

EE / CPR E / S E 492 – [sdmay21-51](#)

Team 51: Constrained Re-Planning in Spatial Crowdsourcing

Bi-Weekly Report 5: 03/16 – 03/29

Client &/Advisor: Goce Trajcevski

Team Members:

- Logan Anderson: Frontend/Test Engineer
- Steven Sheets: Backend/Test Engineer, Report Manager
- Nicholas Heger: Frontend Engineer, Progress Manager
- Jared Weiland: Backend Engineer
- James Volpe: Frontend/Backend Engineer

Past Week Accomplishments:

- Implemented Jobs in the database and on the backend. – Steven
- Modified Skills, Location, Workers, Employers, and Jobs s.t. they could be used on the frontend. – Steven
- Cleaned up Workers and Employees shared parameters by creating an abstract class User. Functionality remains the same. – Steven
- Implemented and tested graphql on the backend. After ensuring it worked properly using its interactive GUI, found (while the backend is running) at localhost:8080/graphiql, moved into the Frontend to test its usage there. Ran into quite an annoying issue with CORS but was able to remedy it (after an hour of searching) with a simple @CrossOrigin tag on the main application. Also got password encryption setup. – Steven
- Successfully had communication between the frontend, backend, and database. Unfortunately, my device was screaming, as I needed to run both the frontend and the backend locally. See Pending Issues for plan of attack. – Steven
- Implemented the skills page for workers that will later query the workers table in the database and put the worker's skills into a list. Added an "Add Skill" button that will pop up a modal with skill name and rating fields workers can populate that will later send the new skills off to the backend to be added to that worker in the database. – Nicholas
- Created an assignments page to serve as the home page for workers that will pull the list of tasks they have been assigned and has a link to their skills page. – Nicholas
- Added a tasks page that will serve as the home page for task generators. This page pulls the current tasks this task generator has requested and displays them in a list. It also has a button that will open a modal with fields the task generator can fill in and submit to send a new task request off to the backend. – Nicholas

- Increased functionality of the sign-in and sign-up pages to now route the user to their respective home page based on the type of account the user indicates they have (worker or task generator) and used Links to carry over the user's information from the sign in/up pages to the user's home page. – Nicholas
- Implemented dynamic start to end point map routing using entered addresses to geocoordinates. – Logan
- Integrated map component into master client system, as well as fixing large issues with integration. – Logan

**Pending Issues:**

- Once we transition to running our code on servers, instances of uri's using localhost will need to be updated to the new link.
- Preparation for our second PIRM meeting tomorrow.

**Individual Contributions:**

<b>Name</b>	<b>Contributions</b>	<b>Bi-Weekly Hours</b>	<b>Cumulative Hours</b>
Logan Anderson	React, Report	8	34
Steven Sheets	Meeting Organization, React/SB, SB/Mongo, graphql, Apollo, Report	12	37
Nicholas Heger	Use Cases/User Story, React/SB, Report	13	43
Jared Weiland	Server and Database, Report	4	23
James Volpe	React/graphql/Mongo Research, Report	3	20

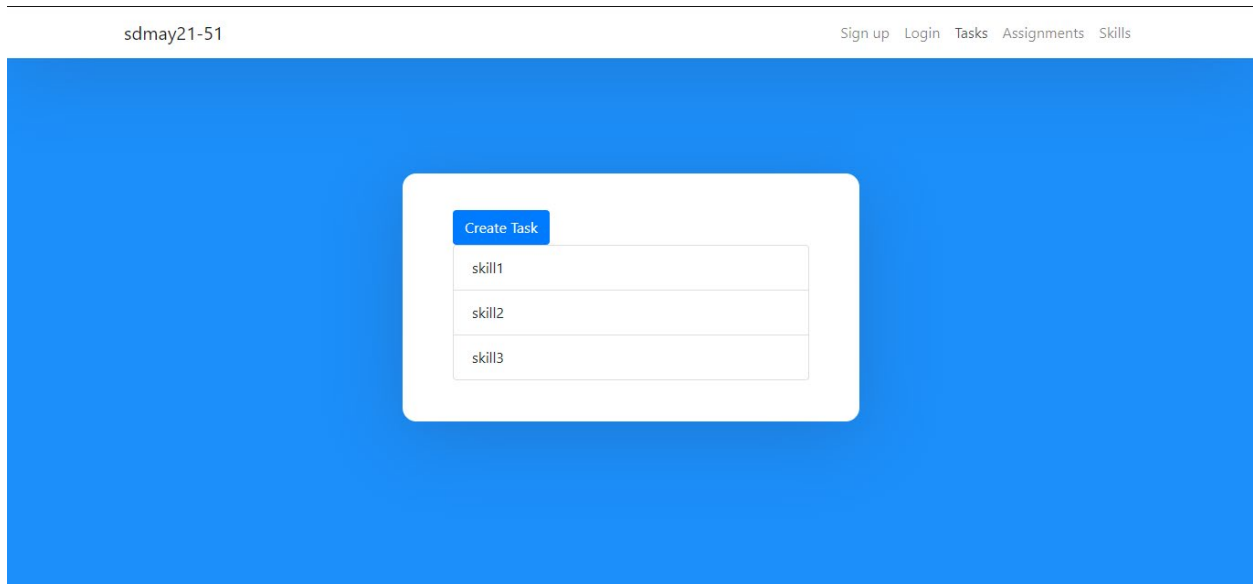
**Plans for Coming Week:**

- Need to establish a server in which to run the backend on. I believe the same applies for the frontend, to some extent. While, for a proof of concept, things running locally is acceptable, for convenience this should be dealt with. On top of that, setting up CI/CD s.t. we do not need to manually update it.
- Add more Mutations to the frontend to send more data to the backend. Specifically, add a mutation for task generators creating new tasks and workers adding new skills.

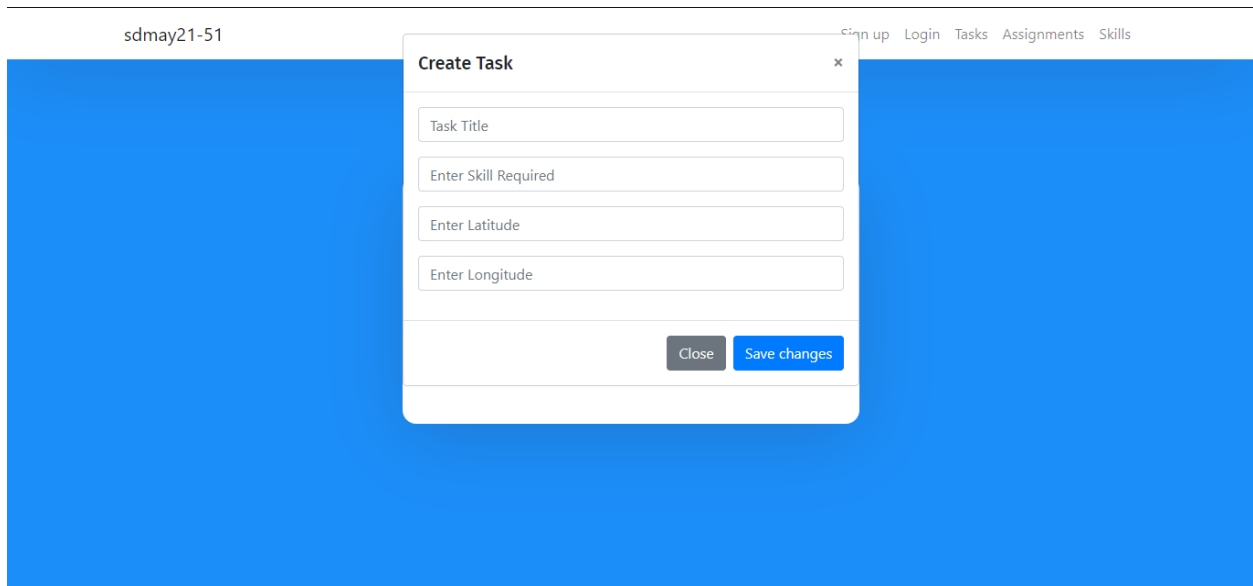
- Add Queries on the frontend to request data from the backend. Specifically, query the database to validate a user trying to sign in on the sign in page, and query the backend to populate the tasks, assignments, and skills lists.
- Move states from map component into app component. Ready map system for requesting route start and end coordinates from backend.

Some Screenshots:

### Tasks Page (modal closed):



### Tasks Page (modal open):



## Assignments Page:

sdmay21-51

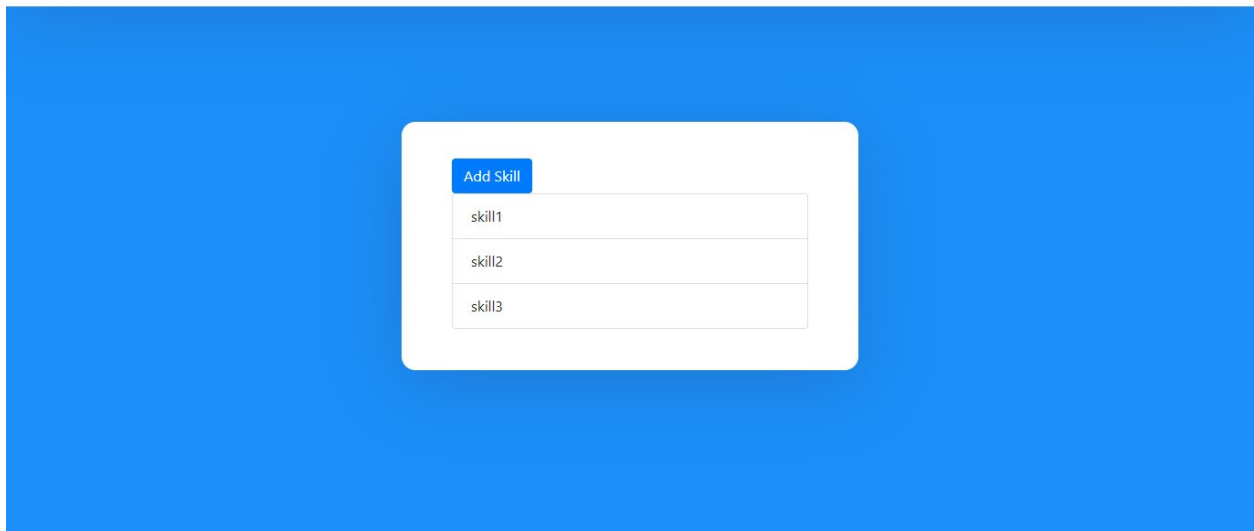
[Sign up](#) [Login](#) [Tasks](#) [Assignments](#) [Skills](#)



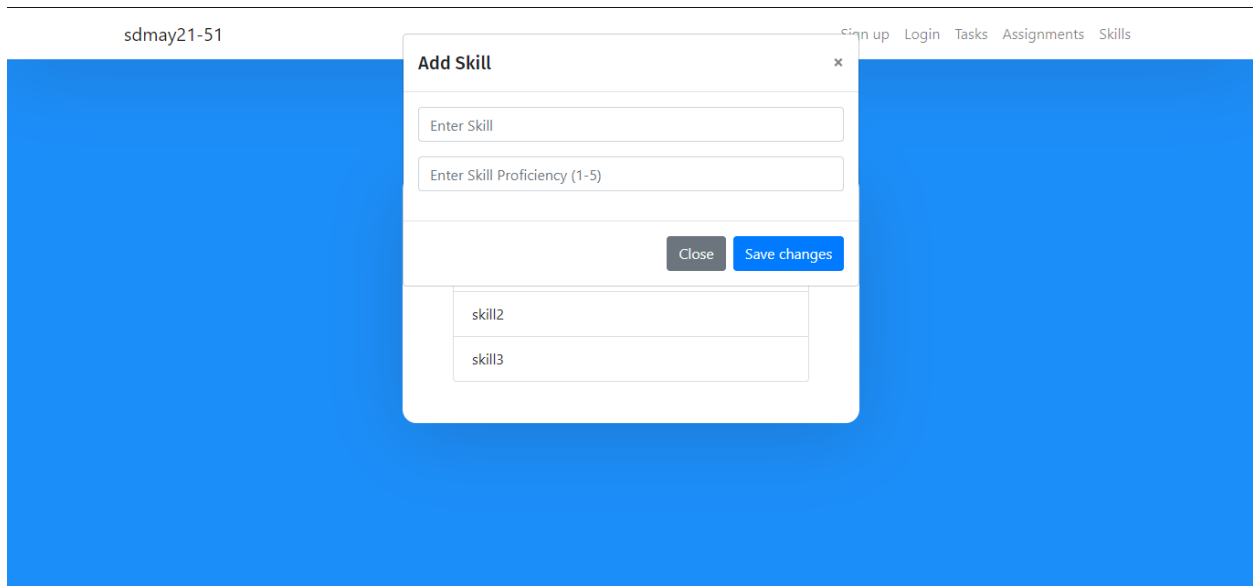
## Skills Page (modal closed):

sdmay21-51

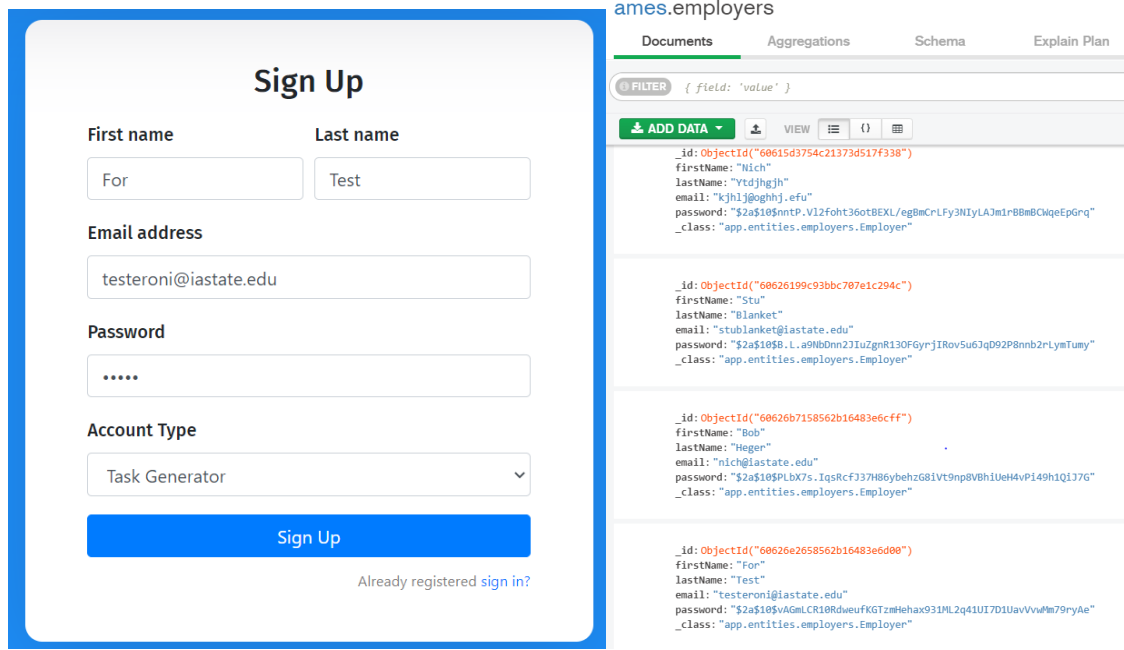
[Sign up](#) [Login](#) [Tasks](#) [Assignments](#) [Skills](#)



## Skills Page (modal open):

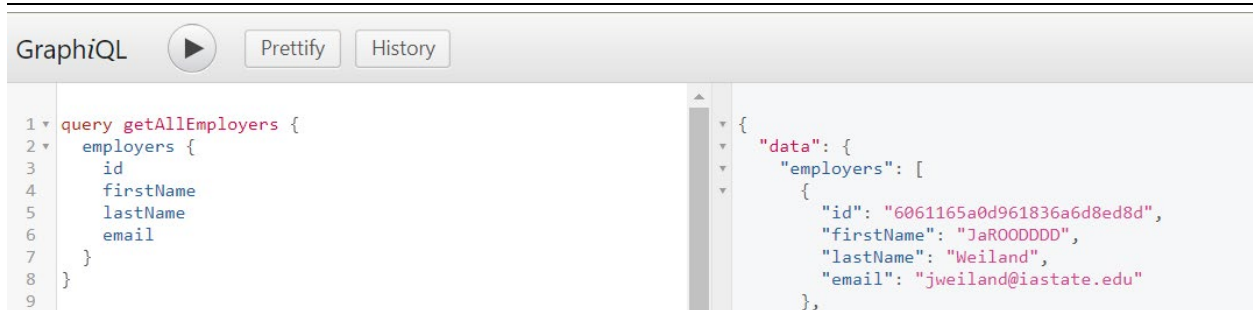


## Sign-up Page Integrated with Backend/DB Test:



## Updating an Employer Test with graphql:

```
_id: ObjectId("6061165a0d961836a6d8ed8d")  
firstName: "JaROODDD"   
lastName: "Weiland"   
email: "jweiland@iastate.edu"   
password: "$2a$10$BpwFhDomRivfWdXfRov/3.pW5yUzrODt4YqZOYznKTuK0JlsEthjW"   
_class: "app.entities.employers.Employer"
```



The screenshot shows the GraphQL IDE interface. On the left, a query is defined: `query getAllEmployers { employers { id firstName lastName email } }`. On the right, the JSON response is displayed: `{ "data": { "employers": [ { "id": "6061165a0d961836a6d8ed8d", "firstName": "JaROODDD", "lastName": "Weiland", "email": "jweiland@iastate.edu" } ] } }`. The interface includes a play button, 'Prettify', and 'History' buttons.



The screenshot shows the GraphQL IDE interface. On the left, a mutation is defined: `mutation updateEmployer { updateEmployer(id: "6061165a0d961836a6d8ed8d", firstName: "Bob") { firstName lastName } }`. On the right, the JSON response is displayed: `{ "data": { "updateEmployer": { "firstName": "Bob", "lastName": "Weiland" } } }`. The interface includes a play button, 'Prettify', and 'History' buttons.

```
> _id: ObjectId("6061165a0d961836a6d8ed8d")  
firstName: "Bob"   
lastName: "Weiland"   
email: "jweiland@iastate.edu"   
password: "$2a$10$BpwFhDomRivfWdXfRov/3.pW5yUzrODt4YqZOYznKTuK0JlsEthjW"   
_class: "app.entities.employers.Employer"
```

## General Backend Structure:

The screenshot shows an IDE window titled "Backend - EmployerMutationResolver.java - Administrator". The left sidebar displays the project structure for "Backend" located at "D:\OneDrive - Iowa State University\spring2021\se492\sdmay21-51\Backend". The main editor displays the code for the `EmployerMutationResolver` class, which implements `GraphQLMutationResolver`. The code includes:

```
11 @Service
12 public class EmployerMutationResolver implements GraphQLMutationResolver {
13
14     @Autowired
15     private PasswordEncoder passwordEncoder;
16
17     @Autowired
18     private EmployerRepository employerRepository;
19
20     public Employer createEmployer(String firstName, String lastName, String email, String password) {
21         Employer employer = new Employer(firstName, lastName, email, passwordEncoder.encode(password));
22         return employerRepository.save(employer);
23     }
24
25     public Employer updateEmployer(String id, Optional<String> firstName, Optional<String> lastName,
26     Optional<Employer> employerOptional = employerRepository.findById(id);
27     try {
28         Employer employer = employerOptional.get();
29
30         firstName.ifPresent(employer::setFirstName);
31         lastName.ifPresent(employer::setLastName);
32         email.ifPresent(employer::setEmail);
33         password.ifPresent(p -> employer.setPassword(passwordEncoder.encode(p)));
34
35         employerRepository.save(employer);
36         return employer;
37     } catch (NoSuchElementException e) {
38         return null;
39     }
```

## Map functioning on Master Client:

