Database Project Report

By,
Isaac Subrahmanyam,
Kenaniah Subrahmanyam,
Travis Wise.

Table of Contents

1. Project Title and Group Information 2
2. Project Description 2
3. GUI 5
    3.1 Menu 6
    3.2 Create RSO 6
    3.4 Create Event 7
    3.5 View Events 7
4. ER-Model 8
5. Relational Data Model 8
6. Sample Data 12
7. SQL Code 13
7.1 Create a New RSO 13
7.2 Insert a New Student into an Existing RSO 13
7.3 Create a New Event 14
7.4 Insert/Update a Comment for Events 15
7.5 Display Events for Event Feed Page 18
7.6 SQL Statements of Interests (Get All Viewable Events) 20
8. Advanced Features 21
    8.1 User Settings 21
    8.2 Prepared Statements and Security 22
    8.3 Information Encryption 22
    8.4 Chatroom 23
9. Constraint Enforcement 24
10. Conclusions/Observations 27

1 Project Title and Group Information

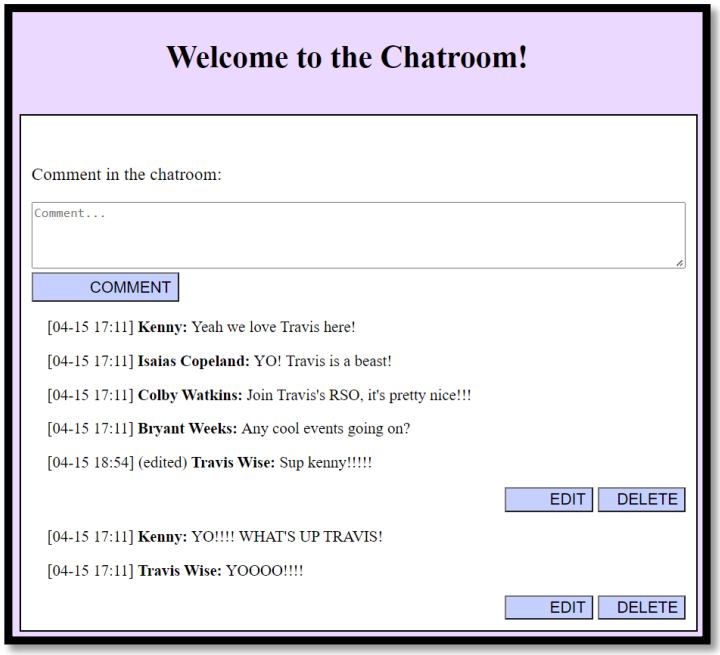
Project title: Club Event Organizer

Course information: COP4710-22 Spring 0001

Group members: Isaac Subrahmanyam, Kenaniah Subrahmanyam, Travis Wise

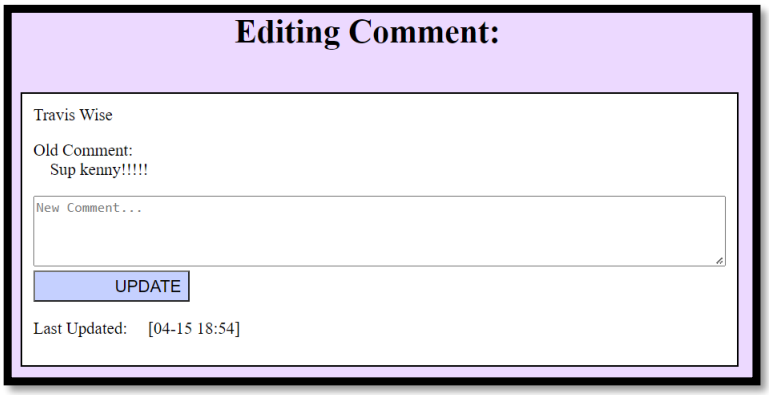
2 Project Description

In addition to the original project description, we implemented a user chatroom, and user settings. The chatroom can be used by anyone that is logged in so that anyone on the website can talk to each other through the chatroom. The user settings are used to change the user's username, password, and display mode for the website (light mode or dark mode). Pictures of both are below:

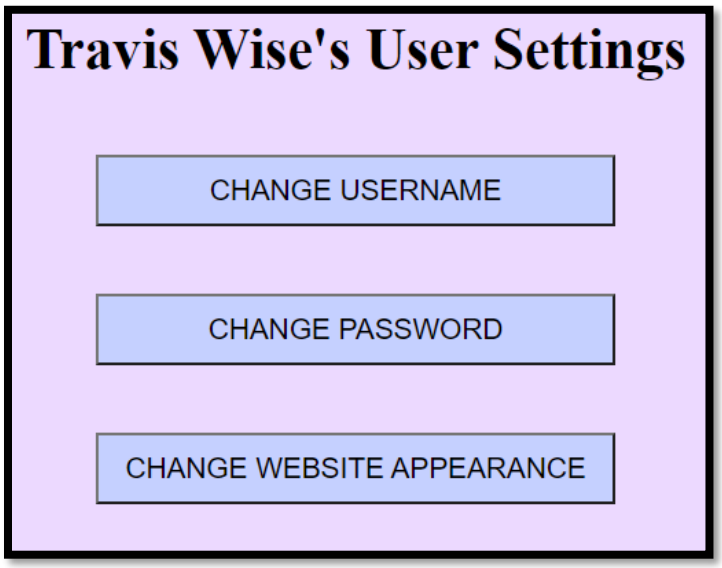


The chatroom displays, all comments made by users, they time they were last updated, and the name of the user that commented. Users can edit and delete their comments and Super Admins can delete any users comment which we will use as a type of moderation.

The chatroom comment editing screen is below:



Here we see the user settings page, which we will talk about in more detail later in this document, but we will briefly discuss the page here. There are three buttons, "Change Username", "Change Password", and "Change Website Appearance". The three buttons are self-explanatory and was implemented to give the user a better experience on the application. The website has two appearances, light and dark, and the current picture is in the default, light, mode. Two examples of dark mode is shown below, which one is a picture of the chatroom when the user changes the appearance to the dark theme and the other is the top of the user's profile page.



Welcome to the Chatroom!

Comment in the chatroom:

Comment...

COMMENT

[04-15 17:11] **Kenny:** Yeah we love Travis here!

[04-15 17:11] **Isaias Copeland:** YO! Travis is a beast!

[04-15 17:11] **Colby Watkins:** Join Travis's RSO, it's pretty nice!!!

[04-15 17:11] **Bryant Weeks:** Any cool events going on?

[04-15 18:54] (edited) **Travis Wise:** Sup kenny!!!!!

EDIT DELETE

[04-15 17:11] **Kenny:** YO!!!! WHAT'S UP TRAVIS!

[04-15 17:11] **Travis Wise:** YOOOO!!!!

EDIT DELETE

Club Event Organizer

Travis Wise's Profile Page

Email: Travis@knights.ucf.edu
University: University of Central Florida
User Status: Student

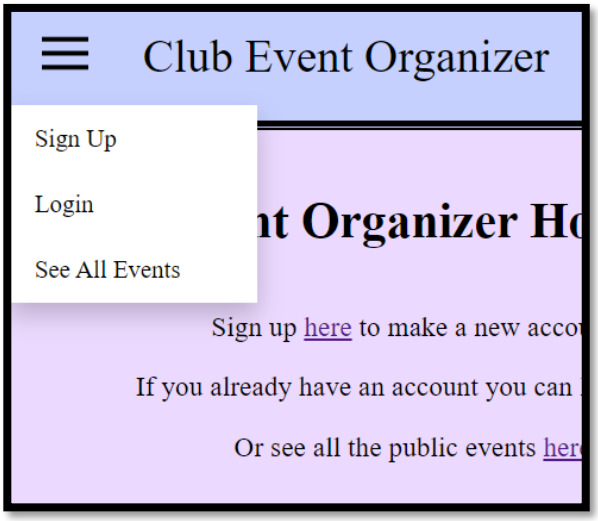
3 GUI

Framework: Windows, Apache, MySql, and PHP (WAMP)

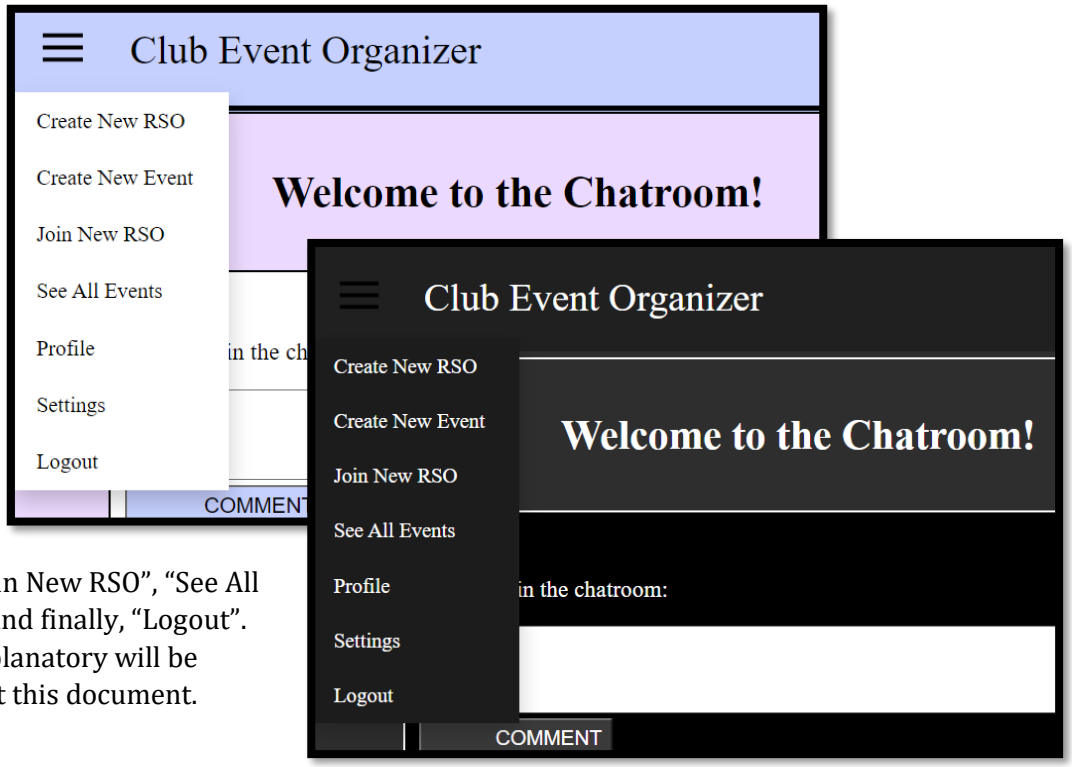
Languages used: HTML, CSS, PHP, and JS

3.1 Menu

When the user enters the website for the first time they can click on the button in the top left of their screen and the menu in the picture to the right will appear. The user can use this menu to click on the three buttons "Sign Up", "Login", and "See All Events". The "Sign Up" button will take the user to the sign-up page, similarly the "Login" button will take the user to the login-page. "See All Events" will take you to the events page. The events page will display all events that the user has access to, since the user is not logged in, the user will only be able to see events that are public. To see private events that pertain to the user's University, the user must sign-up and login with their University and go the events page once again.

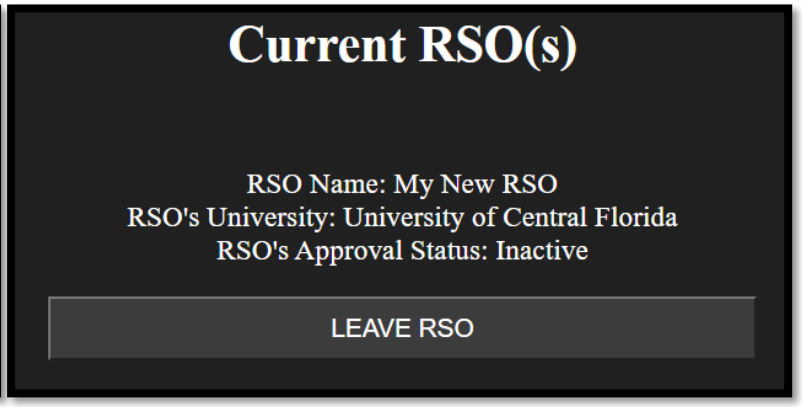


When the user is logged in, they will see the new menu that is displayed to the right (light and dark mode is showed, throughout most of this document we will stick to displaying dark mode). This menu has the follow options; "Create New RSO", "Create New Event", "Join New RSO", "See All Events", "Profile", "Settings", and finally, "Logout". Any button that is not self-explanatory will be described in detail throughout this document.



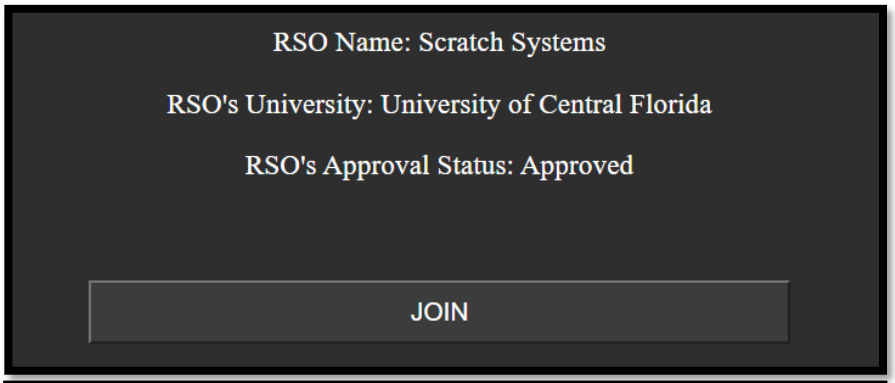
3.2 Create RSO

Below, on the left, we can see the form to create a new RSO, and, on the right, we can see the display that shows the RSO you are in, it's name, university, and approval status. In the next section we will talk more about joining RSOs and all that it entails.



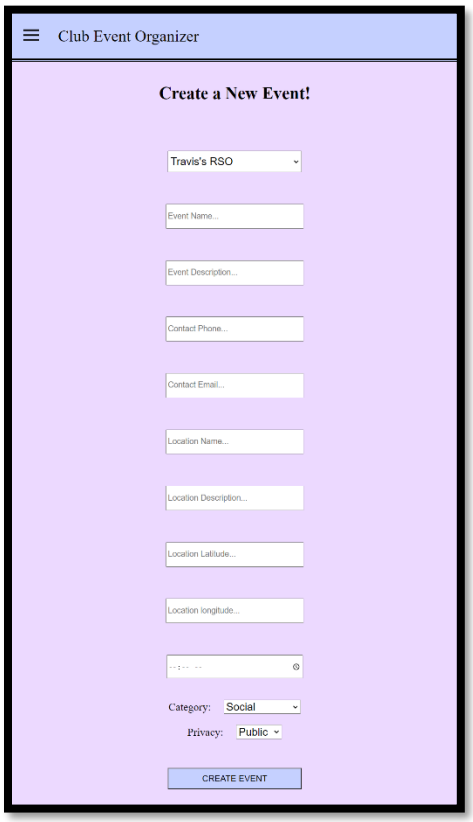
3.3 View/Join RSOs

The image on the right shows the RSOs that I am currently able to join based on my university. Upon joining an RSO you will be able to see the RSO in your profile page under "Current RSO(s)". As shown in the image below you have the option to leave any RSO you are in by accessing it through this same page. Both of these also display the RSO Name, University and whether the RSO has enough members (5 as per the assignment description) to be active/approved. Once 5 members join an RSO the RSO's approval status turns to "Approved" and the RSO's admin can start creating events.



3.4 Create Event

To the right, we display the Create a New Event page which can be accessed through the "Create New Event" button in the user's menu.



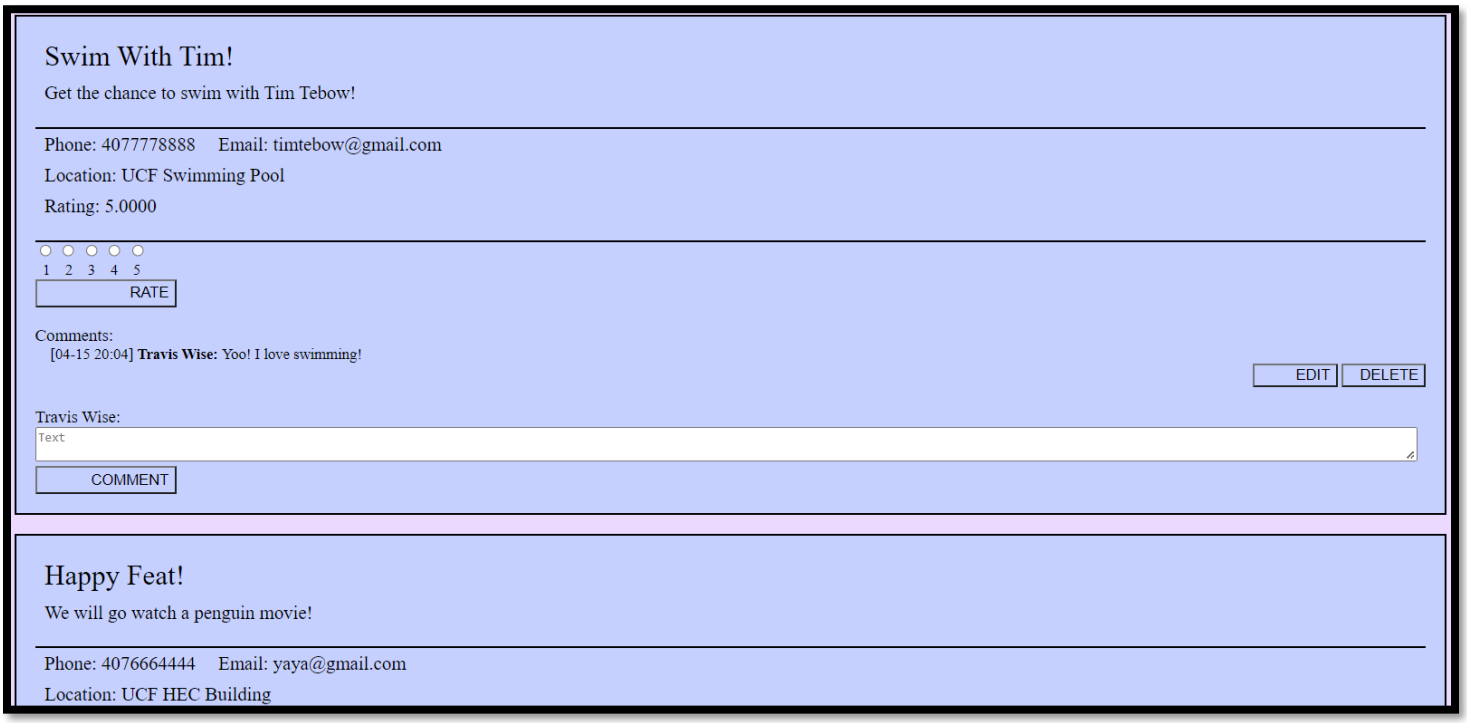
Users can enter the following information; the name of the RSO, the name of the event, the event description, the contact phone and email (which if left blank the event will pull the RSO's admin's phone and email), the location name, the location description, the location's latitude, longitude, and the time of the event (for this we assume events will last roughly an hour at a time), the category of the event, and finally, the privacy of the event.

Note: the dropdowns for categories and RSOs pull data from the database to populate the <option> fields so they will always be up to date.

3.5 View Events

Below, we show some of the event feed which is obviously too big to share 100% of, but as you can see in the picture below, users can:

- View events that they have access to (public, private when they are in the same university, and RSO when they are in the same RSO).
- Rate events that they can view (as well as update their rating which is shown in the demo video).
- Comment on events they can view as well as edit and delete their comments.



4 ER-Model

See attached ERD, filename: ERD.pdf

5 Relational Data Model

```
CREATE TABLE `University` (
  `ID` int NOT NULL AUTO_INCREMENT,
  `Name` varchar(255) NOT NULL,
  `GmailAt` varchar(255) NOT NULL,
  `DataTimeCreated` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `DataTimeUpdated` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`ID`)
);

CREATE TABLE `Users` (
  `ID` int NOT NULL AUTO_INCREMENT,
  `UniversityID` int NOT NULL REFERENCES University(ID),
CONSTRAINT FK_User_Unvi FOREIGN KEY (UniversityID) REFERENCES University(ID),
  `Super` boolean NOT NULL DEFAULT  0,
  `Name` varchar(255)  NOT NULL,
  `Gmail` varchar(255) NOT NULL,
  `Phone` varchar(255) NOT NULL,
  `Password` varchar(255),
  `ColorPreferences` varchar(255) NOT NULL DEFAULT 0,
  `DataTimeCreated` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `DataTimeUpdated` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`ID`),
  FOREIGN KEY (`UniversityID`) REFERENCES `University`(`ID`)
);

CREATE TABLE `RSO` (
  `ID` int NOT NULL AUTO_INCREMENT,
```

```
  `UniversityID` int NOT NULL REFERENCES University(ID),
  `OwnerID` int NOT NULL REFERENCES Users(ID),
  `Status` boolean NOT NULL DEFAULT 0,
  `Name` varchar(255) NOT NULL,
  `DataTimeCreated` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `DataTimeUpdated` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`ID`),
  FOREIGN KEY (`UniversityID`) REFERENCES `University`(`ID`)
);

CREATE TABLE `Registered` (
  `ID` int NOT NULL AUTO_INCREMENT,
  `RSOID` int NOT NULL REFERENCES Users(ID),
  `UserID` int NOT NULL REFERENCES Users(ID),
  `DataTimeCreated` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `DataTimeUpdated` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`ID`),
  FOREIGN KEY (`UserID`) REFERENCES `Users`(`ID`),
  FOREIGN KEY (`RSOID`) REFERENCES `RSO`(`ID`)
);

CREATE TABLE `Location` (
  `ID` int NOT NULL AUTO_INCREMENT,
  `Name` varchar(255) NOT NULL DEFAULT '',
  `Description` text,
  `Longitude` int NOT NULL DEFAULT 0,
  `Latitude` int NOT NULL DEFAULT 0,
  `DataTimeCreated` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `DataTimeUpdated` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`ID`)
```

```
);

CREATE TABLE `Categories` (
  `ID` int NOT NULL AUTO_INCREMENT,
  `Name` varchar(255) NOT NULL,
  PRIMARY KEY (`ID`)
);

CREATE TABLE `Events` (
  `ID` int NOT NULL AUTO_INCREMENT,
  `LocationID` int NOT NULL REFERENCES Location(ID),
  `EventCat` int NOT NULL REFERENCES Categories(ID),
  `ForeignID` int,
CONSTRAINT FK_Events_Loc FOREIGN KEY (LocationID) REFERENCES Location(ID),
CONSTRAINT FK_Events_Eve FOREIGN KEY (EventCat) REFERENCES Categories(ID),
  `Name` varchar(255) NOT NULL DEFAULT '',
  `Description` varchar(255) NOT NULL DEFAULT '',
  `Privacy` int NOT NULL DEFAULT 0,
  `ContactPhone` varchar(255) NOT NULL,
  `ContactEmail` varchar(255) NOT NULL,
  `DataTimeCreated` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `DataTimeUpdated` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`ID`),
  FOREIGN KEY (`LocationID`) REFERENCES `Location`(`ID`),
  FOREIGN KEY (`EventCat`) REFERENCES `Categories`(`ID`),
  FOREIGN KEY (`ForeignID`) REFERENCES `RSO`(`ID`)
);

CREATE TABLE `Ratings` (
  `ID` int NOT NULL AUTO_INCREMENT,
```

```
  `EventID` int NOT NULL REFERENCES Events(ID),
  `UserID` int NOT NULL REFERENCES Users(ID),
CONSTRAINT FK_Rat_Eve FOREIGN KEY (EventID) REFERENCES Events(ID),
CONSTRAINT FK_Rat_User FOREIGN KEY (UserID) REFERENCES Users(ID),
  `Rating` int,
  `DataTimeCreated` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `DataTimeUpdated` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`ID`),
  FOREIGN KEY (`UserID`) REFERENCES `Users`(`ID`),
  FOREIGN KEY (`EventID`) REFERENCES `Events`(`EventCat`)
);

CREATE TABLE `Comments` (
  `ID` Int NOT NULL AUTO_INCREMENT,
  `EventID` int NOT NULL REFERENCES Events(ID),
  `UserID` int NOT NULL REFERENCES Users(ID),
CONSTRAINT FK_Com_Eve FOREIGN KEY (EventID) REFERENCES Events(ID),
CONSTRAINT FK_Com_User FOREIGN KEY (UserID) REFERENCES Users(ID),
  `Text` text,
  `DataTimeCreated` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `DataTimeUpdated` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`ID`),
  FOREIGN KEY (`EventID`) REFERENCES `Events`(`EventCat`),
  FOREIGN KEY (`UserID`) REFERENCES `Users`(`ID`)
);

CREATE TABLE `ChatroomComments` (
  `ID` int NOT NULL AUTO_INCREMENT,
  `UserID` int NOT NULL REFERENCES Users(ID),
  `Comment` text NOT NULL,
```

```
  `DataTimeCreated` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `DataTimeUpdated` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`ID`),
  FOREIGN KEY (`UserID`) REFERENCES `Users`(`ID`)
);
```

6 Sample Data

See attached file app.sql for the SQL creation code and the sample data.

7 SQL Examples

7.1 Create a New RSO

```
function createRSO($UniversityID, $OwnerID, $Name)
{
    $key = encryptionKey();
    $conn = connectToDatabase();
    $sql = "INSERT INTO RSO(UniversityID, OwnerID, Name) VALUES (?, ?, ?);";

    $stmt = $conn->prepare($sql);

    if(!$stmt)
    {
        echo "Prepared statement failed";
        exit();
    }


    $Name_enc = encryptthis($Name, $key);

    $stmt->bind_param("iis", $UniversityID, $OwnerID, $Name_enc);
    $stmt->execute();

    $RSOID = $conn->insert_id;

    registerMember($RSOID, $OwnerID);
}
```

7.2 Insert a New Student into an Existing RSO

```
function registerMember($RSOID, $MemberID)
{
    if(isRegistered($RSOID, $MemberID)) return true;

    $conn = connectToDatabase();
    $sql = "INSERT INTO Registered (RSOID, UserID) VALUES ($RSOID, $MemberID);";
    $result = mysqli_query($conn, $sql);

    updateRSOStatus($RSOID);
    // Return success boolean
    if($result) return True;
    return false;
}
```

7.3 Create a New Event

```
function createEvent($EventName, $EventDescription, $EventCategory, $EventPrivacy,
$ContactPhone, $ContactEmail, $EventLocationName, $EventLocationDescription, $UserID, $RSOID,
$long, $lat, $Time)
{
    $key = encryptionKey();
    $conn = connectToDatabase();

    if (locationExists($long, $lat, $Time)) {
        header("location: ../createEvent.php?error=LocationAndOrTimeOverlap");
    }

    // Add the location to the database and use it's Id top populate $EventLocationID
    // Then insert the values into the event database
    $sql = "INSERT INTO `Location` (`Name`, `Description`, `Longitude`, `Latitude`) VALUES
(?, ?, ?, ?);";
    $stmt = $conn->prepare($sql);
    if(!$stmt) {
        echo "Location Insert Failed: " . mysqli_error($conn);
        exit();
    }

    $EventLocationName_enc = encryptthis($EventLocationName, $key);
    $EventLocationDescription_enc = encryptthis($EventLocationDescription, $key);

    $stmt->bind_param("ssii", $EventLocationName_enc, $EventLocationDescription_enc, $long,
$lat);
    $stmt->execute();
    $stmt->get_result();

    // Get the location id of the location we just inserted
    $EventLocationID = $conn->insert_id;

    // Get the ForeignID
    // If the privacy is 0 or 1 set the ForeignID to the University
    // Else set the ForeignID to the RSO
    if ($EventPrivacy == 0 || $EventPrivacy == 1) {
        $ForeignID = getUserUniversity($UserID);
    }
    else if($RSOID != 0 && isOwner($RSOID ,$UserID)){
        $ForeignID = $RSOID;
    }else if($RSOID == 0)
    {
        header("location: ../createEvent.php?error=An RSO Must be Selected");
        return;
    }
```

```
    else
    {
        header("location: ../createEvent.php?error=Not Owner of RSO");
        return;
    }

    $sql = "INSERT INTO Events (`LocationID`, `EventCat`, `ForeignID`, `Name`, `Description`,
`Privacy`, `ContactPhone`, `ContactEmail`, `Time`)  VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?);";
    $stmt = $conn->prepare($sql);
    if(!$stmt) {
        echo "Prepared statement failed 2";
        exit();
    }

    $EventName_enc = encryptthis($EventName, $key);
    $EventDescription_enc = encryptthis($EventDescription, $key);
    $ContactPhone_enc = encryptthis($ContactPhone, $key);
    $ContactEmail_enc = encryptthis($ContactEmail, $key);


    $stmt->bind_param("iiississs", $EventLocationID, $EventCategory, $ForeignID,
$EventName_enc, $EventDescription_enc, $EventPrivacy, $ContactPhone_enc, $ContactEmail_enc,
$Time);
    $stmt->execute();
    $stmt->get_result();
}
```

7.4 Insert/Update a Comment for Events

Insert:

```
function comment($EventID, $UserID, $Comment)
{
    $key = encryptionKey();
    if(empty($Comment)) return;

    $conn = connectToDatabase();
    $sql = "INSERT INTO Comments (EventID, UserID, Text) VALUES (?, ?, ?)";

    // Execute prepared statement
    $stmt = $conn->prepare($sql);

    if(!$stmt)
    {
        echo "Prepared statement failed";
        exit();
    }
```

```
    $Comment_enc = encryptthis($Comment, $key);

    $stmt->bind_param("iis", $EventID, $UserID, $Comment_enc);
    $stmt->execute();
    $result = $stmt->get_result();

    // Return success boolean
    if($result) return True;
    return false;
}
```

Update:

```
function updateEventComment($CommentID, $NewComment) {
    $conn = connectToDatabase();
    $Date = date('Y-m-d H:i:s');

    $key = encryptionKey();
    $NewComment_enc = encryptthis($NewComment, $key);

    $sql = "UPDATE Comments SET `Text` = '$NewComment_enc', DataTimeUpdated = '$Date'  WHERE
ID = $CommentID";
    $result = mysqli_query($conn, $sql);
    // Return success boolean
    if(!$result) {
        echo mysqli_error($conn);
    }
}
```

Delete:

```
function deleteEventComment($CommentID) {
    $conn = connectToDatabase();
    $sql = "DELETE FROM comments WHERE ID = $CommentID;";
    $result = mysqli_query($conn, $sql);
    if (!$result) {
        echo mysqli_error($conn);
        exit();
    }
    else {
        header("location: ../events.php");
    }
}
```

Display:

```
function displayEventCommentEditing($EventID, $UserID, $CommentID) {
    $key = encryptionKey();
```

```
    $conn = connectToDatabase();
    $sql = "SELECT * FROM Comments WHERE ID = $CommentID;";
    $result = mysqli_query($conn, $sql);

    if($result)
    {
        $EventInformation = EventInfo($EventID);
        $key = encryptionKey();
        $EventName = decryptthis($EventInformation["Name"], $key);
        $EventDescription =  decryptthis($EventInformation["Description"], $key);

        echo '
        <h2 class="pageTitle">Editing Comment:</h2>
        <div class="chatroom_outer">
        <div class="chatroom">

        <div class="inner_inner_event">
            <span class="event_desc eventName">'. $EventName .'</span> <br>
            <span class="event_desc">'. $EventDescription . '</span><br><br>
        </div>
        ';


        $row = mysqli_fetch_assoc($result);
        $UserInfo = getUserInfoById($row["UserID"]);
        $UserName = decryptthis($UserInfo["Name"], $key);
        echo '<p> '. $UserName .'\'s Old Comment:<br> ' . decryptthis($row["Text"],
$key) . '</p>';
        echo '
        <div class="editingComment">
        <form action="api/eventCommentEdit.php" method="POST">
            <input type="hidden" name="CommentID" value='.$row["ID"].'>
            <textarea class="CommentEntry" name="NewComment" rows="4" cols="20"
placeholder="New Comment..."></textarea><br>
            <button class="commentSubmit" type="submit" name="submit">Update</button>
        </form>
        </div>
        ';
        $date = new DateTime($row['DataTimeUpdated']);
        echo "<p>Last Updated:  [" . $date->format('m-d H:i') . "]</p>";

        echo '</div></div>';
    }
    else {
        echo mysqli_error($conn);
    }
```

```
}
```

7.5 Display Events for Event Feed Page

Get all viewable events:

```
function showEvents($UserID)
{
    $conn = connectToDatabase();
    $sql = "SELECT E.ID FROM Events E, Users U  WHERE U.ID = ? AND
        ((U.Super =  1) OR
        (E.Privacy = 0) OR
        (E.Privacy = 1 AND EXISTS (SELECT O.ID FROM University O WHERE E.ForeignID = O.ID AND
U.UniversityID = O.ID)) OR
        (E.Privacy = 2 AND EXISTS (SELECT R.ID FROM Registered R WHERE R.UserID = U.ID AND
R.RSOID = E.ForeignID)));";
    $stmt = $conn->prepare($sql);

    if(!$stmt)
    {
        echo "Prepared statement failed";
        exit();
    }

    $stmt->bind_param("i", $UserID);
    $stmt->execute();
    $result = $stmt->get_result();
    $resultCheck = mysqli_num_rows($result);

    if($resultCheck > 0)
    {
        while($row = mysqli_fetch_assoc($result))
        {
            FormatEvent($row['ID'], $UserID);
        }
    }

}
```

Format Event:

```
function FormatEvent($EventID, $UserID)
{
    $key = encryptionKey();
```

```
    $info = EventInfo($EventID);
    echo '
        <div class="event" style="text-align: left;">
            <div class="inner_event">
                <div class="inner_inner_event">
                    <span class="event_desc eventName">'. decryptthis($info["Name"], $key)
.'</span> <br>
                    <span class="event_desc">'. decryptthis($info["Description"], $key)
.'</span><br><br>
                </div>
                <span class="event_desc">Phone: '. decryptthis($info["ContactPhone"], $key)
.'</span>
                <span class="event_desc">Email: '. decryptthis($info["ContactEmail"], $key)
.'</span><br>
                <span class="event_desc">Location: '.
getLocationNameByLocationID($info["LocationID"]) .'</span><br>
                <span class="event_desc" >Rating: '. rating($EventID) .'</span><br><br>
            </div>
    ';

    if($UserID != 0)
    {
        echo '
            <form action="api/rate.php" method="POST">
                <input type="hidden" name="EventID" value='.$EventID.'>
                <input type="radio" name="rating" value=1>
                <input type="radio" name="rating" value=2>
                <input type="radio" name="rating" value=3>
                <input type="radio" name="rating" value=4>
                <input type="radio" name="rating" value=5>
                <br>
                <p> 1 2 3 4 5 <p><br>
                <button class="commentSubmit" type="submit" name="submit">Rate</button>
            </form>';
    }

    echo '<br><p class="desc">Comments:<br></p>';

    echo getComments($EventID, $UserID);

    if($UserID != 0)
    {
        $UserInfo = getUserInfoById($UserID);
        $UserName = decryptthis($UserInfo["Name"], $key);
        echo '
            <form action="api/Comment.php" method="POST">
```

```
                <input type="hidden" name="EventID" value='. $EventID .'>
                <p class="desc"><br>'. $UserName .':</p>
                <textarea class="CommentEntry" name="Comment" rows="2" cols="20"
placeholder="Text"></textarea>
                <br>
                <button class="commentSubmit" type="submit" name="submit">Comment</button>
            </form>
        ';
    }

    echo '
    </div>';
}
```

7.6 SQL Statements of Interest (Get All Viewable Events)

```
"SELECT E.ID FROM Events E, Users U  WHERE U.ID = ? AND

((U.Super =  1) OR

(E.Privacy = 0) OR

(E.Privacy = 1 AND EXISTS

(SELECT O.ID FROM University O WHERE E.ForeignID = O.ID AND U.UniversityID = O.ID)) OR

(E.Privacy = 2 AND EXISTS (SELECT R.ID FROM Registered R WHERE R.UserID = U.ID AND R.RSOID =

E.ForeignID)));";
```

8 Advanced Features

8.1 User Settings

As described above on page 2, we implemented user settings, which was outside the scope of the project assignment in an effort to improve the user experience and implement more advanced features. Images of this are shown below:

8.2 Prepared Statements and Sessions Security

Our group implemented prepared statements everywhere users can input into the database. This is to prevent SQL injections. It is one form of security. All these are shown in the SQL statement section above but for an example we will show the function insertChatroomComment() which also uses prepared statements:

```
// Insert the $Comment passed in into the chatroom comments table
// with the $UserID that was passed in
function insertChatroomComment($UserID, $Comment)
{
    $key = encryptionKey();
    $conn = connectToDatabase();
    $sql = "INSERT INTO ChatroomComments (UserID, Comment) VALUES (?, ?)";

    // Execute prepared statement
    $stmt = $conn->prepare($sql);

    if(!$stmt)
    {
        echo "Prepared statement failed";
        exit();
    }

    $Comment_enc = encryptthis($Comment, $key);

    $stmt->bind_param("is", $UserID, $Comment_enc);
    $stmt->execute();
    $result = $stmt->get_result();

    // Return success boolean
    if($result) return True;
    return false;
}
```

8.3 Information Encryption

To protect user information, we encrypt all personal information and non-public information. As can be seen in the figure below.

```
mysql> select * from rso;
+----+--------------+---------+--------+-----------------------------------------------------------------------------------------+---------------------+---------------------+
| ID | UniversityID | OwnerID | Status | Name                                                                                    | DataTimeCreated     | DataTimeUpdated     |
+----+--------------+---------+--------+-----------------------------------------------------------------------------------------+---------------------+---------------------+
|  1 |            1 |       1 |      1 | OHlXdlhlUmQ5M01Pdyt0L3dqQUkzUT09Ojpb3oxpKHTPSK+GXDWdvtSg                                | 2022-04-14 00:49:31 | 2022-04-14 00:49:31 |
|  2 |            1 |       2 |      1 | VlRJV3FyMHdtQlhrYXlxOFAreG9TbHUxZ3Vza1hDMVRuQ3lKbnJXM1hvaz06OuWac65GdSwbcL0szw8zmUE= | 2022-04-14 00:49:31 | 2022-04-14 00:49:31 |
|  3 |            2 |       5 |      0 | a2xCeU9DN25uU1pUTjA3Z3NiQW5CQT09Ojr9W/kZv/OScThcXViOeWFg                                | 2022-04-14 00:49:31 | 2022-04-14 00:49:31 |
|  4 |            2 |       6 |      0 | Rm5xakFHTXlkQXdJRmpzQU1GTkNQUT09OjoAMpdlCNdXbD5HG+FJfnMT                                | 2022-04-14 00:49:31 | 2022-04-14 00:49:31 |
|  5 |            3 |       9 |      0 | Yi81cWQ0T2J3TEtvRUtlZHJUWHRRUT09OjpNyG9tmfrt5sxP2PGHPs+b                                | 2022-04-14 00:49:31 | 2022-04-14 00:49:31 |
|  6 |            3 |      10 |      0 | NGswZGNtVG8xV1VXRFdaa2FpeFRxdz09Ojrg4WKeQzByBjkGwax1KbIz                                | 2022-04-14 00:49:31 | 2022-04-14 00:49:31 |
|  7 |            4 |      13 |      0 | T3hIUmZJYi9mRUVFbVY4Rk0zTVNMdz09Ojpn1ltH7BTBvaqbttxo+nzR                                | 2022-04-14 00:49:31 | 2022-04-14 00:49:31 |
|  8 |            4 |      14 |      0 | S0cxMDExQks3cHc1cklCNUJrU2NQUjZxSC9STWF3TUZZNlFsbnhDM0ZaRT06Os31Dl5vLYBQnVxvKQdXcwI= | 2022-04-14 00:49:31 | 2022-04-14 00:49:31 |
|  9 |            5 |      17 |      0 | SkJXajQ2NDRqK2ZOd1ViRHE4MGdFdDN4dkdGS05uLys5MXh3dG03YjNyZz06Oiku+dA6LQzZyjVc98N+qhs= | 2022-04-14 00:49:32 | 2022-04-14 00:49:32 |
| 10 |            5 |      18 |      0 | cVZ2RVVxTXlSNVJCelRtbnhQVndLZz09OjqyvF9JJEoihZQIXpKcjYre                                | 2022-04-14 00:49:32 | 2022-04-14 00:49:32 |
| 11 |            6 |      21 |      0 | NEc5NUpNTHZOS1dlL3NycDFnNjhWSWpmVDBlUlRNS3FJV1IvandicTJEND06Ov/CoUZkIYDNWzSH/YrJvC8= | 2022-04-14 00:49:32 | 2022-04-14 00:49:32 |
| 12 |            6 |      22 |      0 | amJvQ1BCZE9YSmxCZHROMHhUR3dpL2I3TS9QUHlUQmRPK2xrOEc4NWVtVT06Om36VKaZfAdhPezmuM0Qbg0= | 2022-04-14 00:49:32 | 2022-04-14 00:49:32 |
| 13 |            7 |      25 |      0 | bmt0a0wvZ3RFbXFEUXUyL21hWVMyUT09OjrYAN1PImGBh/odygotwens                                | 2022-04-14 00:49:32 | 2022-04-14 00:49:32 |
| 14 |            7 |      26 |      0 | eWFaOCtmajkwVHh4TEluNnhMczUzUT09OjpwUvWI22hxR6BiJMH0IEz9                                | 2022-04-14 00:49:32 | 2022-04-14 00:49:32 |
| 15 |            8 |      29 |      0 | alFPcHBjMmNhdTFkK2NqeTdTTnJ3S0NmdEZOaUR2SFVodUxNcWZjOFRpND06OkdOe8BbvPONiTu1w+K7ot8= | 2022-04-14 00:49:32 | 2022-04-14 00:49:32 |
| 16 |            8 |      30 |      0 | RlJVV3VZRUZ5Z2dlSm9Wa1BTYjZCYmlNUUJoL0FqbTBRM2hITE0rV1drTT06OqWvFX9n1Ocdlm9bAFWpYOI= | 2022-04-14 00:49:32 | 2022-04-14 00:49:32 |
| 17 |            9 |      33 |      0 | UHhHcVY3NHd4RldFdjZtK2d0K1dEaU9XTkxrL245N0xlMmZKTXM2SjVaRT06Ov9FIvWgcCLjKxhHiOxIe78= | 2022-04-14 00:49:32 | 2022-04-14 00:49:32 |
| 18 |            9 |      34 |      0 | R2hLRlhpRFV1T1Uxa080bkQzSTRJVjAwVmhEYkFqeW1ZM3VMRFNodWtnST06OtxcUToZ1c4NiC+5fWITZMQ= | 2022-04-14 00:49:32 | 2022-04-14 00:49:32 |
| 19 |           10 |      37 |      0 | WmlIZnlERWRSOFJ3V2RSNnFKMmFkdz09OjqFCIN4eNO5i+BHr1B7v9ec                                | 2022-04-14 00:49:32 | 2022-04-14 00:49:32 |
| 20 |           10 |      38 |      0 | S0V4NmFBanBzUUpmQ1pqR1ZURjBEdz09OjrBImN9XZFHWfu5winyFE5A                                | 2022-04-14 00:49:32 | 2022-04-14 00:49:32 |
+----+--------------+---------+--------+-----------------------------------------------------------------------------------------+---------------------+---------------------+
```

Here is the following code for how we approached encryption:

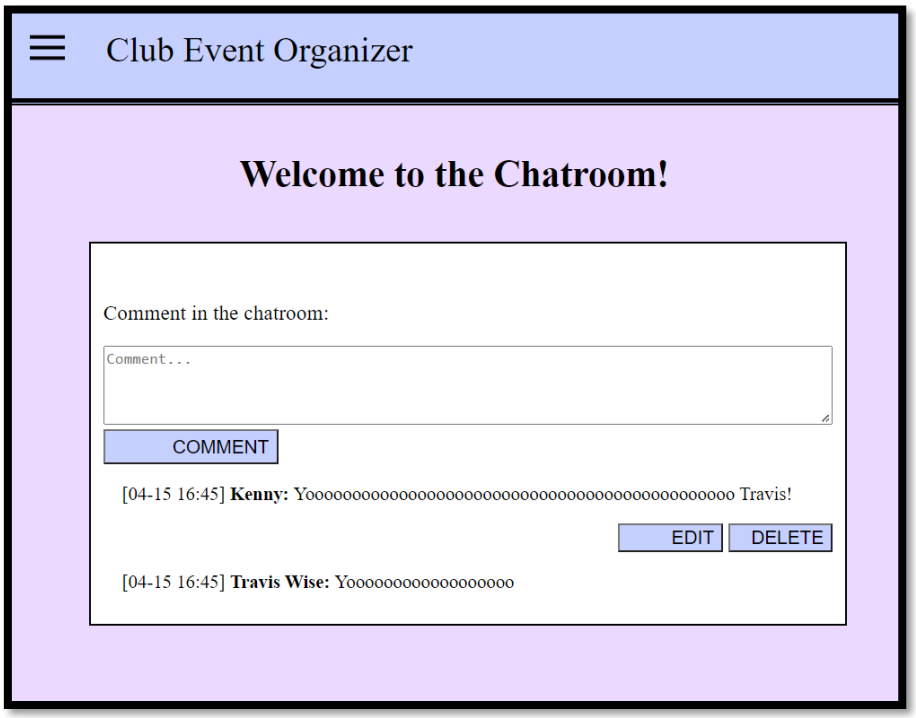
```
function encryptionKey()
{
    return 'qkwjdiw239&&jdafweihbrhnan&^%$ggdnawhd4njshjwuuO';
}

//ENCRYPT FUNCTION
function encryptthis($data, $key)
{
    $encryption_key = base64_decode($key);
    $iv = openssl_random_pseudo_bytes(openssl_cipher_iv_length('aes-256-cbc'));
    $encrypted = openssl_encrypt($data, 'aes-256-cbc', $encryption_key, 0, $iv);
    return base64_encode($encrypted . '::' . $iv);
}

//DECRYPT FUNCTION
function decryptthis($data, $key)
{
    $encryption_key = base64_decode($key);
    list($encrypted_data, $iv) = array_pad(explode('::', base64_decode($data), 2),2,null);
    return openssl_decrypt($encrypted_data, 'aes-256-cbc', $encryption_key, 0, $iv);
}
```

8.4 Chatroom

As described above, we implemented a chatroom feature to improve the user's experience and add more advanced features to the application. The image to the right shows the chatroom in the application where the users "Travis Wise" and "Kenny" are communicating with each other. Like with the comments for events, the user can add, edit, and delete their comments in the chatroom. When edited, the keyword "(edited)" shows up next to the comment and the comment display the time the comment was updated.



9 Constraint Enforcement

Owner constraint:

Member count constraint:

After joining as the 5th member of the RSO:

After leaving again:

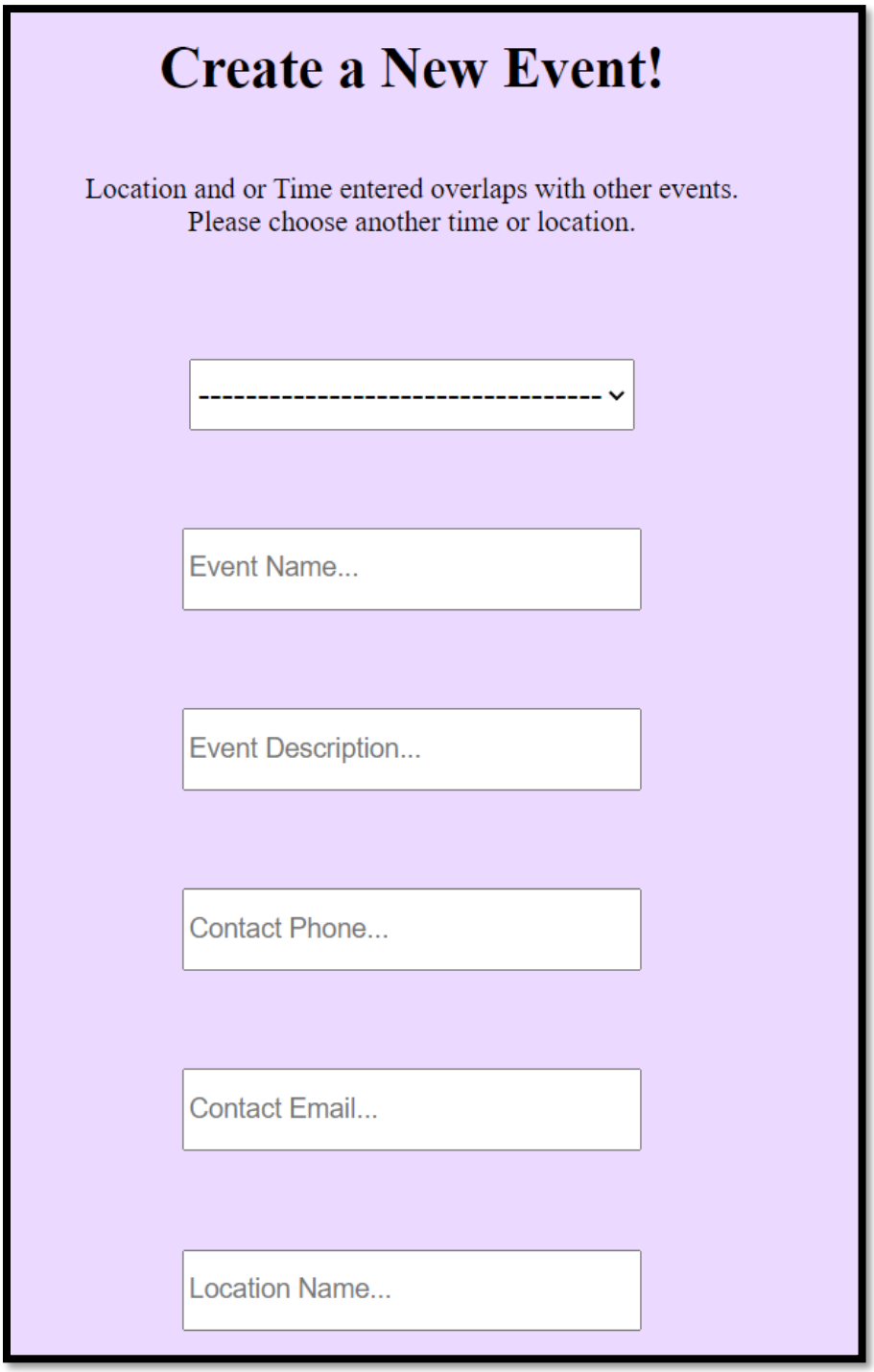
RSO Name: Testing RSO

RSO's University: Florida State University

RSO's Approval Status: Inactive

JOIN

Location/Time Overlap Constraint:



10 Conclusion/Observations

Overall the project was a fun and challenging project that made me increase my skills in backend and frontend web development, I am happy I took this class and excited to use the skills I learn while making this project on future projects. The only part of the project I would have liked to improve is the project description as it is very difficult to read and understand. The project description could easily be simplified and explained in a clearer and more concise way, as well as giving some examples in the project description as that would further increase understanding.