

Campus Connect 360

Database Systems
COP 4710 - Spring 2024
Group 88 - David Umanzor

Table of Contents

● Project Description	2
● GUI	2
● ER-Model	5
● Schema	5
● Table Sample	9
● SQL Examples	10
● Constraint Enforcement	17
● Advanced Features	22
● Conclusion	23

Project Description

Campus Connect 360 is a web application to connect students within their own campus and outside to other universities. It lets them stay connected to events and search for things happening nearby or at their own university.

GUI

Platform:

- Web Application accessible through web browsers such as Chrome, Firefox and Safari

Languages:

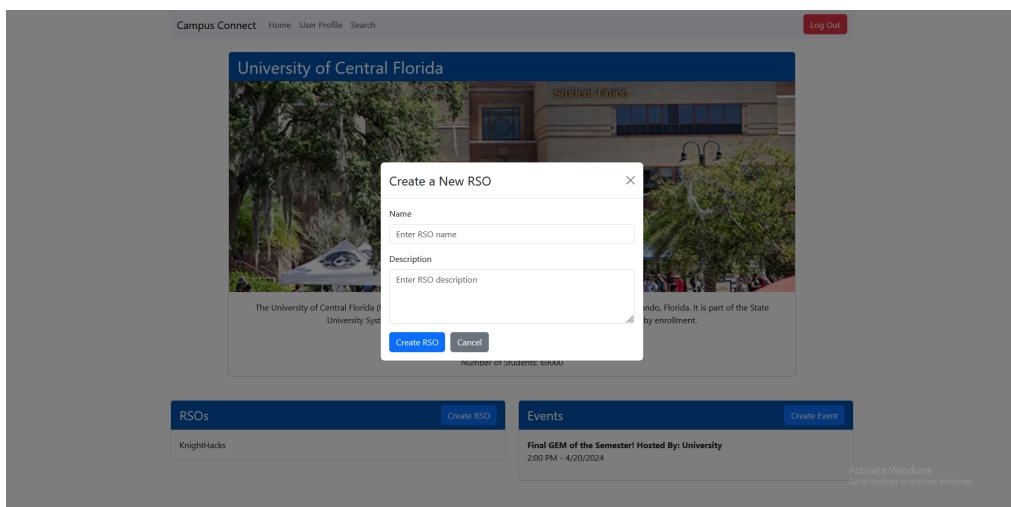
- Frontend: JavaScript (ES6+), utilizing the React framework for dynamic component-based architecture.
- CSS3 and Bootstrap for styling and responsive design to ensure a consistent look and feel across various devices and screen sizes.
- Backend: Node.js with Express.js framework to manage server-side logic and HTTP requests

DBMS (Database Management System):

- PostgreSQL, chosen for its robustness, reliability, and support for advanced data types and functionality.
 - Hosted by Heroku.

Screen Shots:

Create RSO:



Create Event:

Campus Connect Home User Profile Search Log Out

Create a New Event

Name
Enter event name

Description
Enter event description

Category
Enter event category

Date
April 17, 2024

Time
10 :00 AM

Contact Phone
Enter contact phone

Contact Email
Enter contact email

Location Name
Enter location name

Visibility
RSO

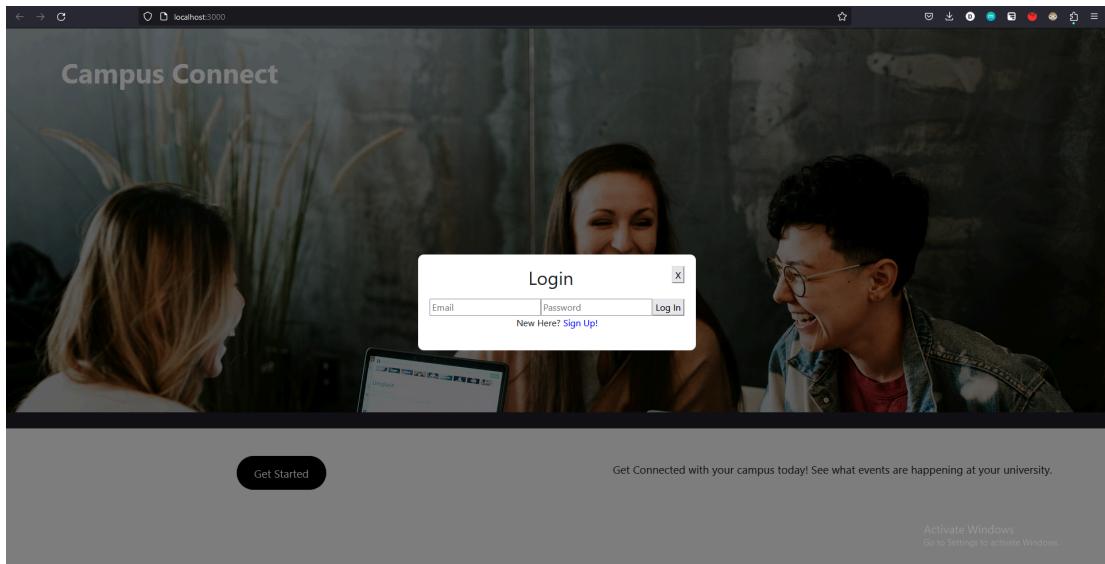
Event Location

Create Event Cancel

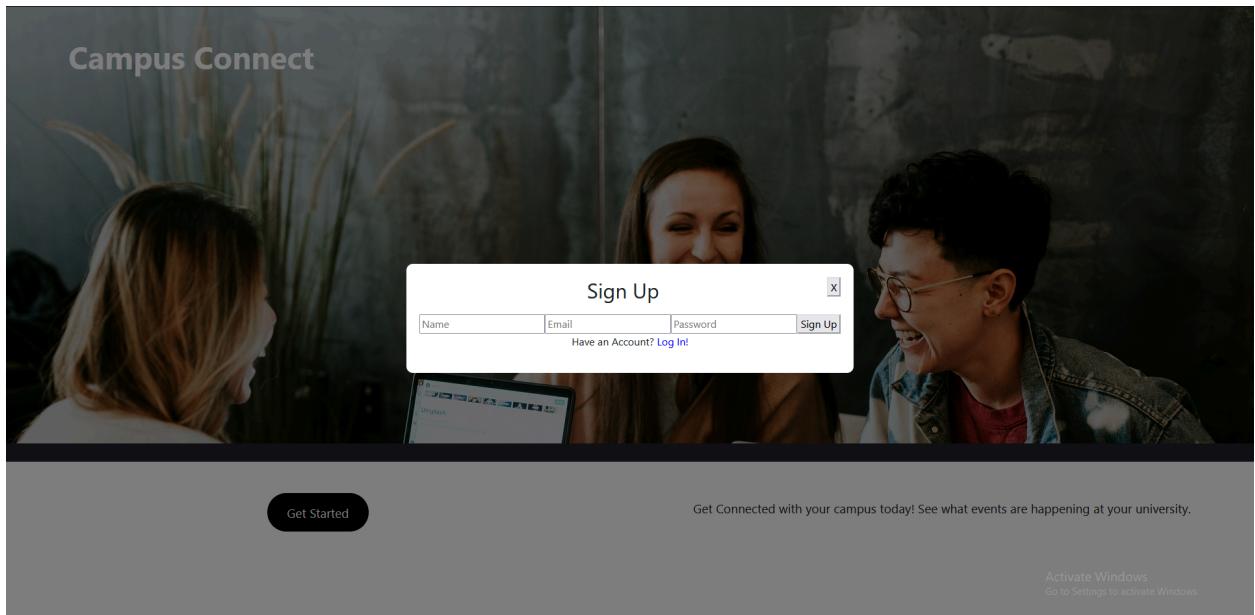
KnightHacks
We are a UCF student organization focusing on dev
www.instagram.com/knighthacks/
Member Count: 5
Status: Active
Create Event Edit RSO
Leave RSO

Events
Final GEM of the Semester!
Officer Elections and Board Nominations for next Se

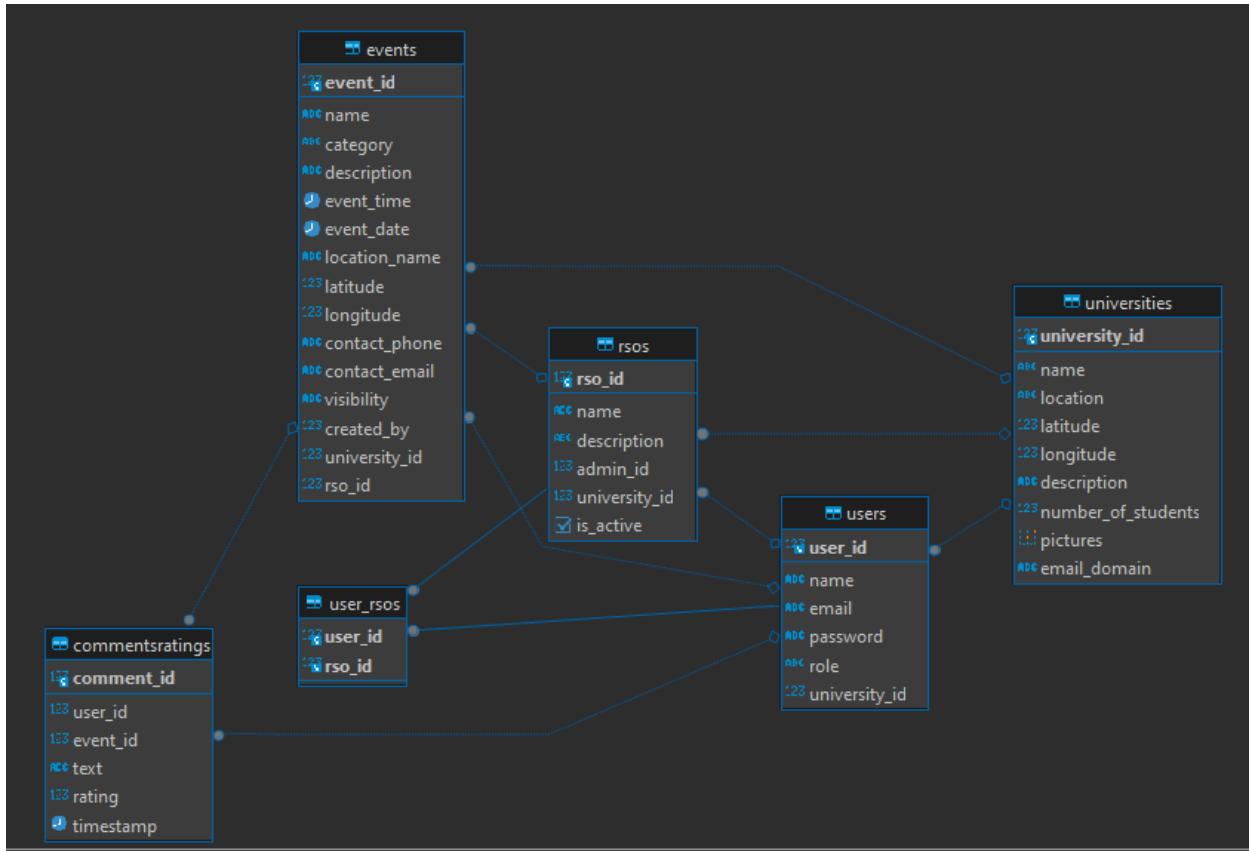
Login Form:



Sign Up Form:



ER-Model



Schema

```

CREATE DATABASE campusconnect;

CREATE TABLE Universities (
    university_id SERIAL PRIMARY KEY,
    name VARCHAR(255) UNIQUE NOT NULL,
    location TEXT,
    latitude DECIMAL(9,6),
    longitude DECIMAL(9,6),
    description TEXT,
    number_of_students INT,
    pictures TEXT[], -- Array of image URLs
    email_domain VARCHAR(255) UNIQUE
);

```

```

CREATE TABLE Users (
    user_id SERIAL PRIMARY KEY,
    name VARCHAR(255),
    email VARCHAR(255) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    role VARCHAR(50) CHECK (role IN ('student', 'admin',
    'superAdmin')),
    university_id INT,
    FOREIGN KEY (university_id) REFERENCES
Universities(university_id)
) ;

CREATE TABLE RSOs (
    rso_id SERIAL PRIMARY KEY,
    name VARCHAR(255),
    description TEXT,
    admin_id INT,
    university_id INT,
    is_active BOOLEAN DEFAULT FALSE,
    FOREIGN KEY (admin_id) REFERENCES Users(user_id),
    FOREIGN KEY (university_id) REFERENCES
Universities(university_id)
) ;

CREATE TABLE Events (
    event_id SERIAL PRIMARY KEY,
    name VARCHAR(255),
    category VARCHAR(50),
    description TEXT,
    event_time TIME,
    event_date DATE,
    location_name VARCHAR(255),
    latitude DECIMAL(9,6),

```

```

longitude DECIMAL(9, 6),
contact_phone VARCHAR(20),
contact_email VARCHAR(255),
visibility VARCHAR(50) CHECK (visibility IN ('public',
'private', 'rso')),
created_by INT,
university_id INT,
rso_id INT,
FOREIGN KEY (created_by) REFERENCES Users(user_id),
FOREIGN KEY (university_id) REFERENCES
Universities(university_id),
FOREIGN KEY (rso_id) REFERENCES RSOS(rso_id)
);

CREATE TABLE CommentsRatings (
comment_id SERIAL PRIMARY KEY,
user_id INT,
event_id INT,
text TEXT,
rating INT CHECK (rating >= 1 AND rating <= 5),
timestamp TIMESTAMP WITHOUT TIME ZONE DEFAULT
CURRENT_TIMESTAMP,
FOREIGN KEY (user_id) REFERENCES Users(user_id),
FOREIGN KEY (event_id) REFERENCES Events(event_id)
);

CREATE TABLE User_RSOS (
user_id INT,
rso_id INT,
PRIMARY KEY (user_id, rso_id),
FOREIGN KEY (user_id) REFERENCES Users(user_id) ON DELETE
CASCADE,
FOREIGN KEY (rso_id) REFERENCES RSOS(rso_id) ON DELETE CASCADE
);

```

```
CREATE OR REPLACE FUNCTION update_rso_status()
RETURNS TRIGGER AS $$

BEGIN

END;

$$ LANGUAGE plpgsql;

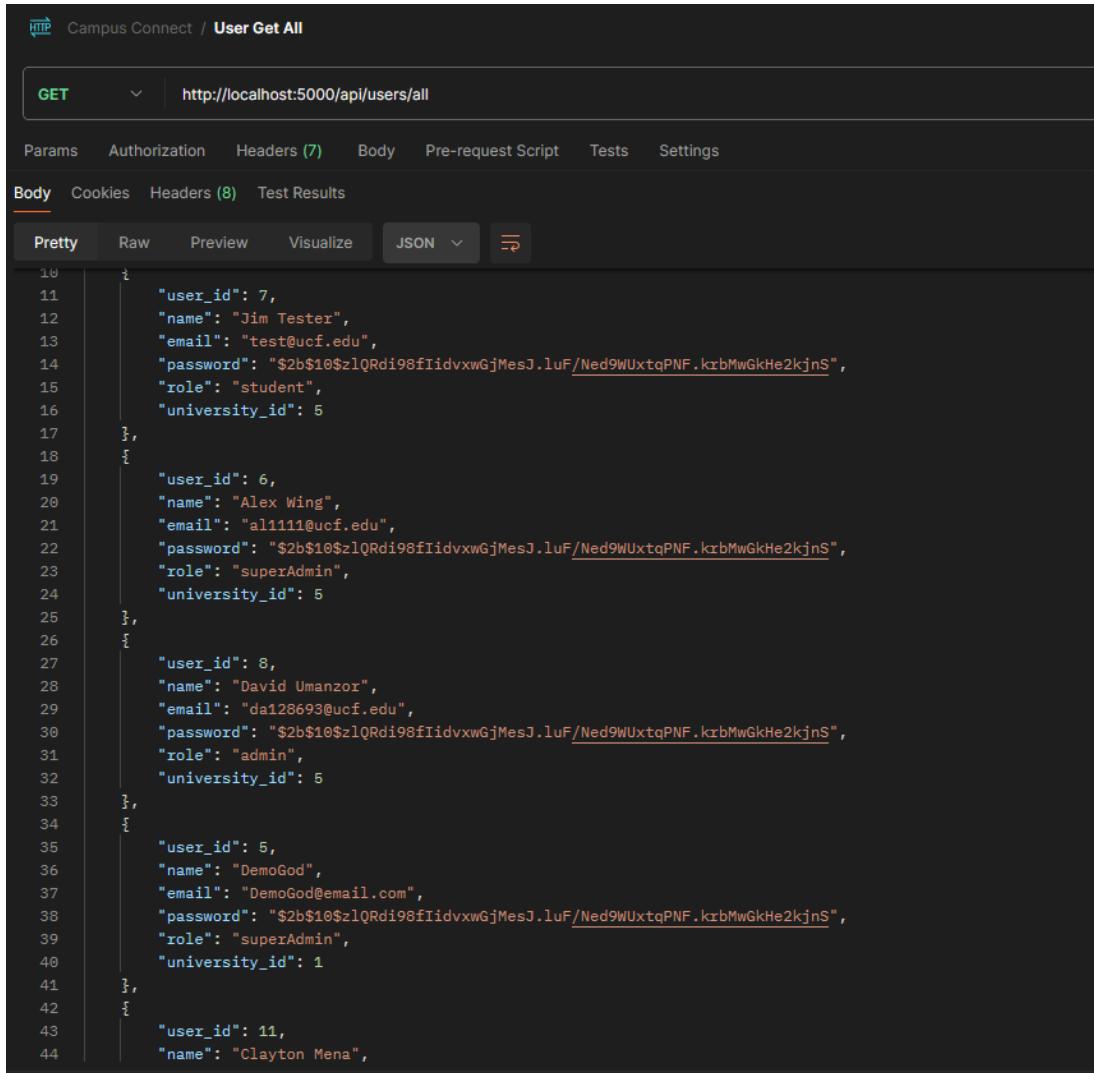
CREATE TRIGGER after_user_rso_insert
AFTER INSERT ON User_RSOs
FOR EACH ROW
EXECUTE FUNCTION update_rso_status();

CREATE TRIGGER after_user_rso_delete
AFTER DELETE ON User_RSOs
FOR EACH ROW
EXECUTE FUNCTION update_rso_status();
```

The Tables

are prepopulated with multiple users, universities, events, and rsos

Example of Users:



```

10  [
11    {
12      "user_id": 7,
13      "name": "Jim Tester",
14      "email": "test@ucf.edu",
15      "password": "$2b$10$zlQRdi98fIidvxwGjMesJ.luF/Ned9WUxtqPNF.krbMwGkHe2kjnS",
16      "role": "student",
17      "university_id": 5
18    },
19    {
20      "user_id": 6,
21      "name": "Alex Wing",
22      "email": "al1111@ucf.edu",
23      "password": "$2b$10$zlQRdi98fIidvxwGjMesJ.luF/Ned9WUxtqPNF.krbMwGkHe2kjnS",
24      "role": "superAdmin",
25      "university_id": 5
26    },
27    {
28      "user_id": 8,
29      "name": "David Umanzor",
30      "email": "da128693@ucf.edu",
31      "password": "$2b$10$zlQRdi98fIidvxwGjMesJ.luF/Ned9WUxtqPNF.krbMwGkHe2kjnS",
32      "role": "admin",
33      "university_id": 5
34    },
35    {
36      "user_id": 5,
37      "name": "DemoGod",
38      "email": "DemoGod@email.com",
39      "password": "$2b$10$zlQRdi98fIidvxwGjMesJ.luF/Ned9WUxtqPNF.krbMwGkHe2kjnS",
40      "role": "superAdmin",
41      "university_id": 1
42    },
43    {
44      "user_id": 11,
45      "name": "Clayton Mena",
46    }
  
```

SQL examples and results

Insert a new rso (user must be a admin of rso making them an member)

```
// Create RSO and add maker as member
router.post('/create', async (req, res) => {
    const { name, description, admin_id, university_id } =
req.body;

    try {
        // Start a transaction
        await pool.query('BEGIN');

        // Insert the new RSO
        const newRso = await pool.query(
            'INSERT INTO rsos (name, description, admin_id,
university_id) VALUES ($1, $2, $3, $4) RETURNING *',
            [name, description, admin_id, university_id]
        );

        if (newRso.rows.length > 0) {
            const rso_id = newRso.rows[0].rso_id;

            // Insert the creator as a member of the RSO
            const userRso = await pool.query(
                'INSERT INTO User_RSOs (user_id, rso_id) VALUES
($1, $2) RETURNING *',
                [admin_id, rso_id]
            );

            // Commit the transaction
            await pool.query('COMMIT');

            res.json({ rso: newRso.rows[0], membership:
userRso.rows[0] });
        } else {
            // Rollback in case of any failure
        }
    } catch (err) {
        console.error(err);
        res.status(500).json({ error: 'Internal Server Error' });
    }
});
```

```

        await pool.query('ROLLBACK');
        res.status(400).json({ message: "Failed to create
RSO" });
    }
}
} catch (err) {
    // Rollback the transaction on error
    await pool.query('ROLLBACK');
    console.error(err.message);
    res.status(500).json({ message: "Server error" });
}
}
);

```

Campus Connect / RSO Create

POST <http://localhost:5000/api/rsos/create>

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies Beautify

Body (8)

```

1 {
2     "name": "Baking Club",
3     "description": "Club for all knights into baking pastries and desserts!",
4     "admin_id": 7,
5     "university_id": 5
6 }

```

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```

1 {
2     "rso": {
3         "rso_id": 10,
4         "name": "Baking Club",
5         "description": "Club for all knights into baking pastries and desserts!",
6         "admin_id": 7,
7         "university_id": 5,
8         "member_count": 0,
9         "is_active": false
10     },
11     "membership": [
12         {
13             "user_id": 7,
14             "rso_id": 10
15         }
16     ]
}

```

Status: 200 OK Time: 159 ms Size: 487 B Save as example

Add user to a rso

```

router.post('/add', async (req, res) => {
    const { userId, rsoid } = req.body;
    try {
        const newUserRso = await pool.query(
            'INSERT INTO User_RSOS (user_id, rso_id) VALUES ($1,
$2) ON CONFLICT DO NOTHING RETURNING *',
            [userId, rsoid]
        );
        if (newUserRso.rows.length > 0) {

```

```

        res.json({ message: "User added to RSO
successfully.", newUserRso: newUserRso.rows[0] });
    } else {
        res.status(400).json({ message: "User already a
member or invalid IDs." });
    }
} catch (err) {
    console.error(err.message);
    res.status(500).json({ message: "Server error" });
}
}) ;

```

POST http://localhost:5000/api/userRsos/add

Body (JSON)

```

1 {
2     "user_id": 12,
3     "rso_id": 10
4 }

```

Body (Pretty)

```

1 {
2     "message": "User added to RSO successfully.",
3     "newuserRso": [
4         {
5             "user_id": 12,
6             "rso_id": 10
7         }
8     ]
9 }

```

Create Event

```

router.post('/create', async (req, res) => {
    const { name, category, description, event_time, event_date,
location_name, latitude, longitude, contact_phone,
contact_email, visibility, created_by, university_id, rso_id } =
req.body;

    try {
        // Check for any events at the same location and time
        const conflictCheckQuery = `
            SELECT r.name AS rso_name, e.location_name,
e.event_time
            FROM events e
            JOIN RSOs r ON e.rso_id = r.rso_id
        `;

```

```

        WHERE e.location_name = $1 AND e.event_date = $2 AND
e.event_time = $3 AND r.is_active = true
        LIMIT 1;
    `;

    const conflictCheck = await
pool.query(conflictCheckQuery, [location_name, event_date,
event_time]);

    if (conflictCheck.rows.length > 0) {
        const conflictEvent = conflictCheck.rows[0];
        return res.status(409).json({ message: `Conflict
with event from "${conflictEvent.rso_name}" -
${conflictEvent.location_name}, at ${conflictEvent.event_time}` });
    }

    // No conflict found
    const createEventQuery = `
        INSERT INTO events
            (name, category, description, event_time,
event_date, location_name, latitude, longitude, contact_phone,
contact_email, visibility, created_by, university_id, rso_id)
        VALUES
            ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, $11, $12,
$13, $14)
        RETURNING *`;
}

// Execute the query to create a new event
const newEvent = await pool.query(createEventQuery,
[name, category, description, event_time, event_date,
location_name, latitude, longitude, contact_phone,
contact_email, visibility, created_by, university_id, rso_id ||
null]);
res.json(newEvent.rows[0]);

```

```

} catch (err) {
    console.error(err.message);
    res.status(500).json({ message: "Server error", details: err.message });
}
};

HTTP Campus Connect / Event Create
POST http://localhost:5000/api/events/create
Params Authorization Headers (9) Body Pre-request Script Tests Settings
 none  form-data  x-www-form-urlencoded  raw  binary  GraphQL JSON
1 {
2     "name": "From Couch to 5k Runner, Workshop",
3     "category": "",
4     "description": "Learn how to go from not being a runner to being ready to run your first 5k!",
5     "event_time": "4:00 PM",
6     "event_date": "04/05/2024",
7     "location_name": "CB1 204",
8     "contact_phone": "813-203-5321",
9     "contact_email": "Magicrunner@mag.edu",
10    "visibility": "rso",
11    "created_by": 5,
12    "university_id": 1,
13    "rso_id": 1,
14    "latitude": 32.43,
15    "longitude": "-34.000000"
16 }
17
Body Cookies Headers (8) Test Results
Pretty Raw Preview Visualize JSON
1 {
2     "event_id": 14,
3     "name": "From Couch to 5k Runner, Workshop",
4     "category": "",
5     "description": "Learn how to go from not being a runner to being ready to run your first 5k!",
6     "event_time": "16:00:00",
7     "event_date": "2024-04-05T04:00:00.000Z",
8     "location_name": "CB1 204",
9     "contact_phone": "813-203-5321",
10    "contact_email": "Magicrunner@mag.edu",
11    "visibility": "rso",
12    "created_by": 5,
13    "university_id": 1,
14    "rso_id": 1,
15    "latitude": "32.430000",
16    "longitude": "-34.000000"
17 }

```

Add comment

```
router.post('/create', async (req, res) => {
  const { user_id, event_id, text, rating } = req.body;
  try {
    const newCommentRating = await pool.query(
      'INSERT INTO commentsratings (user_id, event_id, text, rating) VALUES ($1, $2, $3, $4) RETURNING *',
      [user_id, event_id, text, rating]
    );
    res.json(newCommentRating.rows[0]);
  } catch (err) {
    console.error(err.message);
  }
});
```

POST <http://localhost:5000/api/commentsratings/create>

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1  {
2   "user_id": 8,
3   "event_id": 5,
4   "text": "What a swell time to be out with friends",
5   "rating": 4
6 }
```

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```

1  {
2   "comment_id": 7,
3   "user_id": 8,
4   "event_id": 5,
5   "text": "What a swell time to be out with friends",
6   "rating": 4,
7   "timestamp": "2024-04-18T06:21:05.575Z"
8 }
```

View events (I called it allowed for what user is allowed to see)

```

router.get('/allowed/', async (req, res) => {
  const { userId } = req.query;
  try {
    const events = await pool.query(`

      SELECT e.* FROM Events e
      LEFT JOIN RSOS r ON e.rso_id = r.rso_id
      WHERE e.visibility = 'public'
      OR (e.visibility = 'private' AND e.university_id =
      (SELECT university_id FROM Users WHERE user_id = $1))
      OR (e.visibility = 'rso' AND e.rso_id IN (SELECT
      rso_id FROM User_RSOS WHERE user_id = $1) AND r.is_active)
      ` , [userId]);
    res.json(events.rows);
  } catch (err) {
    console.error(err.message);
    res.status(500).json({ message: "Server error" });
  }
});
  
```

Campus Connect / Event Allowed

GET http://localhost:5000/api/events/allowed?userId=8

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

<input checked="" type="checkbox"/> Key	Value	Description
<input checked="" type="checkbox"/> userId	8	

Body Cookies Headers (8) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```

    7   "event_time": "19:00:00",
    8   "event_date": "2024-04-20T04:00:00.000Z",
    9   "location_name": "ENG1 - 432",
   10   "contact_phone": "776-564-2003",
   11   "contact_email": "knighthacks@gmail.com",
   12   "visibility": "private",
   13   "created_by": 5,
   14   "university_id": 5,
   15   "rso_id": 2,
   16   "latitude": "28.662400",
   17   "longitude": "-81.200100"
   18 },
   19 {
   20   "event_id": 15,
   21   "name": "Intercollegiate Race",
   22   "category": "Competition",
   23   "description": "Race between schools come out make friends and learn something new or put up a new record",
   24   "event_time": "10:00:00",
   25   "event_date": "2024-04-28T04:00:00.000Z",
   26   "location_name": "State Track",
   27   "contact_phone": "321-322-3333",
   28   "contact_email": "contact@gmail.com",
   29   "visibility": "public",
   30   "created_by": 5,
   31   "university_id": 1,
   32   "rso_id": null,
   33   "latitude": "28.629618",
   34   "longitude": "-81.435428"
   35 }
```

Constraint Enforcement

Active vs Inactive RSO:

KnightHacks

We are a UCF student organization focusing on developing computer science skills. Welcome to all majors, identities, and schools. <https://linktr.ee/knighthacks> <https://www.instagram.com/knighthacks/>

Member Count: 4

Status: Inactive

[Join RSO](#)

Events

RSO is not active. RSO requires more members to be an actively registered RSO.

Campus Connect Home User Profile Search Log Out

KnightHacks

We are a UCF student organization focusing on developing computer science skills. Welcome to all majors, identities, and schools. <https://linktr.ee/knighthacks> <https://www.instagram.com/knighthacks/>

Member Count: 5

Status: Active

[Leave RSO](#)

Events

- Game Night
Bring Spike Ball and Foot Ball and have a blast!
- First GEM of the Semester!
Officer Elections and Board Nominations for next Semester! Links on the linktree
- RSO Only
RSO ONLY EVENT
- University Event
University students only

Only Admin can make events:

KnightHacks

We are a UCF student organization focusing on developing computer science skills. Welcome to all majors, identities, and schools. <https://linktr.ee/knighthacks> <https://www.instagram.com/knighthacks/>

Member Count: 5

Status: Active

[Create Event](#) [Edit RSO](#) [Leave RSO](#)

Events

Game Night
Bring Spike Ball and Foot Ball and have a blast!

First GEM of the Semester!
Officer Elections and Board Nominations for next Semester! Links on the linktree

RSO Only

In my site the buttons for creating event and editing the rso do not show up for anyone outside the admin.

KnightHacks

We are a UCF student organization focusing on developing computer science skills. Welcome to all majors, identities, and schools. <https://linktr.ee/knighthacks> <https://www.instagram.com/knighthacks/>

Member Count: 5

Status: Active

[Leave RSO](#)

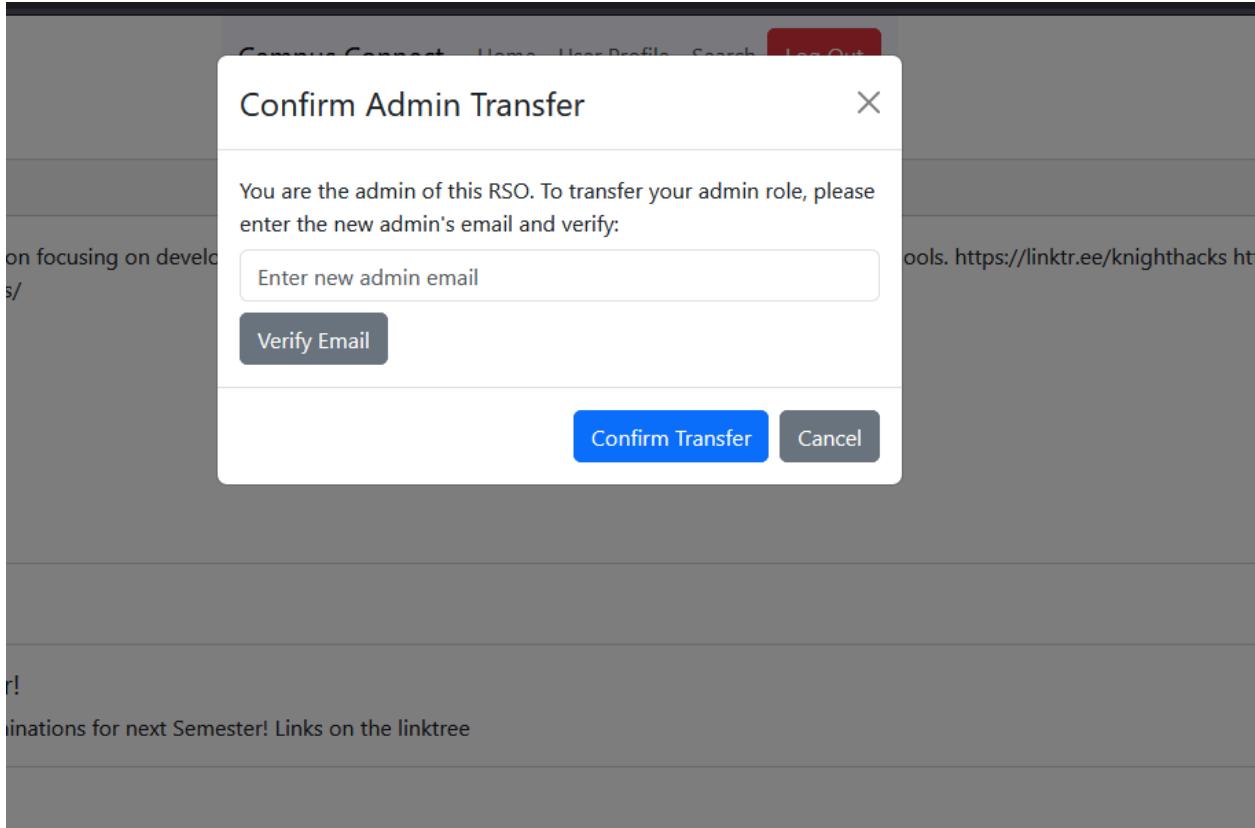
Events

Game Night
Bring Spike Ball and Foot Ball and have a blast!

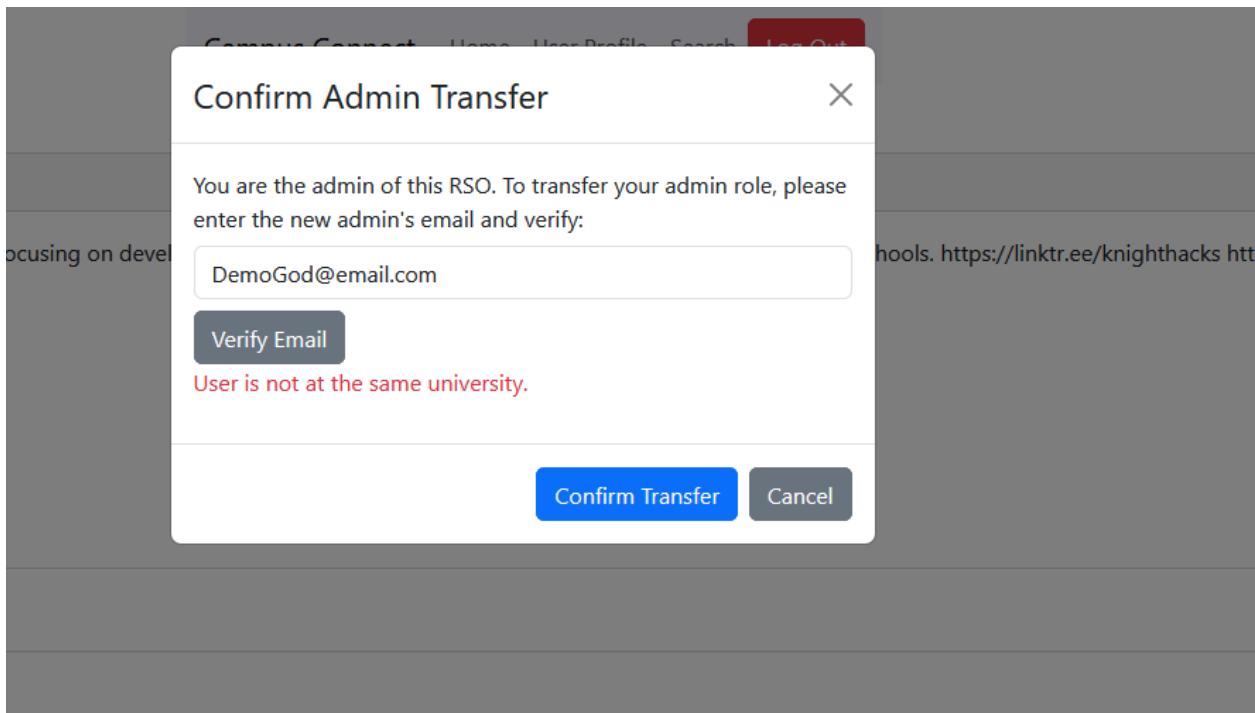
First GEM of the Semester!

Admin in RSO:

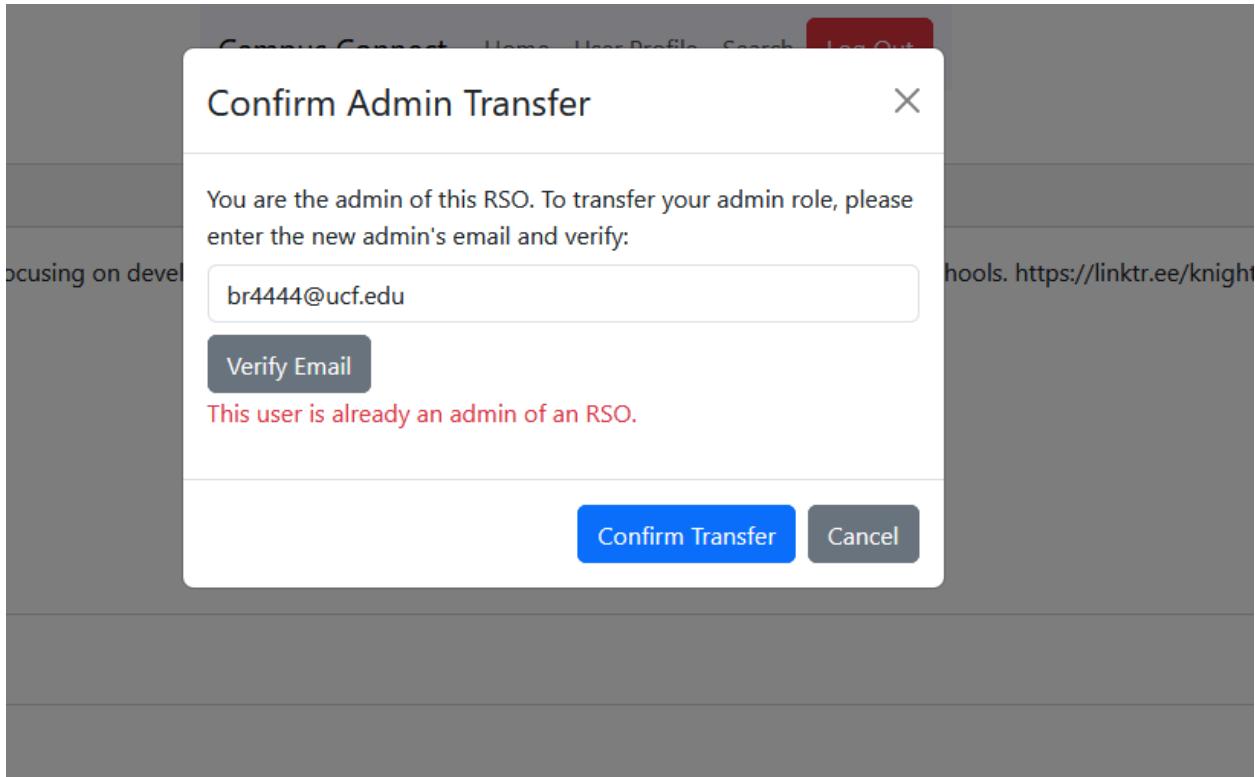
This makes sure Admins can not leave without transferring ownership to another user.



User must be eligible such that, at same university

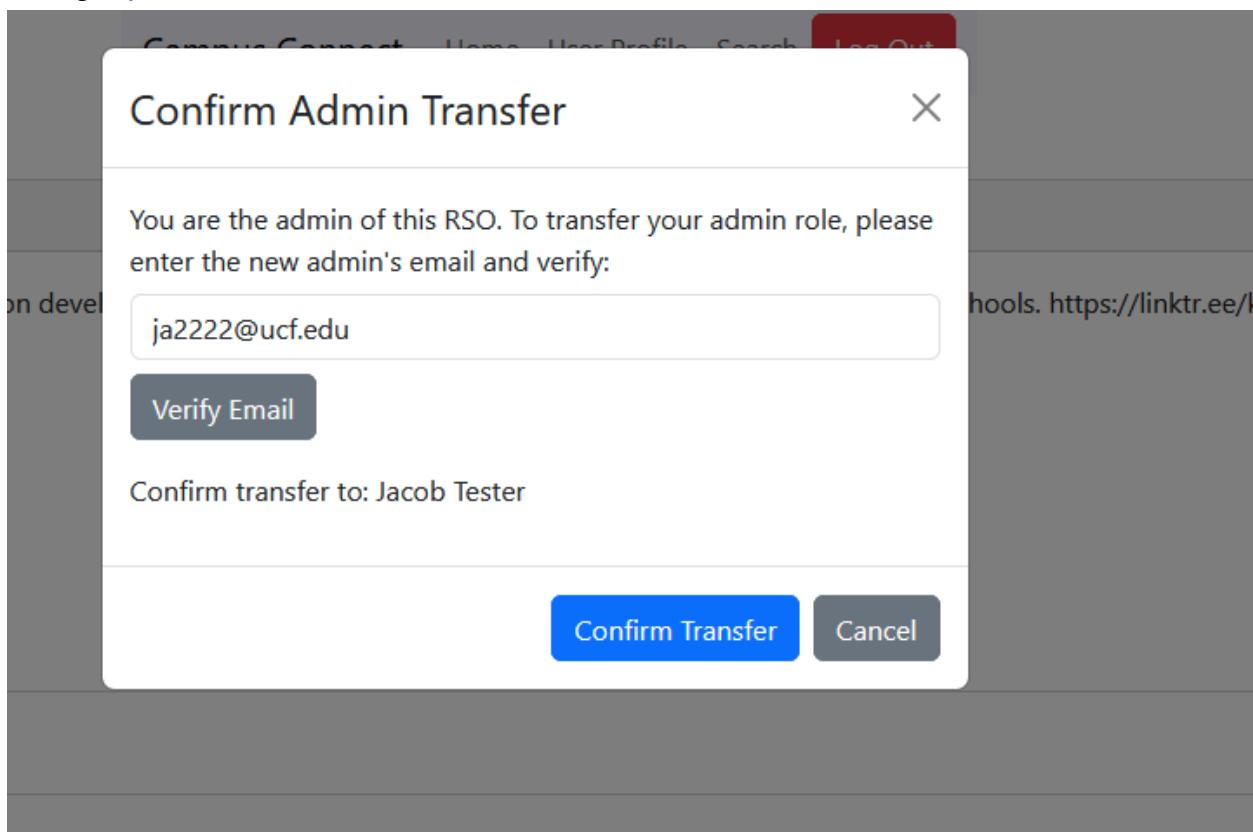


And must not be admin of another RSO

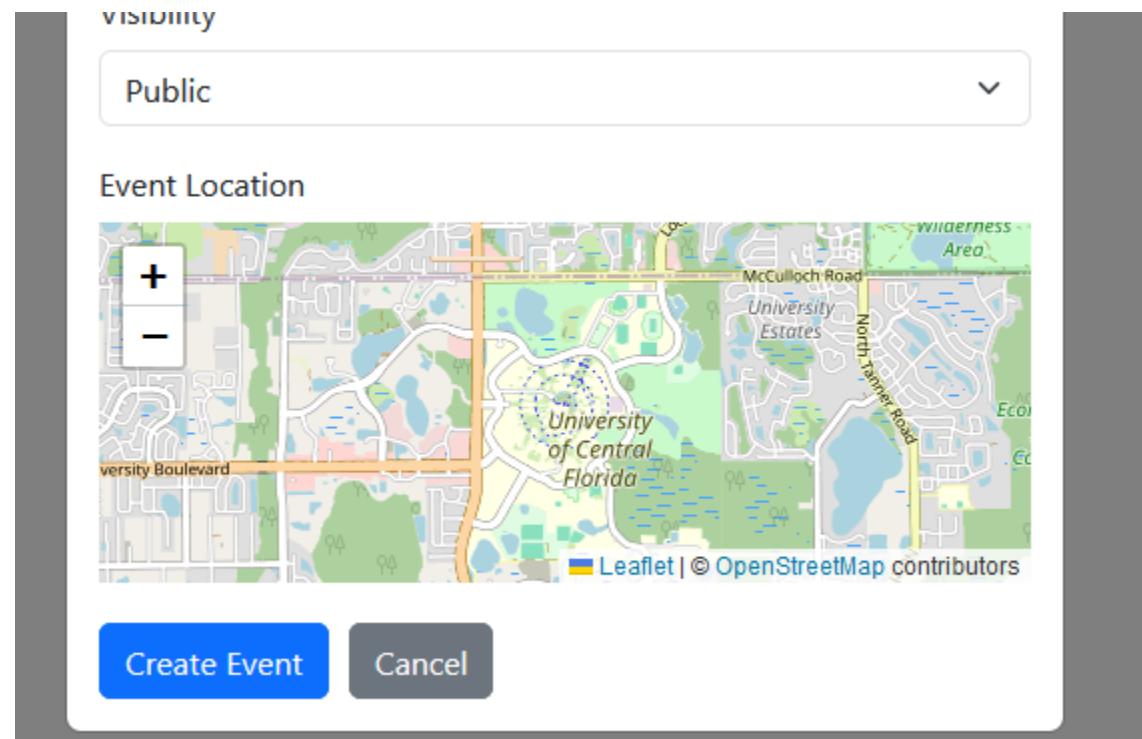


Otherwise it displays the users name for assurance your giving administration role to

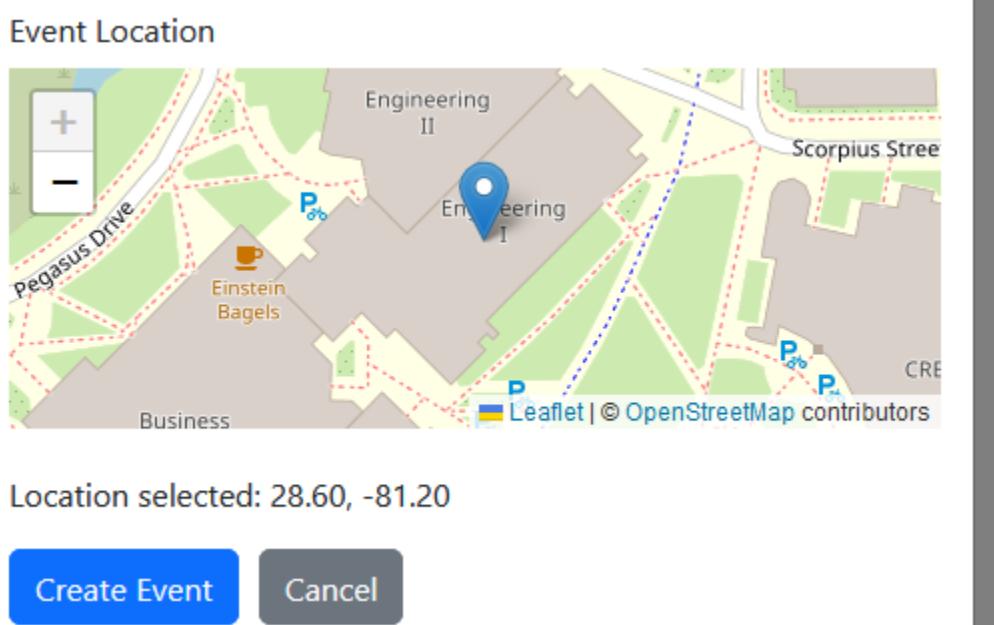
the right person



Advanced Features



Added ability to put pin on the map with lat and lon coordinates. Along with a location section for more detail such as room number, hallway directions, ect



Conclusions

Database performance:

I found query response time reasonable fast and had no issues having multiple queries happen for different checks as pages load.

Desired features:

I liked being able to add the map as it gives a better visual especially for longer range compared to just a building name and number it helps verify for locations possibly outside the standard buildings and like in memory mall.

Also being able to test using hashed passwords so that it can't just be copied from the database.

Problems encountered:

In retrospect I would have tried to separate things more clearly I tried to make forms and functions that could be used multiple times such as (create form is also the edit form) but it didn't work all the time and needed more reworks and taken more time than I would have liked.