FOR EACH ROW BEGIN
		IF((SELECT COUNT(*) FROM Students_RSOs M where M.RSOName = OLD.RSOName) < 5)
			THEN
				UPDATE RSOs
				SET Status = 'inactive'
				WHERE RSOName = OLD.RSOName;
		END IF;
	END$$

	CREATE TRIGGER UpdateUserEmailDomain
	AFTER UPDATE ON Universities
	FOR EACH ROW BEGIN
		IF OLD.Domain != NEW.Domain THEN
			UPDATE USERS
			SET EMAIL = CONCAT(SUBSTRING_INDEX(Email, '@', 1), '@', NEW.Domain)
			WHERE UniversityName = NEW.UniversityName;
		END IF;
	END$$

DELIMITER ;
```

Sample Data

```
SELECT * FROM `Users`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	UserID	UniversityName	Name	Email	Password
Edit Copy Delete	1	University of Central Florida	John Doe	jdoe@ucf.eduedu	5f4dcc3b5aa765d61d8327deb882cf99
Edit Copy Delete	4	University of Central Florida	Art Blakey	jazz@ucf.eduedu	3e0eb4eb6f7ba146d674f61bbe960052
Edit Copy Delete	8	University of Central Florida	Mr. User	user@ucf.eduedu	c42c1c0308913c1ef44bce18c2ff94d8
Edit Copy Delete	9	University of Central Florida	Avery Wynn	avery.wynn@ucf.eduedu	57fc8244d4d6709ceb2fd2d400a6d632
Edit Copy Delete	10	University of Central Florida	Skyler Dale	skyler.dale@ucf.eduedu	07dd98ba0a1161ce034e783a90d29f1c
Edit Copy Delete	11	University of Central Florida	Reese Lang	reese.lang@ucf.eduedu	61470e5d73ff61e2faade331e97a986a
Edit Copy Delete	12	University of Central Florida	Quinn Mercer	quinn.mercer@ucf.eduedu	029a142b263a32d13fc99ba9561b7d9d
Edit Copy Delete	13	University of Central Florida	Admin 2	admin2@ucf.eduedu	a14dbe401885797b43ce9d66e98968c3
Edit Copy Delete	14	University of South Florida	Mr. USF User	user@usf.edu	5f4dcc3b5aa765d61d8327deb882cf99
Edit Copy Delete	15	University of South Florida	Jane Doe	jdoe@usf.edu	5f4dcc3b5aa765d61d8327deb882cf99
Edit Copy Delete	16	University of South Florida	USF Admin	admin@usf.edu	a14dbe401885797b43ce9d66e98968c3
Edit Copy Delete	19	University of Central Florida	Taylor Finch	taylor.finch@ucf.eduedu	2fd91169bd3f9c26bab811b9322ed6c9

```
SELECT * FROM `Admins`
```

Profiling [Edit inline] [Edit] [Explain SQL]

Show all | Number of rows: 25

Extra options

	UserID
Edit Copy Delete	4
Edit Copy Delete	13
Edit Copy Delete	16
Edit Copy Delete	19

```
SELECT * FROM `SuperAdmins`
```

Profiling [Edit inline] [Edit] [Explain SQL]

Show all | Number of rows: 25

Extra options

	UserID
Edit Copy Delete	1
Edit Copy Delete	15

```
SELECT * FROM `Universities`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	UniversityName	NumStudents	Domain	Description	LocationID
Edit Copy Delete	University of Central Florida	68000	ucf.eduedu	charge on! charge on! charge on! charge on! charge...	5
Edit Copy Delete	University of South Florida	48562	usf.edu	University of South Florida is a school in Florida...	9

```
SELECT * FROM `Locations`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key:

Extra options

	LocationID	Name	Longitude	Latitude
Edit Copy Delete	1	UCF Campus	28.6024	-81.2001
Edit Copy Delete	3	The Embassy	43.7014	-79.4126
Edit Copy Delete	5	Blaze Pizza	28.598907	-81.208381
Edit Copy Delete	8	Lib	28.596957	-81.214478
Edit Copy Delete	9	USF Campus	28.0527	-82.4078
Edit Copy Delete	12	Buisness Building, Room 201	28.597635	-81.197891
Edit Copy Delete	13	UCF Reflecting Pond	28.598087	-81.202354
Edit Copy Delete	14	Engineering Hall, Romm 102	28.600348	-81.198406

```
SELECT * FROM `RSOs`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	RSOName	UniversityName	AdminID	Status	Description
Edit Copy Delete	Future Innovators	University of Central Florida	19	active	A student-led initiative promoting innovating and ...
Edit Copy Delete	Help Evil	University of Central Florida	13	inactive	evil
Edit Copy Delete	Jazz	University of Central Florida	4	inactive	
Edit Copy Delete	Test RSO	University of Central Florida	4	active	This is a test RSO
Edit Copy Delete	Test RSO 2	University of Central Florida	4	inactive	not cool
Edit Copy Delete	Test RSO 3	University of Central Florida	4	inactive	teate
Edit Copy Delete	Test RSO 4	University of Central Florida	4	active	active time
Edit Copy Delete	USF Hacks	University of South Florida	16	inactive	we do the hacks

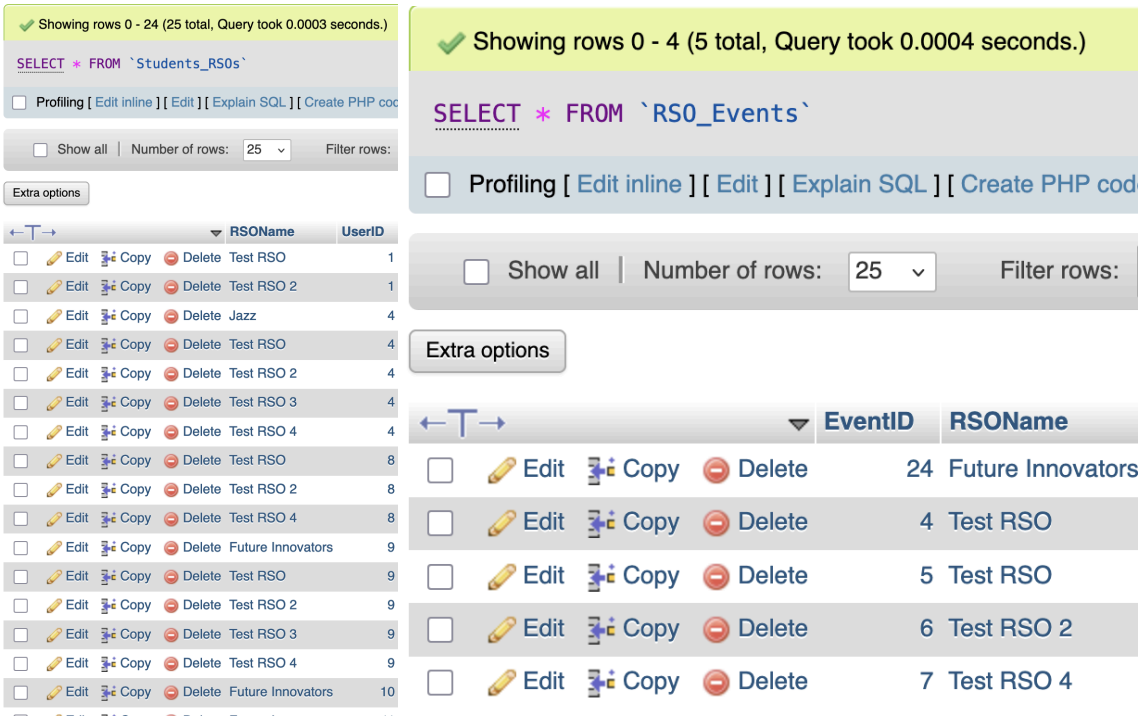
```
SELECT * FROM `Events`
```

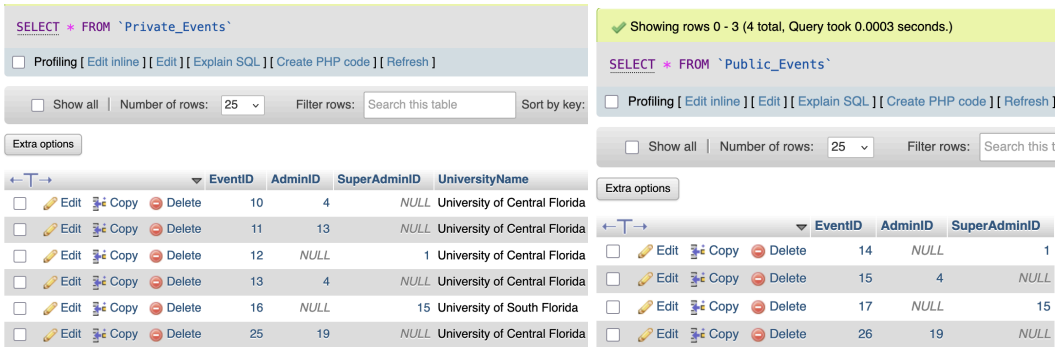
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

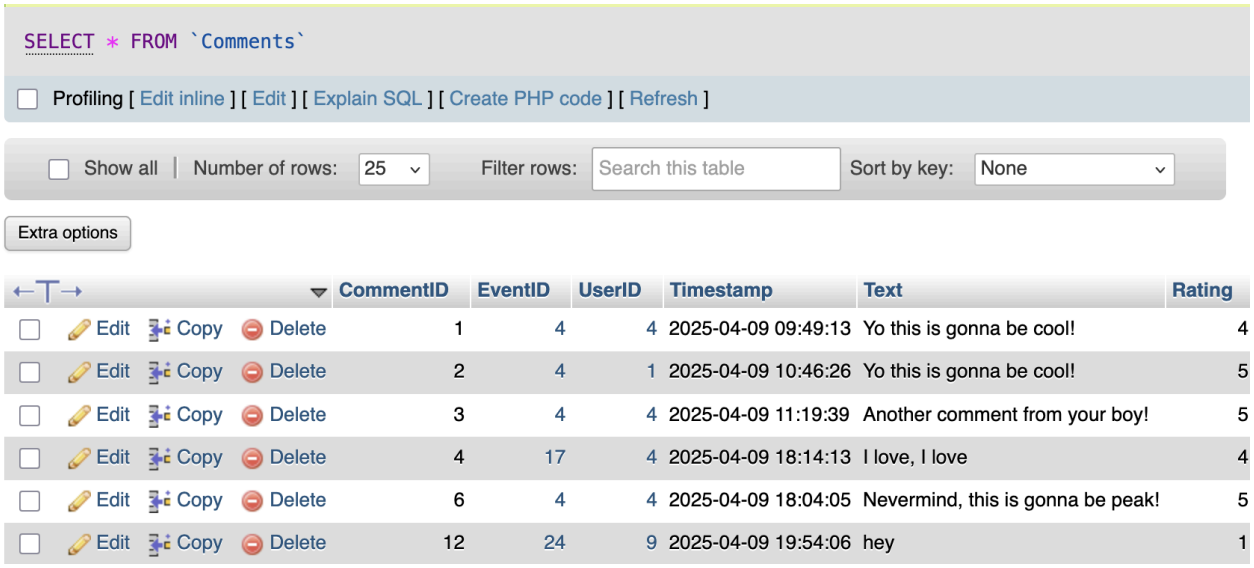
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	EventID	LocationID	EventName	Time	Phone	Email
Edit Copy Delete	4	1	Test RSO Event	2002-02-02 02:00:00	421-421-3212	es@use.edu
Edit Copy Delete	5	1	Test RSO (Event 2)	2002-02-02 14:00:00	643-321-9020	hey@ucf.edu
Edit Copy Delete	6	5	test rso 2 event	2002-02-14 02:00:00	321-421-3212	reea@ucf.edu
Edit Copy Delete	7	5	Science Meeting	2012-04-02 03:00:00	321-421-3322	john@ucf.edu
Edit Copy Delete	10	1	Private UCF Event	2042-02-02 02:00:00	421-422-3222	user@ucf.edu
Edit Copy Delete	11	1	Private UCF Event 2	2042-02-02 05:00:00	421-422-3222	user@ucf.edu
Edit Copy Delete	12	5	Private UCF Event 2	2042-02-02 05:00:00	421-422-3222	user@ucf.edu
Edit Copy Delete	13	5	UCF Epic Sauce	2005-04-21 18:00:00	421-855-9922	sauce.place@ucf.edu
Edit Copy Delete	14	5	Public UCF Event 1	2142-02-02 15:00:00	421-422-3222	user@ucf.edu
Edit Copy Delete	15	5	Public UCF Event 2	2001-01-02 15:00:00	421-422-3222	user@ucf.edu
Edit Copy Delete	16	9	USF Kickoff	2000-03-04 20:00:00	241-421-9003	organizer@usf.edu
Edit Copy Delete	17	9	Public USF Event	2025-08-02 12:00:00	900-322-9000	organizer@usf.edu
Edit Copy Delete	24	14	Innovation Kickoff	2025-04-20 16:00:00	407-123-4567	taylor.finch@ucf.edu
Edit Copy Delete	25	12	Career Prep Workshop	2025-04-21 18:00:00	407-987-6543	admin@ucf.edu
Edit Copy Delete	26	13	Campus Cleanup Day	2025-04-22 10:00:00	407-555-1212	outreach@ucf.edu


Showing rows 0 - 24 (25 total, Query took 0.0003 seconds.)
SELECT * FROM `Students_RSOs`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP cod
Show all | Number of rows: 25 Filter rows:
Extra options
RSOName UserID
Edit Copy Delete Test RSO 1
Edit Copy Delete Test RSO 2 1
Edit Copy Delete Jazz 4
Edit Copy Delete Test RSO 4
Edit Copy Delete Test RSO 2 4
Edit Copy Delete Test RSO 3 4
Edit Copy Delete Test RSO 4 4
Edit Copy Delete Test RSO 8
Edit Copy Delete Test RSO 2 8
Edit Copy Delete Test RSO 4 8
Edit Copy Delete Future Innovators 9
Edit Copy Delete Test RSO 9
Edit Copy Delete Test RSO 2 9
Edit Copy Delete Test RSO 3 9
Edit Copy Delete Test RSO 4 9
Edit Copy Delete Future Innovators 10
Showing rows 0 - 4 (5 total, Query took 0.0004 seconds.)
SELECT * FROM `RSO_Events`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP cod
Show all | Number of rows: 25 Filter rows:
Extra options
EventID RSOName
Edit Copy Delete 24 Future Innovators
Edit Copy Delete 4 Test RSO
Edit Copy Delete 5 Test RSO
Edit Copy Delete 6 Test RSO 2
Edit Copy Delete 7 Test RSO 4


SELECT * FROM `Private_Events`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table Sort by key:
Extra options
EventID AdminID SuperAdminID UniversityName
Edit Copy Delete 10 4 NULL University of Central Florida
Edit Copy Delete 11 13 NULL University of Central Florida
Edit Copy Delete 12 NULL 1 University of Central Florida
Edit Copy Delete 13 4 NULL University of Central Florida
Edit Copy Delete 16 NULL 15 University of South Florida
Edit Copy Delete 25 19 NULL University of Central Florida
Showing rows 0 - 3 (4 total, Query took 0.0003 seconds.)
SELECT * FROM `Public_Events`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows:
Extra options
EventID AdminID SuperAdminID
Edit Copy Delete 14 NULL 1
Edit Copy Delete 15 4 NULL
Edit Copy Delete 17 NULL 15
Edit Copy Delete 26 19 NULL



```
SELECT * FROM `Comments`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	CommentID	EventID	UserID	Timestamp	Text	Rating
Edit Copy Delete	1	4	4	2025-04-09 09:49:13	Yo this is gonna be cool!	4
Edit Copy Delete	2	4	1	2025-04-09 10:46:26	Yo this is gonna be cool!	5
Edit Copy Delete	3	4	4	2025-04-09 11:19:39	Another comment from your boy!	5
Edit Copy Delete	4	17	4	2025-04-09 18:14:13	I love, I love	4
Edit Copy Delete	6	4	4	2025-04-09 18:04:05	Nevermind, this is gonna be peak!	5
Edit Copy Delete	12	24	9	2025-04-09 19:54:06	hey	1

SQL Examples

Insert New RSO

```
SELECT RSOName FROM RSOs WHERE RSOName = "My RSO"
```

```
INSERT into RSOs (RSOName, UniversityName, AdminID, Status, Description) VALUES ("My RSO", "University of Central Florida", 4, "inactive", "pretty cool club")
```

```
INSERT INTO Students_RSOs (RSOName, UserID) VALUES ("My RSO", 4)
```

```
SELECT UserID FROM Users WHERE Email = "john@ucf.edu"
```

```
INSERT INTO Students_RSOs (RSOName, UserID) VALUES ("My RSO", 7)
```

Example of inserting a new RSO with name "My RSO" with University being "University of Central Florida" with the status being inactive by default; checks for duplicate RSO Names first. The admin associated with this has a UserID = 4. An additional member is added by finding associated UserID of email "john@ucf.edu", and is associated with the new RSO by adding to the relationship table "Students_RSOs".

Add Student to Existing RSO

```
SELECT UserID FROM Users WHERE Email = "jdoe@ucf.edu"
```

```
SELECT 1 FROM Students_RSOs WHERE RSOName = "Test RSO" AND UserID = 1
```

```
INSERT INTO Students_RSOs (RSOName, UserID) VALUES (“Test RSO”, 1)
```

Example of adding a student to an existing RSO, “Test RSO”, given the email of the user, “jdoe@ucf.edu”. We query for the UserID using the user’s email and we use this to insert the user to the relationship table of “Students_RSOs” with the two primary keys.

Insert New Event

```
INSERT INTO Events (EventName, Time, LocationID, Phone, Email) VALUES (“RSO Event”, “2002-02-02T20:00”, 1, “964-213-8020”, “contact@ucf.edu”)
```

```
INSERT INTO RSO_Events (RSOName, EventID) VALUES (“RSO Event”, 4)
```

Example of adding an event, specifically a RSO event. Only admins can add an event for an associated RSO. Do insertion into Events table, and then get generated EventID (auto-increment) if successful; will not be successful if same time and LocationID. Then add to the RSO_Events table with RSOName (primary key for RSOs) and the EventID.

Comment

Insert

```
INSERT INTO Comments (EventID, UserID, Text, Rating) VALUES (4, 4, “This is a cool event!”, 5)
```

Insert a comment with the associated EventID and UserID foreign keys and the Text and Rating attributes. In this case a comment is inserted for an event with EventID = 4 with the user having UserID = 4. The Text is “This is a cool event!” and the Rating is 5.

Update

```
UPDATE Comments SET Text = “Nevermind. This is going to be peak!”, Rating = “5” WHERE CommentID= 6
```

Update a comment with the associated CommentID (CommentID = 6), updating Text and Rating attributes. The timestamp attribute is set to automatically update when UPDATE occurs, so we do not need to supply the query with it.

Display Events

Public

```
SELECT PUE.EventID, E.LocationID, E.EventName, E.Time, E.Phone, E.Email
          FROM Public_Events PUE
          JOIN Events E ON PUE.EventID = E.EventID
```

Get rows of information about Public events to display. This query joins the Public_Events and Events tables to get the events that are public, then gets the event information for public events only.

Private

```
SELECT PRE.EventID, E.LocationID, E.EventName, E.Time, E.Phone, E.Email
          FROM Private_Events PRE
          JOIN Events E ON PRE.EventID = E.EventID
          WHERE PRE.UniversityName = "University of Central Florida"
```

Get rows of information about Private events to display. This query joins the Private_Events and Events tables to get the events that are private, then gets the event information for private events of the UniversityName, which in this case is "University of Central Florida"

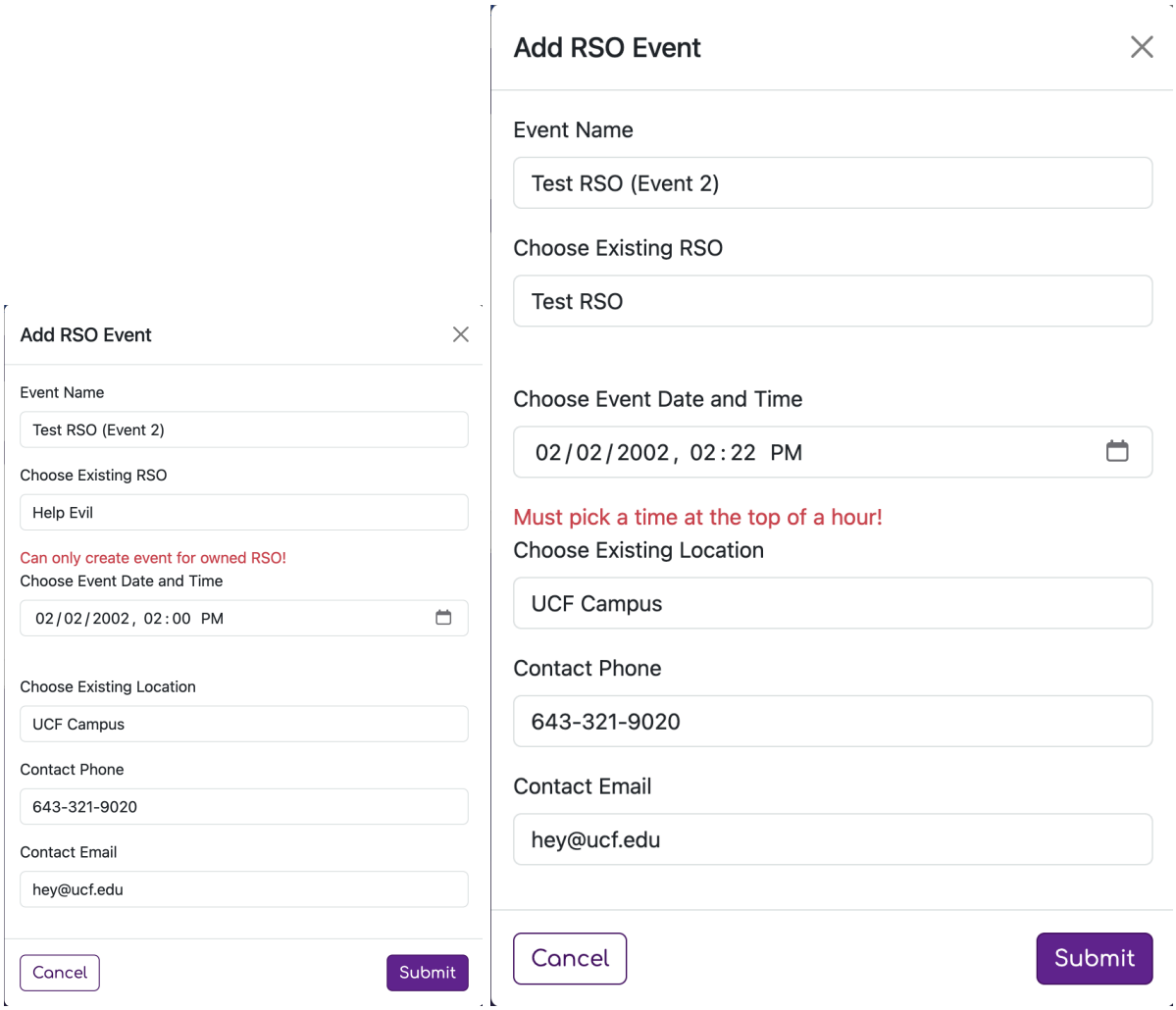
RSO

```
SELECT RE.EventID, RE.RSOName, E.LocationID, E.EventName, E.Time, E.Phone, E.Email
          FROM Students_RSOs SR
          JOIN RSO_Events RE ON SR.RSOName = RE.RSOName
          JOIN Events E ON RE.EventID = E.EventID
          WHERE SR.UserID = 10
```

Get rows of information about RSO events to display including the "RSOName". This query joins the RSO_Events and Events tables with the Students_RSOs table, searching for membership of the UserID (in this case UserID = 10) in RSO, then gets the event information for each RSO that is associated with the user.

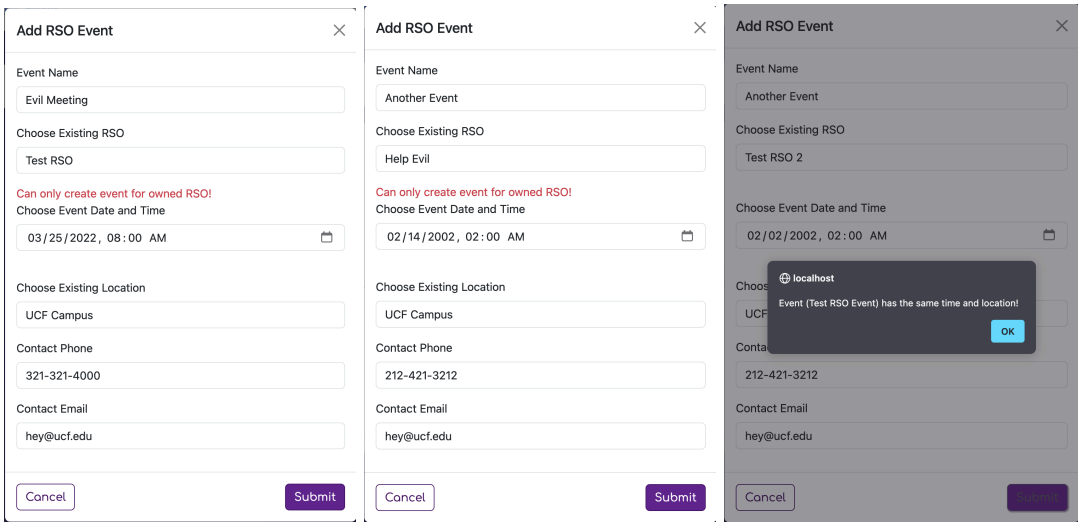
Constraint Enforcement

Conflicting Events



I am using the approach that all events are an hour long and can only start at the top of the hour. Error message for the “Choose Event Date and Time” field is shown if the chosen time is not at the top of the hour. Error alert is shown if an event has overlapping time and location, which is enforced by having the LocationID and Time combination of fields be set to UNIQUE and also shows the conflicting event in the alert.

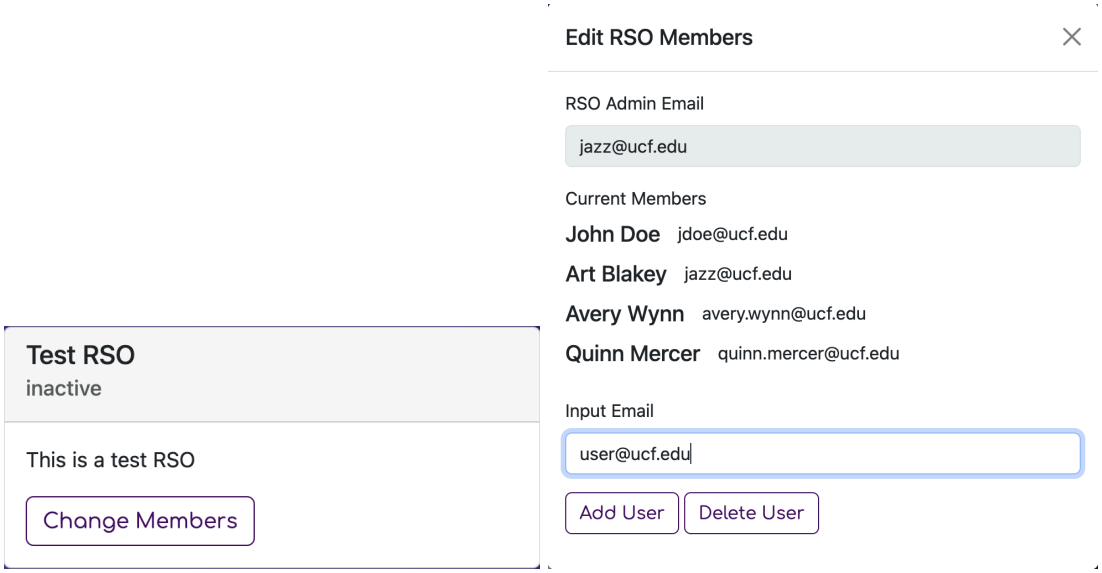
RSO Event Creation

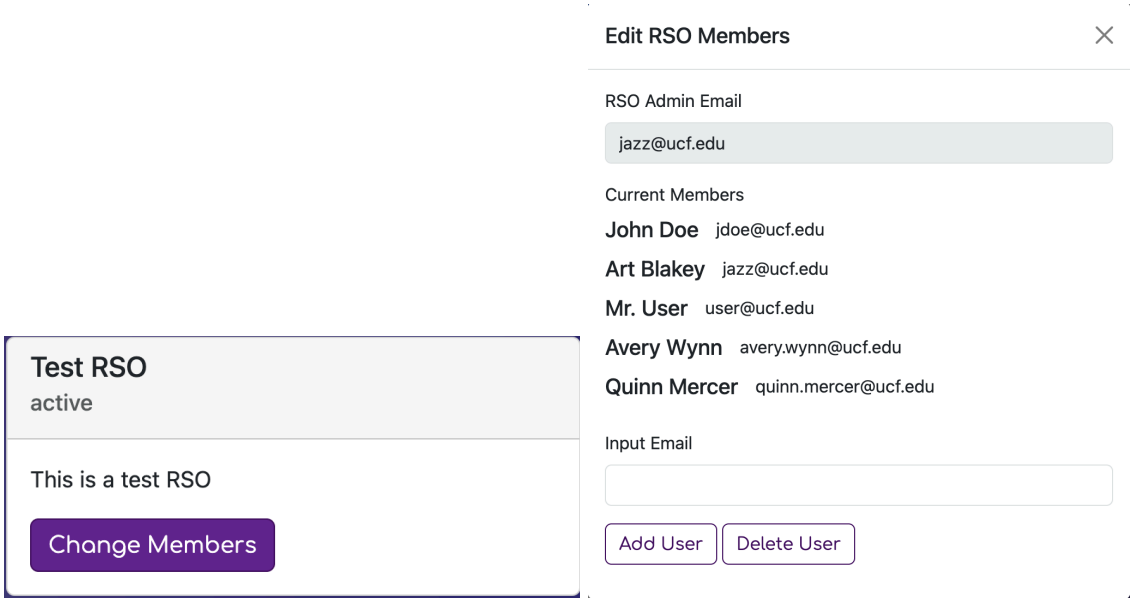


This user is trying to create a RSO Event for a RSO they don't own and it shows an error for the "Choose Existing RSO" field as a result.

RSO Status Change

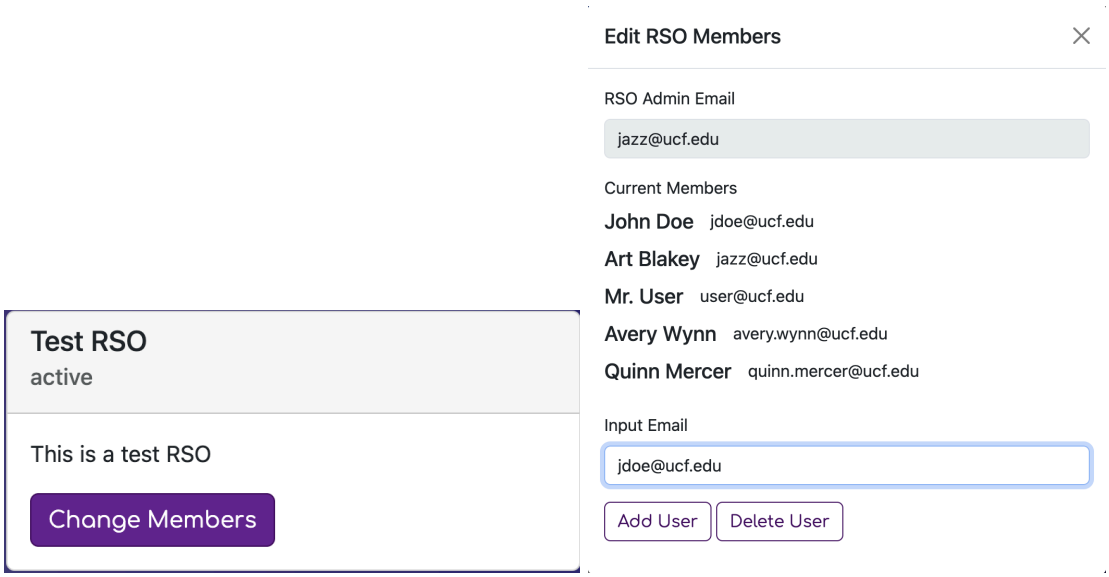
Insert Member into 4 Member RSO (Active Status)

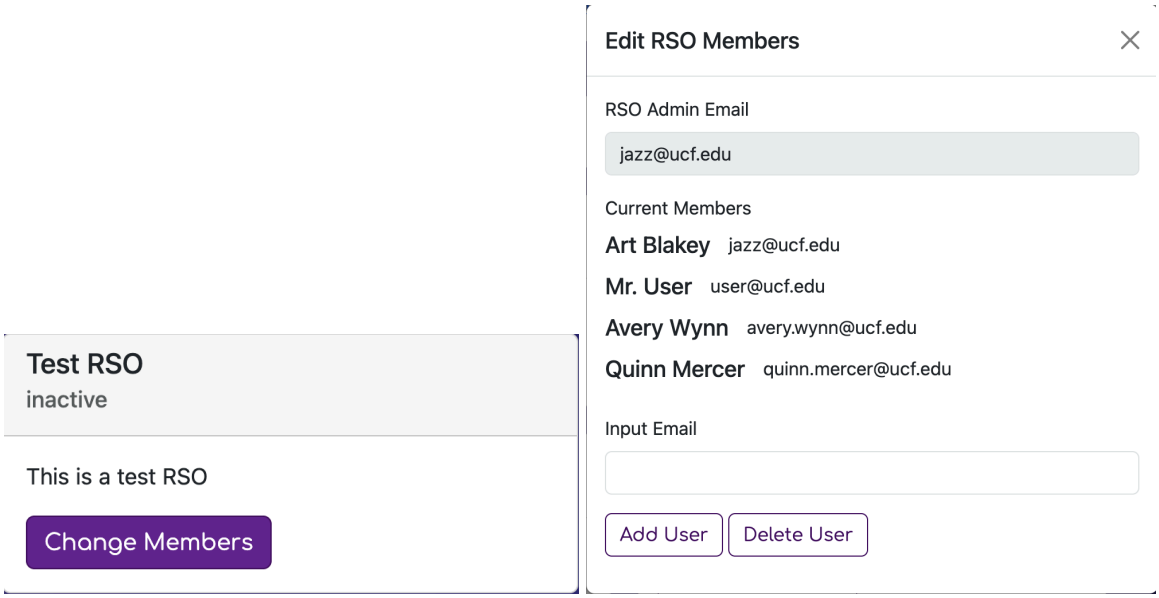




Before adding a member to RSO "Test RSO" with 4 members, the status is 'inactive'. Afterwards, with 5 members now, the status is 'active'.

Delete Member from 5 Member RSO (Inactive Status)





Before deleting a member from RSO “Test RSO” with 5 members, the status is ‘active’. Afterwards, with 4 members left, the status is ‘inactive’.

Conclusion

Database Performance

Due to not limiting the results of queries in my implementation, the performance can suffer due to having to gather a bunch of results from the table. Advanced indexing could be implemented to increase efficiency in data retrieval. For the small scale project and running locally on my machine, it was efficient enough to not cause any noticeable issues. I do feel I made sure to limit the queries when updating fields or adding items such as events, RSOs, or comments, making sure not to fetch all results after updating.

Desired Features

Advanced features were not implemented here such as event feed integration and social media sharing. I believe these could be added into the current project without too many problems, but it may require refactoring some of the code.

Retrospective

While I was able to include cookies for saving user data and encryption for passwords, role access for pages is not completely secure; while navigation to certain features are disabled, and there are checks for the role with it being determined after logging in or registering, many elements are merely just disabled rather than so it is possible to navigate to certain pages if the URL is known which shouldn't be the case.

Due to the higher amount of code, I would consider using another frontend framework that facilitates separation of code and better organization to practice SOLID design principles. That being said, using a simpler stack helped to improve my skills in the backend which I had been less familiar with. I understand the reasoning behind building an ERD first as the foundational structure of the database, and then constructing the schema based off of it. These steps helps to reinforce my ability to retrieve and store data using a DBMS.