

Group Report

College Event Website

COP2710

T/TH 3:00pm - 4:15pm

2022 Spring

Team members: Kriya Naidu, Aliza Grabowski, Julian Boaz

Table Of Contents:

[Project Description](#)

[GUI](#)

[ER-Model](#)

[Relational Data Model](#)

[Populated Data](#)

[SQL Examples & Results](#)

[Constraints](#)

[Advanced Features](#)

[Obstacles Faced & Overcame](#)

[Demo](#)

[Conclusion](#)

## **Project Description**

College Events Website is a website that utilizes a database to hold college information, student information, and events that happen within RSO's (Registered Student Organizations) of each college. This website should have the ability for a college to create a signup and establish its college onto the College Events Website. It also should have a signup for students to enroll in an existing college.

After a student signs up and logs into the website they will be presented with a screen that allows them to choose between creating an RSO, Join an RSO, View Events. To create an RSO it must have a RSO Leader and 4 additional members. When an RSO is established the RSO Leader can create events. These events must have a title, description, location, date, time, and finally specify if it's public, private, or an RSO event. In the View Events a student should be able to scroll through public events, their own school's private events, and RSO events that they are a part of. Then be able to comment and rate on the viewable events.

When a college signs up and logs into the College Event Website, they will be presented with a list of public events that their college's RSOs want to establish. These events must be accepted before they can be viewed by students across all colleges. If these events are not approved they will decline and not be viewable by students.

Our group had to place a couple of assumptions into the College Event Website project. Both being about the RSO side of the project. The first being that we assume that a student can only be a part of one RSO and not a part of multiple. We also assumed that if a student is creating an RSO, that they are the RSO admin for the created RSO. These assumptions helped to create a sturdier website.

## **GUI**

The GUI of the 'College Event Website' is written in both HTML and CSS language. The GUI then communicates to the Javascript file which receives its information from the PHP file, to gather needed information from the database.

Utilizing a GUI allows for the database to communicate with the users, with an easy to understand interface.



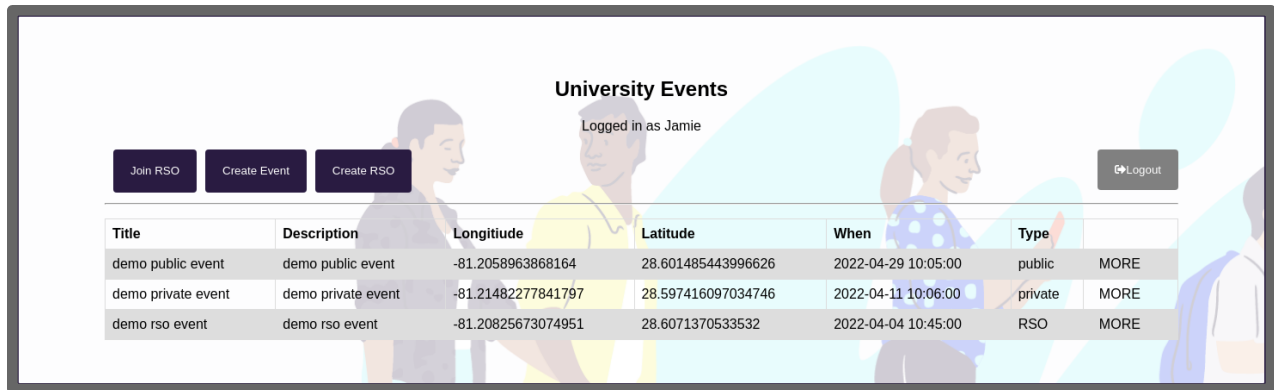


Figure 3: View Events

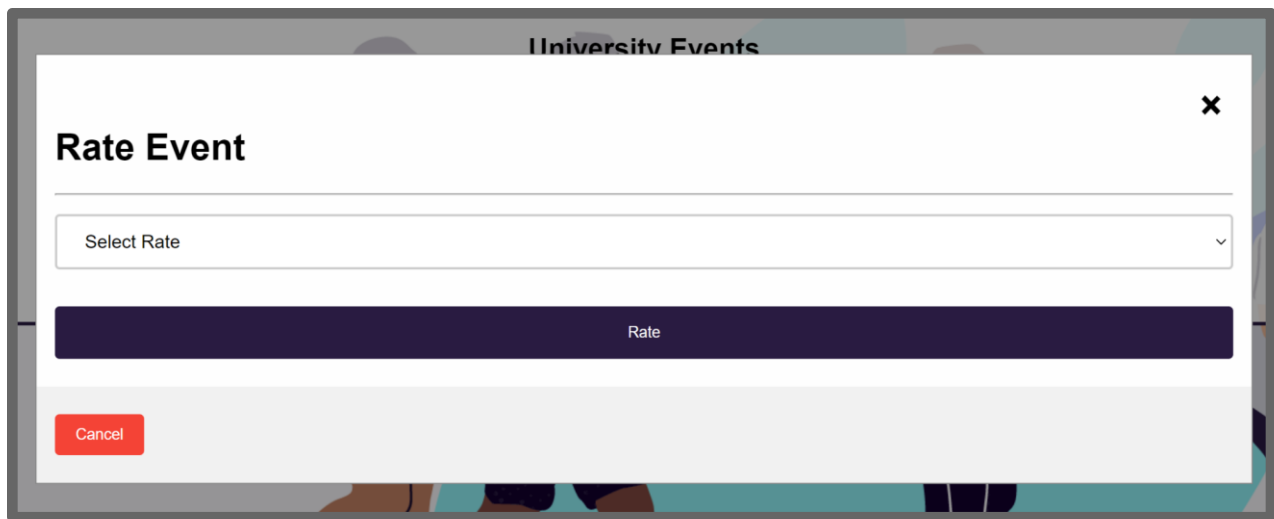
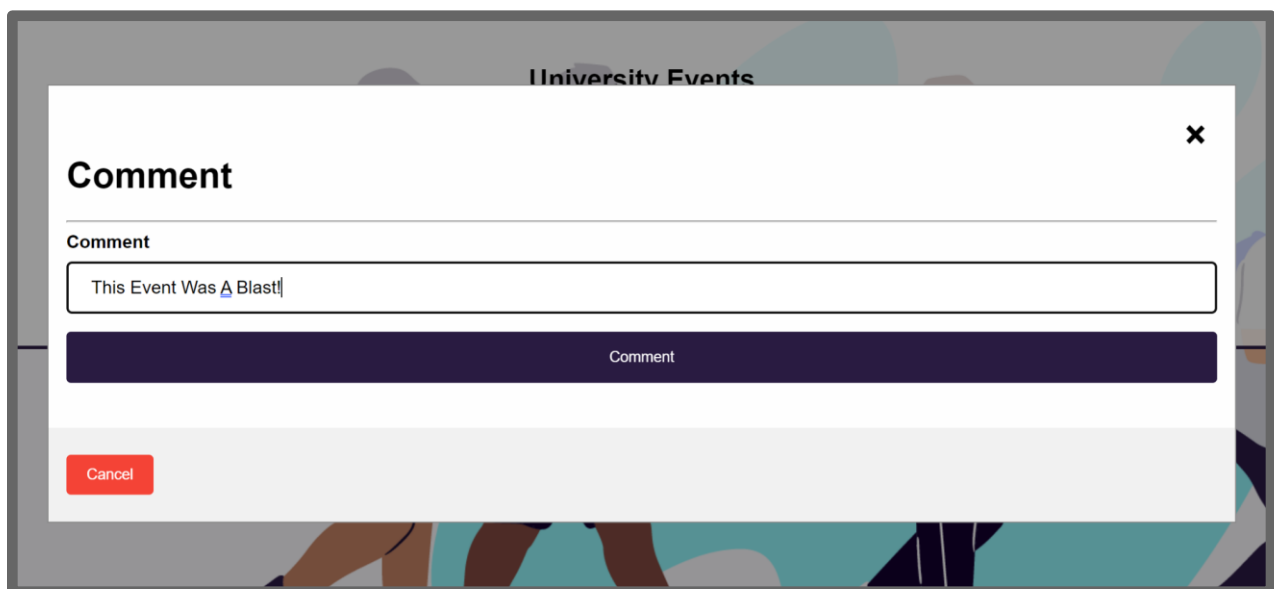


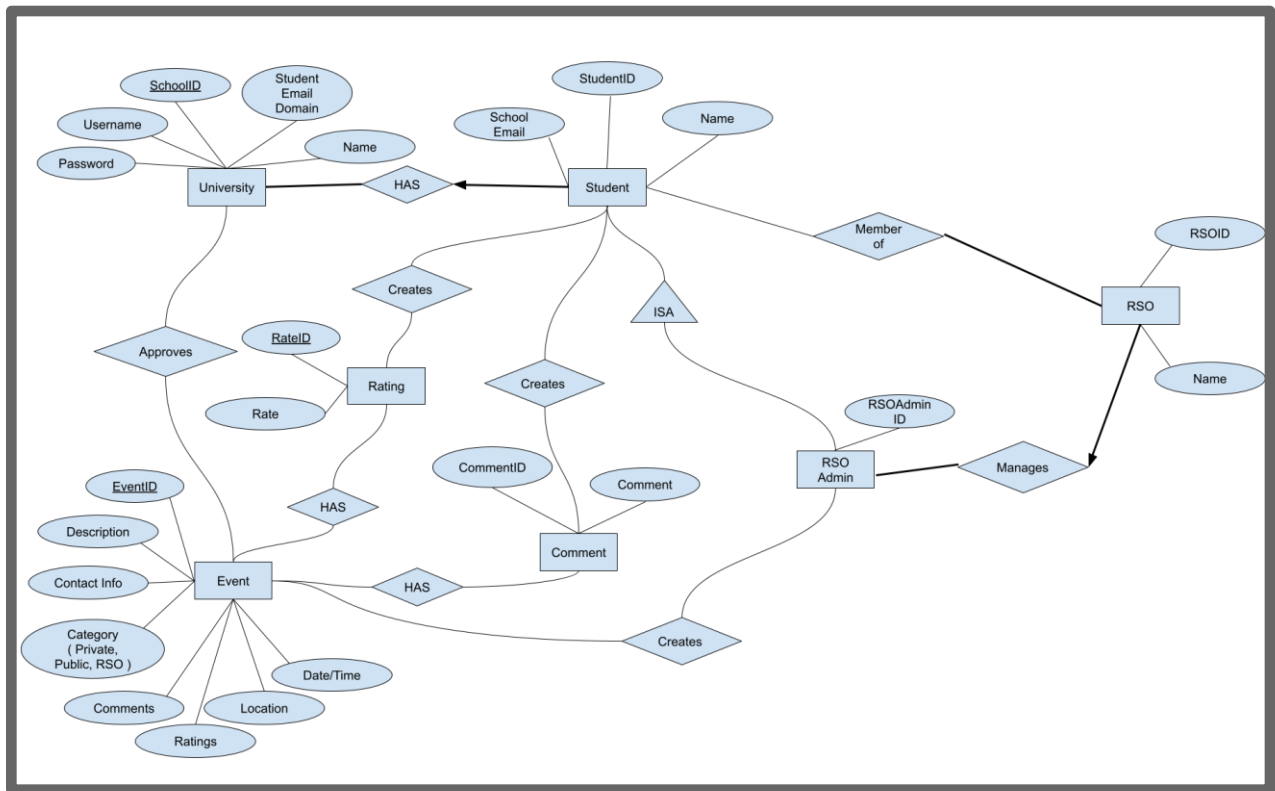
Figure 4: Rate Event



**Figure 5: Comment on Event**

## ER-Model

This project entails multiple constraints that the ER diagram cannot illustrate. One being that a Private and RSO Event do not need approval from the University, but a Public Event does need the approval of the university it is taking place in. Another constraint that cannot properly be presented is establishing an RSO group. To have an active RSO group that can create events, there must be a leader and 4 additional members. In addition to establishing a student user their university must be created prior to the student being able to create an account.



**Figure 6: College Event Website's ER Diagram**

## Relational Data Model

The relational data model shows all the tables that are needed in the College Event Website project. These tables are then created in SQL. The following are the tables that are used in our group's College Event Website.

### SQL 'Create A Comment Table'-

```
CREATE TABLE `Comments` (  
    `id` int NOT NULL primary key,  
    `Comment` varchar(255) NOT NULL,  
    `EventID` varchar(255) NOT NULL,  
    `StudentName` varchar(255)  
        CHARACTER SET utf8mb4  
        COLLATE utf8mb4_0900_ai_ci NOT NULL  
);
```

### SQL 'Create A Event Table'-

```
CREATE TABLE `Events` (  
    `id` tinyint NOT NULL primary key,  
    `Datetime` datetime NOT NULL,  
    `Name` varchar(255) NOT NULL,  
    `Description` varchar(255) NOT NULL,  
    `Type` varchar(255) NOT NULL,  
    `SchoolID` varchar(255) NOT NULL,  
    `Approved` varchar(255) NOT NULL,  
    `RSOID` varchar(255) NOT NULL,  
    `Longitude` varchar(255) NOT NULL,  
    `Latitude` varchar(255) NOT NULL  
);
```

### SQL 'Create A Rating Table'-

```
CREATE TABLE `Ratings` (  
    `id` int NOT NULL primary key,  
    `EventID` varchar(255) NOT NULL,  
    `Rating` varchar(255) NOT NULL
```

);

**SQL 'Create A RSO Table'-**

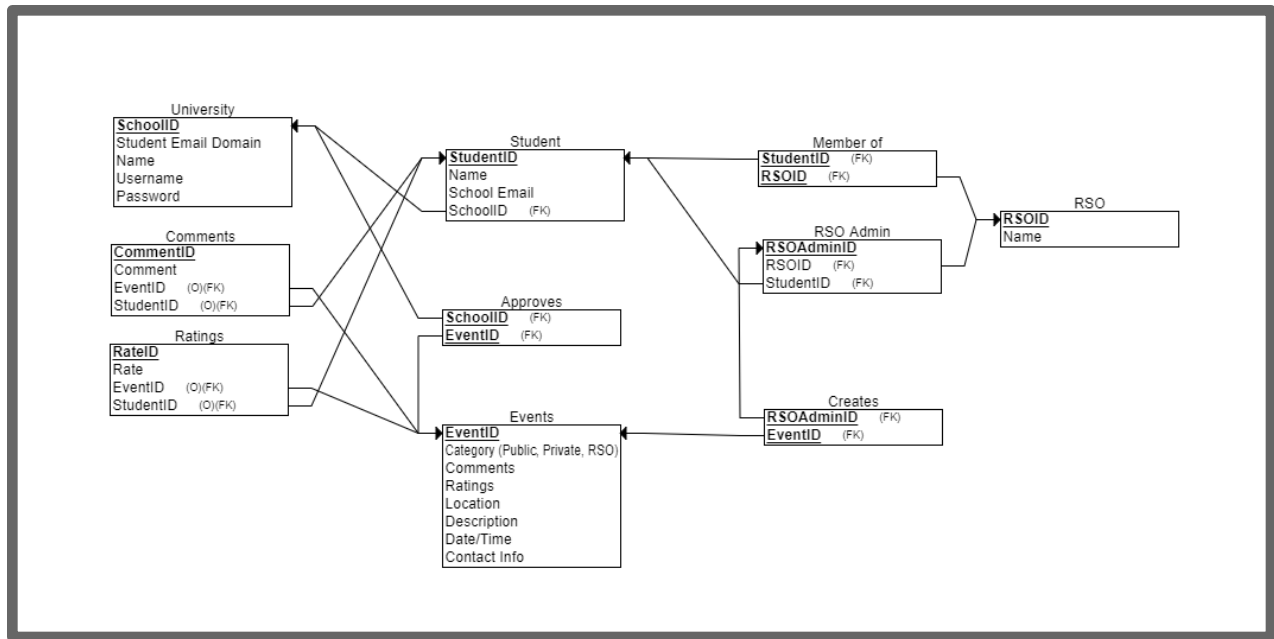
```
CREATE TABLE `RSO` (  
    `id` int NOT NULL primary key,  
    `Name` varchar(255) NOT NULL,  
    `SchoolID` varchar(255) NOT NULL  
);
```

**SQL 'Create A Student Table'-**

```
CREATE TABLE `Student` (  
    `StudentID` tinyint NOT NULL primary key,  
    `Name` varchar(255) NOT NULL,  
    `Password` varchar(255) NOT NULL,  
    `School_Email` varchar(255) NOT NULL,  
    `School` varchar(255) NOT NULL,  
    `RSOAdminID` varchar(255) CHARACTER SET utf8mb4 COLLATE  
utf8mb4_0900_ai_ci NOT NULL DEFAULT '-1',  
    `RSOID` varchar(255) CHARACTER SET utf8mb4 COLLATE  
utf8mb4_0900_ai_ci NOT NULL DEFAULT '-1'  
)
```

**SQL 'Create A University Table'-**

```
CREATE TABLE `University` (  
    `SchoolID` tinyint NOT NULL primary key,  
    `Name` varchar(255) NOT NULL,  
    `Password` varchar(255) NOT NULL,  
    `Username` varchar(255) NOT NULL  
);
```



**Figure 7: College Event Website's Relational Diagram**

## Populated Data

What's a database without any data to make sure the GUI is connected and fully functionable?

This section will show multiple screenshots of each of our tables with established input.

SchoolID	Name	Password	Username
1	test	test	test
3	jules	58e537189c53a9d3969e8795588ea574	jules
4	UCF	098f6bcd4621d373cade4e832627b4f6	ucf
5	kriya	fcad881731be3ea01d1189eb7ce37337	kriya
6	Aliza	25b805c991c15d944f95f1ef33899615	Aliza

**Figure 8: University Database Table With 6 Universities**



StudentID	Name	Password	School_Email	School	RSOAdminID	RSOID
1	test	098f6bcd4621d373cade4e832627b4f6	test	5	9	
2	julian	0bb22c74181003d22f2f6382dc47c758	jb@gmail.com	5	-1	
3	secondtest	42f4cc484e89c74f33ca1d2c1cace4d3	secondtest	5	-1	9
4	Kriya	fc8d881731be3ea01d1189eb7ce37337	kirya@ucf.edu	4	-1	-1
5	Aliza	25b805c991c15d944f95f1ef33899615	Aliza@ucf.edu	4	-1	-1
6	Dylan	709488e90b981b35e9a634b6ab414277	dylan@ucf.edu	4	-1	-1
7	Nick	06f7c4187975fb44eb9314b9a4019912	nick@ucf.edu	4	-1	-1
8	Jamie	ff57033d9796ab4c4c8c8d5badcf20a2	jamie@ucf.edu	4	-1	-1
9	Katrina	caa9d463ca593d626c753f4bec0c1450	katrina@ucf.edu	4	-1	-1
10	Jacob	c22289fcb12f66d070f1e802864d4dc4	jacob@ucf.edu	4	-1	-1
11	Marshall	84484fa33f17f7a763c9555f46bac8d2	marshall@ucf.edu	4	-1	-1
12	Michael	6b14f8250b391463ed27635dc528bb21	michael@ucf.edu	4	-1	-1
13	Thomas	a7397d8d7e79daa0609c1e92782712f7	thomas@ucf.edu	4	-1	-1
14	Timothy	fab6a71812c39c2703b634c40bb76221	timothy@ucf.edu	4	-1	-1
15	Olivia	2764761ddc6ce843b4badd542b742693	olivia@ucf.edu	4	-1	-1
16	Denise	c4802856e4fa3f0a980605b5fce91970	denise@ucf.edu	4	-1	-1
17	Anthony	de4e6ec2d2ab2e0907aab4145559ee5d	anthony@ucf.edu	4	-1	-1
18	Geoff	66e764e4ea6e3a7535d256e356ef84f7	geoff@ucf.edu	4	-1	-1
19	Jesuani	4b4efa8b787d904a755f3f38bd934c6c	jesuani@ucf.edu	4	-1	-1
20	James	5721c634b6c516e6b87417f1ca775be5	james@ucf.edu	4	-1	-1

**Figure 9: Student Database Table With 20 Student Users**

StudentID	Name	Password	School_Email	School	RSOAdminID	RSOID
1	test	098f6bcd4621d373cade4e832627b4f6	test	5	9	9
2	julian	0bb22c74181003d22f2f6382dc47c758	jb@gmail.com	5	-1	9
3	secondtest	42f4cc484e89c74f33ca1d2c1cace4d3	secondtest	5	-1	9
4	Kriya	fc8d881731be3ea01d1189eb7ce37337	kirya@ucf.edu	5	-1	-1
5	Aliza	25b805c991c15d944f95f1ef33899615	Aliza@ucf.edu	5	-1	9
6	Dylan	709488e90b981b35e9a634b6ab414277	dylan@ucf.edu	5	-1	9

**Figure 10: RSO Database Table with 6 members**

## SQL Examples & Results

This section will show functions that are needed to access the database via SQL and the result they output onto the website.

**SQL ‘Create New RSO’ -**

```

SELECT RSO AdminID
FROM Student
WHERE (StudentID = S)

```

```

INSERT INTO RSO (Name)
VALUES (S)
UPDATE Student
SET RSOAdminID = (SELECT id
                  FROM RSO
                  WHERE Name = name)
                  WHERE StudentID = ID

UPDATE Student
SET RSOID = (SELECT id
             FROM RSO
             WHERE Name = name)
            WHERE StudentID = ID

```

**Result -**

StudentID	Name	Password	School_Email	School	RSOAdminID	RSOID
1	test	098f6bcd4621d373cade4e832627b4f6	test	5	9	9

**Figure 11: Create New RSO Result**

**SQL 'Join RSO' -**

```

UPDATE Student
SET RSOID = rsoID
WHERE StudentID = ID

```

**Result -**

StudentID	Name	Password	School_Email	School	RSOAdminID	RSOID
1	test	098f6bcd4621d373cade4e832627b4f6	test	5	9	9
2	julian	0bb22c74181003d22f2f6382dc47c758	jb@gmail.com	5	-1	9

**Figure 12: Join RSO Result**

**SQL 'Create Event' -**

```

SELECT RSOAdminID
    FROM Student
    WHERE (StudentID = id)
SELECT Datetime
    FROM Events
    WHERE Longitude = long AND Latitude = lat
SELECT *
    FROM Student
    WHERE RSOID = (SELECT RSOAdminID
                    FROM Student
                    WHERE (StudentID = id))
INSERT INTO Events (Name, Description, Longitude, Latitude, Datetime, Type,
                    Approved, RSOID, SchoolID)
    VALUES (name, description, long, lat, date, type, approval, rsoID,
            schoolID)
UPDATE Events
    SET SchoolID = (SELECT School
                    FROM Student
                    WHERE StudentID = id)
    WHERE Name = name

```

#### Result -

	id	Datetime	Name	Description	Type	SchoolID	Approved	RSOID	Longitude	Latitude
10	10	2022-04-15 20:37:00	maps test	maps test	RSO	5	-1	9	-81.2000599	28.6024274

**Figure 13: Create Event Result**

#### SQL ‘Create/update Event Comment’ -

```

SELECT Name
    FROM Student
    WHERE StudentID = id
INSERT INTO Comments (Comment, EventID, StudentName)

```

VALUES (comment, event, id)

**Result -**

	id	Comment	EventID	StudentName
e	5	test comment	16	Jamie

**Figure 14: Create Event Comment Result**

**SQL ‘View Events By User’ -**

SELECT \*

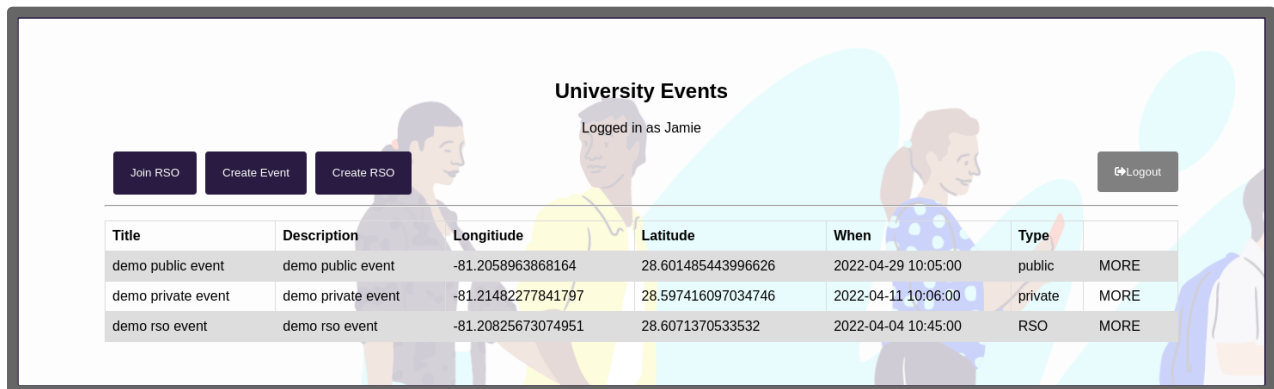
FROM Events

WHERE (SchoolID = (SELECT SchoolID

FROM Student

WHERE StudentID = id)

**Result -**



Title	Description	Longitude	Latitude	When	Type	
demo public event	demo public event	-81.2058963868164	28.601485443996626	2022-04-29 10:05:00	public	MORE
demo private event	demo private event	-81.21482277841797	28.597416097034746	2022-04-11 10:06:00	private	MORE
demo rso event	demo rso event	-81.20825673074951	28.6071370533532	2022-04-04 10:45:00	RSO	MORE

**Figure 15: View Event Result**

## Constraints

Here we will show the effects of the constraints and the errors they present. Such as an error displayed for if there is an overlap or conflict with the time of an event, a non RSOadmin trying to create an event, and the need for 5 group members to ‘Activate’ an RSO.

Select Category

Add Event

Error creating event, conflicting time/location

Cancel

**Figure 16: Error Message When Events Overlap**

Select Category

Add Event

You are not an RSO admin so you cannot create events. Create an RSO to access this function.

Cancel

**Figure 17: Error Message When NonRSO Admin Tries Creating Event**

Add Event

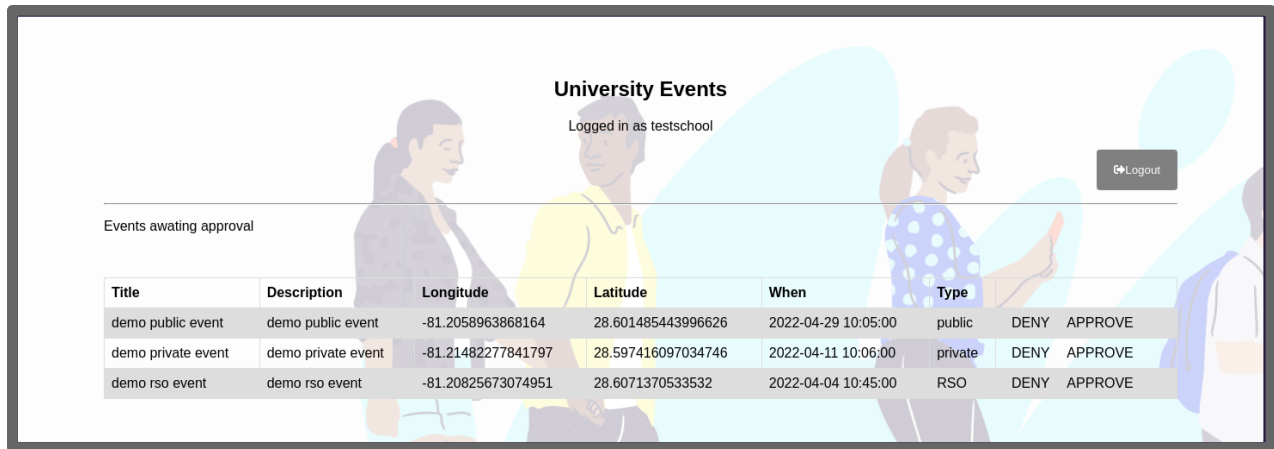
RSO must have 5 members to be active and create events

Cancel

**Figure 18: RSO That is 'InActive' RSO Cannot Create Event**

<input type="checkbox"/>	Edit	Copy	Delete	8	2022-05-06 14:35:00	First meeting!	First Meeting!	UCF Campus	RSO	5	-1	9
--------------------------	------	------	--------	---	---------------------	----------------	----------------	------------	-----	---	----	---

**Figure 19: RSO Becomes 'Active' Event Goes Into Database**



**Figure 20: Approval / Non Approval Page In School User Side**

Id	Datetime	Name	Description	Type	School	Approved	RSOID	Longitude	Latitude
10	2022-04-15 20:37:00	maps test	maps test	RSO	5	-1	9	-81.2000599	28.6024274
11	2022-05-07 20:38:00	second maps	second maps	RSO	5	-1	9	-81.2044372651123	28.598734884141795
12	2022-04-29 10:05:00	demo public event	demo public event	public	7	-1	-1	-81.2058963868164	28.601485443996626
13	2022-04-11 10:06:00	demo private event	demo private event	private	7	-1	-1	-81.21482277841797	28.597416097034746
16	2022-04-04 10:45:00	demo rso event	demo rso event	RSO	7	-1	11	-81.20825673074951	28.6071370533532

**Figure 21: Approval / Non Approval Database**

## Advanced Features

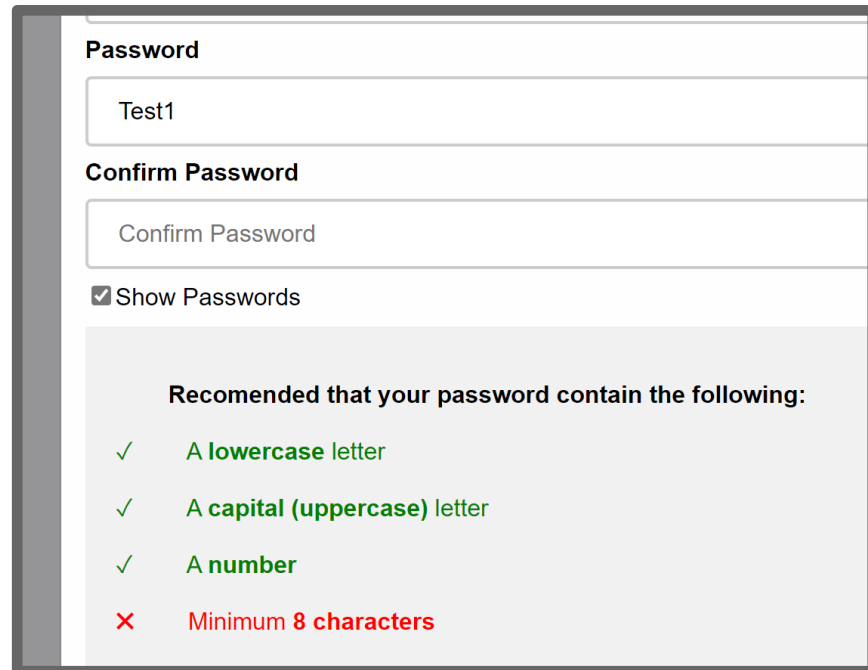
The advanced features our group decided on is the establishment of an encrypted password, a specified requirements for a password, as well as cookies. The mixture of an encrypted password and the requirement for a password allows our website to be more secure from hackers. Then Cookies allow the user to access the webpage within a specified amount of time without the need of logging back in.

StudentID	Name	Password	School_Email
1	test	098f6bcd4621d373cade4e832627b4f6	test
2	julian	0bb22c74181003d22f2f6382dc47c758	jb@gmail.com
3	secondtest	42f4cc484e89c74f33ca1d2c1cace4d3	secondtest
4	Kriya	fc8d881731be3ea01d1189eb7ce37337	kirya@ucf.edu
5	Aliza	25b805c991c15d944f95f1ef33899615	Aliza@ucf.edu
6	Dylan	709488e90b981b35e9a634b6ab414277	dylan@ucf.edu
7	Nick	06f7c4187975fb44eb9314b9a4019912	nick@ucf.edu
8	Jamie	ff57033d9796ab4c4c8c8d5badcf20a2	jamie@ucf.edu
9	Katrina	caa9d463ca593d626c753f4bec0c1450	katrina@ucf.edu
10	Jacob	c22289fcb12f66d070f1e802864d4dc4	jacob@ucf.edu
11	Marshall	84484fa33f17f7a763c9555f46bac8d2	marshall@ucf.edu
12	Michael	6b14f8250b391463ed27635dc528bb21	michael@ucf.edu
13	Thomas	a7397d8d7e79daa0609c1e92782712f7	thomas@ucf.edu
14	Timothy	fab6a71812c39c2703b634c40bb76221	timothy@ucf.edu
15	Olivia	2764761ddc6ce843b4badd542b742693	olivia@ucf.edu
16	Denise	c4802856e4fa3f0a980605b5fce91970	denise@ucf.edu
17	Anthony	de4e6ec2d2ab2e0907aab4145559ee5d	anthony@ucf.edu
18	Geoff	66e764e4ea6e3a7535d256e356ef84f7	geoff@ucf.edu
19	Jesuani	4b4efa8b787d904a755f3f38bd934c6c	jesuani@ucf.edu
20	James	5721c634b6c516e6b87417f1ca775be5	james@ucf.edu

**Figure 22: Password Encryption**

Sources	Network	Performance	Memory	Application	Security	Lighthouse
Filter						
Name	Value	Domain	Path	Expires ...	Size	Ht... Se...
userid	1,userName=test	localhost	/	2022-04...	21	

**Figure 23: Website Cookie**



**Password**

Test1

**Confirm Password**

Confirm Password

☒ Show Passwords

**Recomended that your password contain the following:**

- ✓ A lowercase letter
- ✓ A capital (uppercase) letter
- ✓ A number
- ✗ Minimum 8 characters

**Figure 24: Password Requirements Screenshot**

## Obstacles Faced & Overcame

The first issue we faced as a group was a steep learning curve getting to know the WAMP/LAMP stack and the languages necessary for creating this application (JavaScript, HTML, PHP, etc.). Fortunately, everyone in the group had some prior experience with some part of the stack so it only took us about a week to get into the flow of things. The next obstacle we faced was learning to use the SQL commands that we learned in class to efficiently extract and append data to our database. For example, to get the rating of an event, rather than sending all the individual ratings an event received to the frontend and manually calculating it, we made use of SQLs AVG() function. This saved us a lot of time and space and made our application more efficient.

Our largest issue occurred when connecting the frontend JavaScript and HTML to the backend API and PHP. We had to ensure that variable names matched, and the expected JSON payloads and responses were all correct otherwise we would run into some issues. Most of the time spent on this project was spent paired-programming and debugging issues that came up when trying to connect the frontend to the backend. Of course, after we got into the flow of things, this process became much quicker and smoother.



## **Demo**

Please follow this link to be transferred to our demo video: <https://youtu.be/DhMkjaE1F8k>

## **Conclusion**

Throughout the construction and editing the College Event Website we've learned a number of database skills. We got multiple desired features into our design, such as password requirements, encrypted passwords, as well as cookies.

If we had additional time on this assignment the features, we would have liked to add would be the ability to display images for the events in addition to rating and commenting on an event.

Another cool feature to add would be to attach the feed of the universities' social media pages. (Instagram, TikTok, Twitter, etc.)