

*HOW WORKRIGHT  
DELIVERS THE  
RIGHT JOBS*

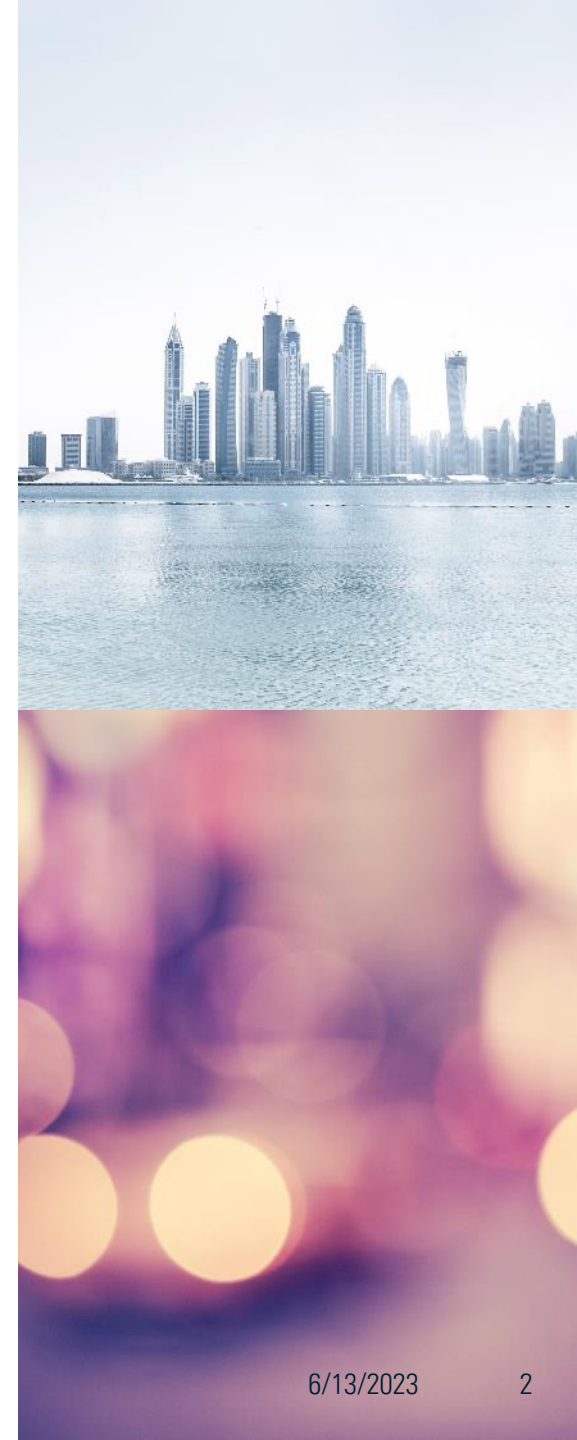
*PROOF OF  
CONCEPT FOR A  
USER-FRIENDLY  
JOBS PLATFORM*

Dustin, Jing, Lewis



# *THE PROBLEM*

- LinkedIn? Indeed? Monster?
- Modern apps and the job search?
- Opportunity for a **low-friction** model





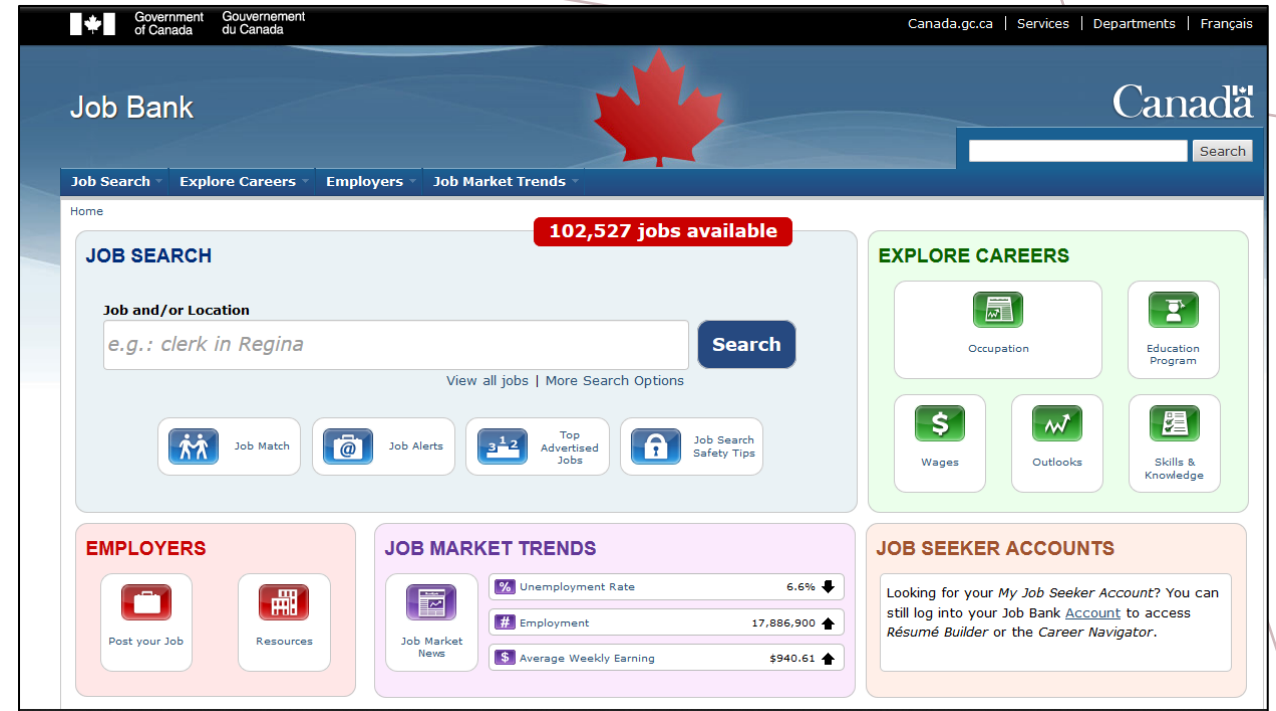
# THE WORKRIGHT SOLUTION

- Simplifying the job search process
- Proximity is the key
- Employers get **directly** involved; no middlemen

The screenshot displays a web application for job searching. At the top, there is a search bar with a dropdown menu currently open, showing a list of industries: All Industries, Medical, Agriculture, Finance, Information, Catering (highlighted with a mouse cursor), Transportation, Publishing, and Technology. To the right of the dropdown is a red 'Search' button. Below the search bar, there are four job listings arranged in a 2x2 grid. Each listing has a red header bar with the job title, a sub-header for the industry, and a brief description. The listings are: 'Registered Nurse' in the Medical industry, 'Financial Analyst' in the Finance industry, 'Chef de Cuisine' in the Catering industry, and 'Medical Lab Technologist' in the Medical industry. The 'Financial Controller' listing is partially visible at the bottom right.

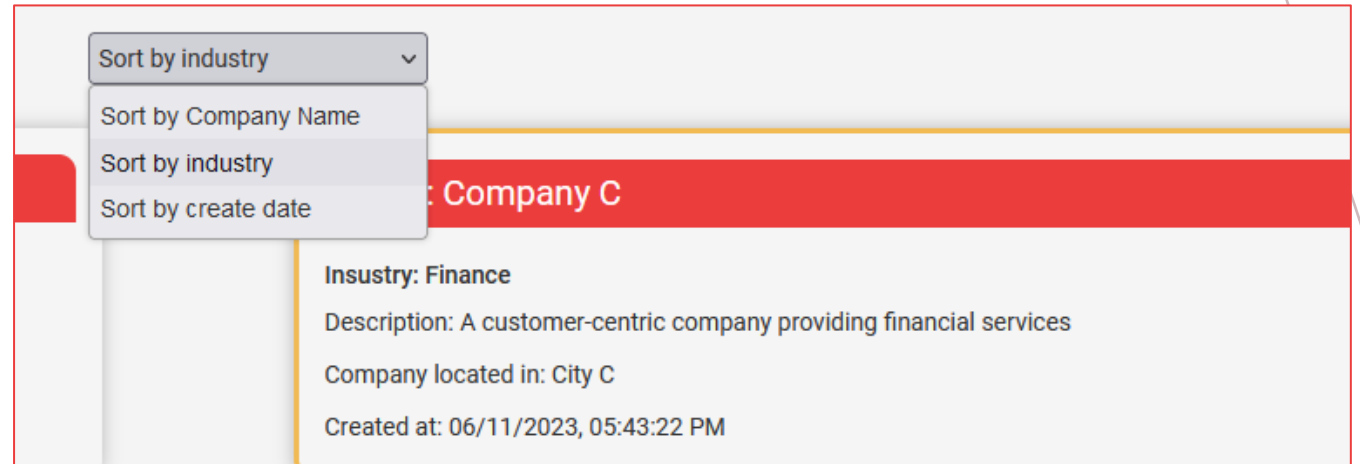
Job Title	Industry	Description
Registered Nurse	Medical	Seeking a registered nurse with experience in a hospital setting.
Financial Analyst	Finance	Looking for a skilled financial analyst to join our team.
Chef de Cuisine	Catering	Experienced chef needed to lead our culinary team.
Medical Lab Technologist	Medical	
Financial Controller		

# LET'S TRADE FIRST IMPRESSIONS



- Compare with Job Bank Canada, JobIllico, and others
- **Busy-ness** on home pages especially a deterrent
- The user interface itself a source of “**decision fatigue**” ON TOP of the job search

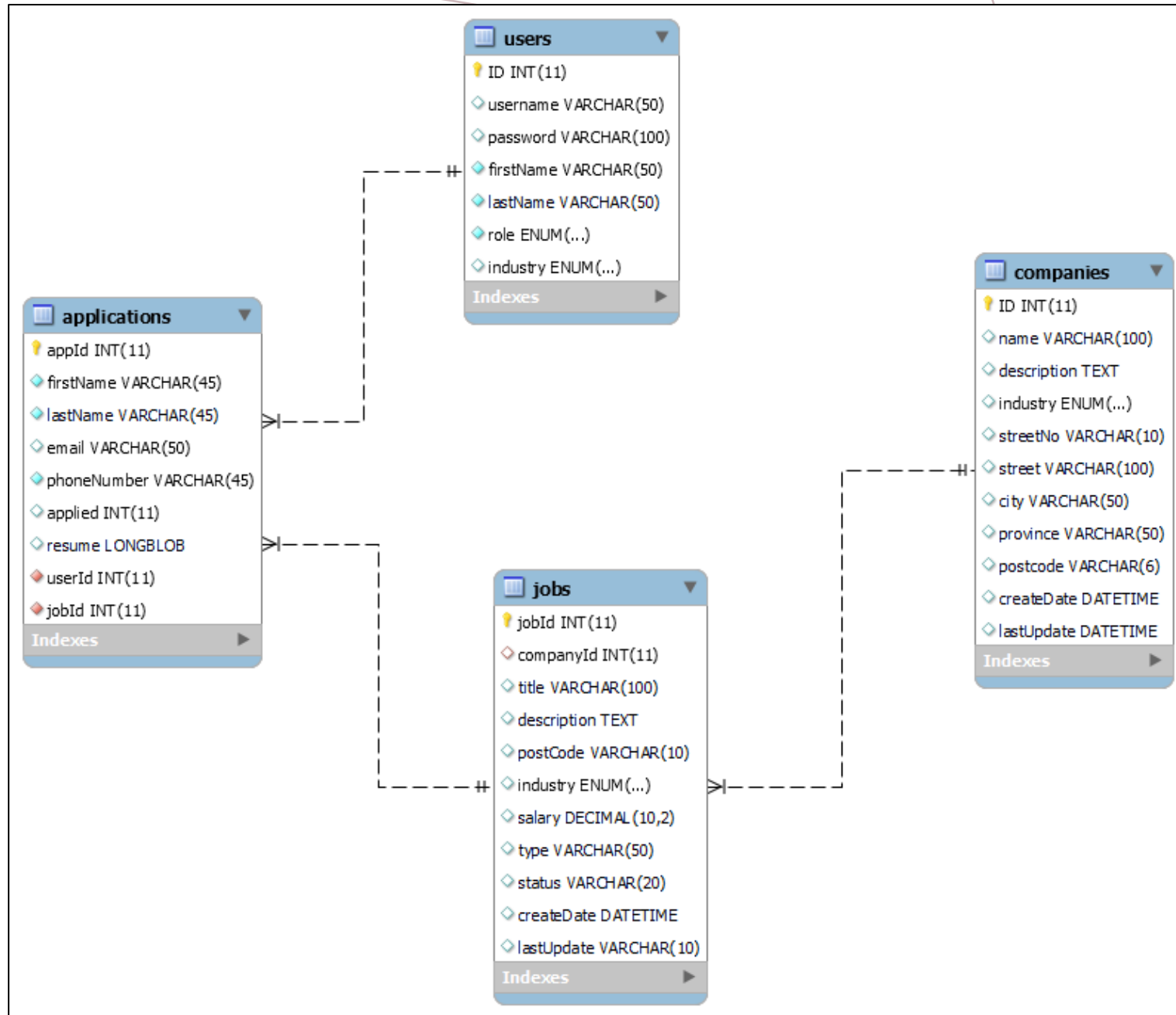
# *IMPLEMENTING A SMOOTHER USER EXPERIENCE*



- **Minimalist** registration
- Companies and positions accessible within two clicks
- Easy, lightweight search tools
- Preventing user attrition

# DATABASE STRUCTURE

- Users table holds job seekers and employers data
- Jobs table for active postings
- Company information as an addendum to job postings



# NODE JS TECHNOLOGIES

- nodemon allowed for **rapid debugging** "npm run dev" development script
- jQuery, very simply easy to use in manipulating DOM for front-end validation
- log.info available through npmlog



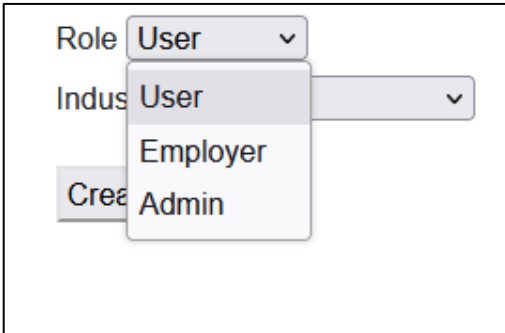
```
"scripts": {  
  "start": "node server",  
  "dev": "nodemon server"  
},  
"keywords": [],  
"author": "",  
"license": "ISC",  
"dependencies": {  
  "body-parser": "^1.20.2",  
  "cors": "^2.8.5",  
  "express": "^4.18.2",  
  "log": "^6.3.1",  
  "mysql2": "^3.3.3",  
  "npm": "^9.7.1",  
  "npmlog": "^7.0.1"  
},  
"devDependencies": {  
  "nodemon": "^2.0.22"  
}
```



```
info WorkRightApp IpAddress: ::ffff:127.0.0.1 add a company Customers Inc. into database  
req.query.sortBy = name
```

# *A MULTIPLICITY OF ROLES*

- Primary users are **job-seekers**, but there are also employer roles in the WorkRight app
- Challenge of database queries and permissions across different roles



A screenshot of a web application interface showing a dropdown menu for role selection. The menu is open, displaying three options: 'User', 'Employer', and 'Admin'. The 'User' option is currently selected. The dropdown is part of a form with labels 'Role', 'Indus', and 'Crea' visible to the left of the dropdown.

Role
User
Employer
Admin



# BUILDING A SHIP A SEA

- Authentication can be tough to implement after the fact
- With multiple roles, many moving parts
- Avoiding code duplication while not cutting any corners
- You validated front-end registration – but did you cover all your vulnerabilities? (Postman)

I am a: Job-Seeker ▾

Job-Seeker  
Employer

First

Last Name:

Username:

Password:

Password (repeated):

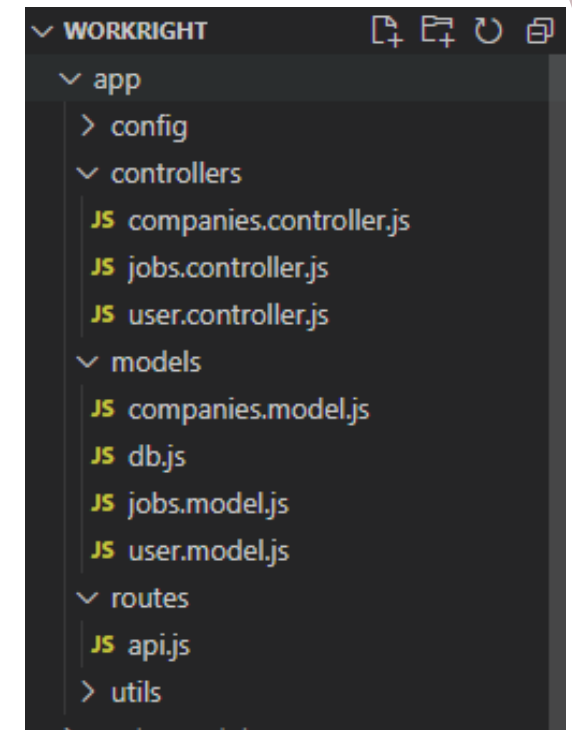
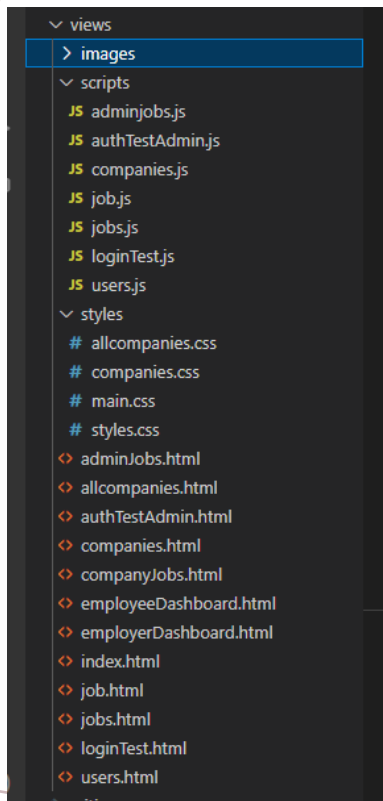
```
POST http://localhost:7077/api/users

Body
none form-data x-www-form-urlencoded raw
1
2
3 ..... "username": "bronco123",
4
5 ..... "password": "bronco123",
6
7 ..... "firstName": "slippy",
8
9 ..... "lastName": "slippy",
10
11 ..... "role": "admin"
12
13
```

```
1
2 "message": "Authentication required but not provided"
3
```

# *DIFFICULTY PROJECT SCOPE:*

- Multiple controllers, multiple models
- Almost 20 static pages
- How to organize a project development **efficiently** & **realistically**?
- Project management a crucial, albeit skill



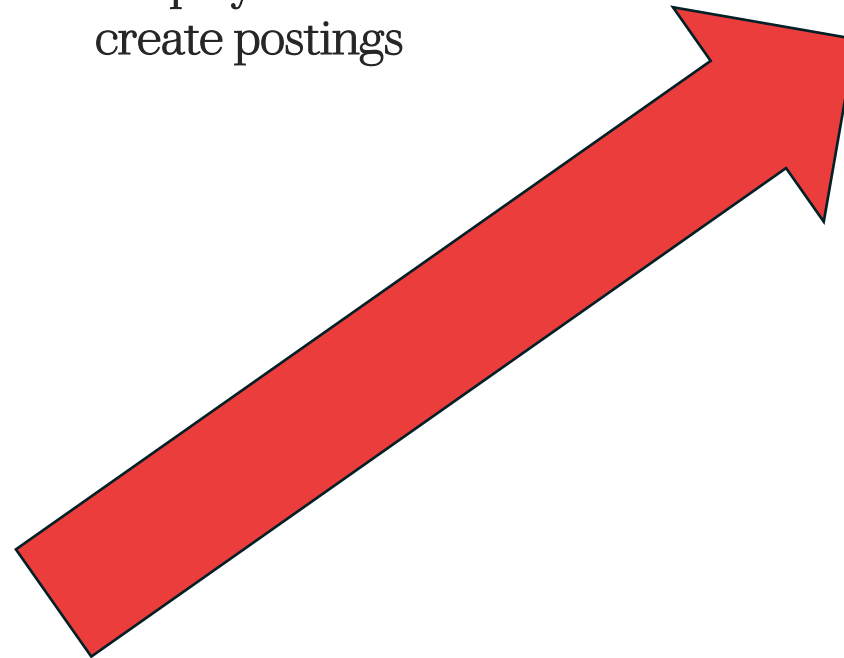
# *BUILDING TOWARD A UX*

Job-seekers enjoy a pleasant, low mental overhead search

Employers enticed to create postings

Administrators empowered to facilitate

Rock steady server-side



# *SIMPLE PROBLEMS IN A BROAD SCOPE*

- Finding small but **hard to detect** issues
- Figuring out their implications for surrounding code
- Iterating this over a whole code base!

```
22
23 //doesn't work this way, job_listing doesn't trigger click event
24 $(".job-listing").on("click", function() {
25     const jobID = $(this).data('id');
26     window.location.href = '../job.html?id=' + jobID;
27 });
28
29
30 $("#searchForm").on("submit", function(e) {
31     e.preventDefault(); // prevent the default form submission
32     page = 0; // reset to 0
33     var postcode = $("#postcodeSearchBox").val();
34     var industry = $("#industrySearchBox").val();
--
```



# CHALLENGE PAGES

- The jobs table is huge, on user side infinite scroll
- How to request only a certain number of rows for Admin
- Avoiding code duplication while not cutting any corners

Job Postings			
Previous Page		Next Page	
Job Id	Company Id	Title	Industry
48	28	Registered Nurse	Medical
88	28	Financial Analyst	Finance
90	28	Financial Analyst	Finance
91	30	Chef de Cuisine	Catering
92	28	Medical Lab Technologist	Medical
93	28	Financial Controller	Finance
94	28	Catering Sales Manager	Catering
95	30	Medical	Medical

```
Job.getTenRows = (jobId, result) => {
  let query;

  let absId = Math.abs(jobId);

  if (jobId < 0){
    query = "SELECT * FROM jobs WHERE jobId < ? ORDER BY jobId DESC LIMIT 10;";
  } else if (jobId > 0){
    query = "SELECT * FROM jobs WHERE jobId > ? ORDER BY jobId LIMIT 10;";
  }
}
```

# JING, HTML, AND JQUERY

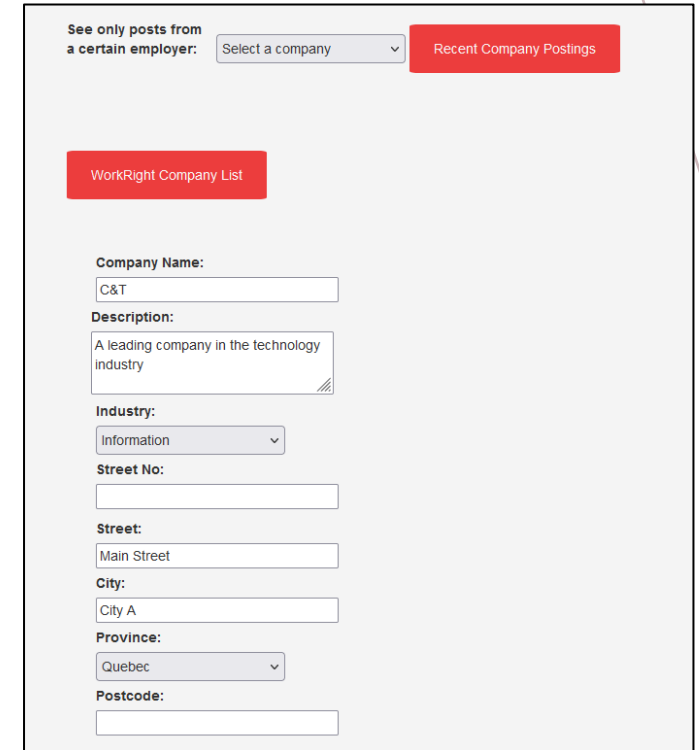
- Centering divs is a difficult task!
- Back end can deliver, but even minimal style needed
- Making sure validation works on front-end and backend without stepping on each other's toes
- Double click is still unconquered

```
<script>
$(document).ready(function() {
  var delay = 400;
  var clickTimer = null;

  $('tr').click(function() {
    var tr = $(this);

    if (clickTimer === null) {
      clickTimer = setTimeout(function() {
        console.log('Single click');
        clickTimer = null;
      }, delay);
    } else {
      clearTimeout(clickTimer);
      clickTimer = null;
      console.log('Double click');
    }
  });

  $('tr').dblclick(function() {
    return false;
  });
});
```



The screenshot shows a web interface for filtering and viewing company listings. At the top, there is a filter section with the text "See only posts from a certain employer:" followed by a dropdown menu labeled "Select a company" and a red button labeled "Recent Company Postings". Below this is a red button labeled "WorkRight Company List". The main form area contains several input fields: "Company Name:" with a text box containing "C&T", "Description:" with a text area containing "A leading company in the technology industry", "Industry:" with a dropdown menu showing "Information", "Street No:" with a text box, "Street:" with a text box containing "Main Street", "City:" with a text box containing "City A", "Province:" with a dropdown menu showing "Quebec", and "Postcode:" with a text box.

# *JING PDF UPLOADS*

- How to allow the sever to receive a job application?
- Storing the file in the users table
- What kind of validation required?

**Active**  
**06/11/2023, 09:30:00 AM**  
Looking for a skilled financial analyst to join our team.  
Full Name  Email  Phone  Resume  
 No file selected.

# *WHAT WE LEARNED*

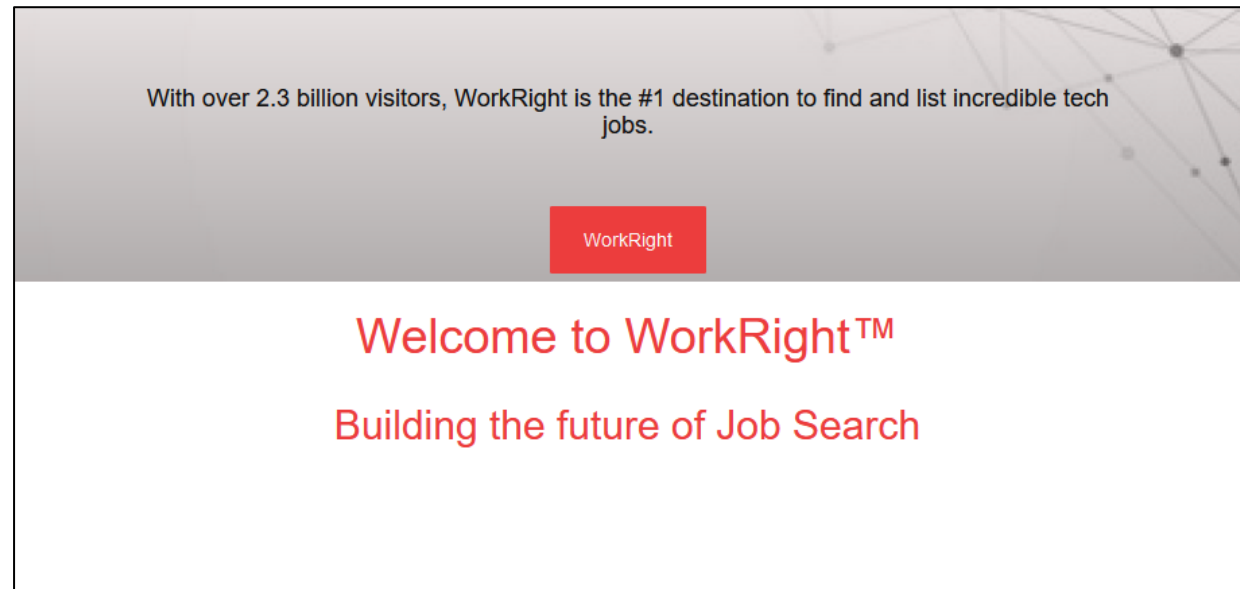
- Pagination is difficult
- Callbacks can get very intense!
- Sometimes the issue is tiny! (calling for the array.length eth object)
- Planning a codebase requires a lot of thought

```
        tableRow.append(companyIdCell, titleCell, 1);  
        tableRow.css("background-color", "#f0f0f0");  
        tableBody.append(tableRow);  
    }  
    topPage = data[0].jobId;  
    bottomPage = data[data.length - 1].jobId;  
    })  
}
```



# *WHAT WE LEARNED*

- Handling API content dynamically
- We bit off more than we could chew – but through this learned so much about how **long it takes to actually get things done.**
- Newfound appreciation for even simple working websites and apps – every functional full stack is an achievement.





# *FUTURE*

- Love to implement the maps feature google API
- Direct sending of pdf files needs to be completed
- Getting more complicated queries from database,  
like join on columns and tables

# *SUMMARY*

- Difficult to combine code between team members
  - Difficult to know in advance how long things will take
- 
- Felt as though we **gained months of back-end** knowledge in only a week.
  - Looking forward to next projects!

