Lewis Innes-Miller, Dustin Ruck, Jing Wei

Dustin Ruck

Jing Wei

WorkRight



WorkRight Web Services Project

Lewis, Dustin, Jing

# Project

### Proposal:

A job postings application where job seekers and employers can interface with one another. Job and candidate search supplemented by map display in addition to standard filtering criteria.

### Description:

Our full-stack project combines what we have learned about server-side development with our front-end knowledge in order to implement a job/candidate search application. We seek to develop an application with versatile appeal to job seekers and job offerors. Taking inspiration from the functionality of sites like GlassDoor, Indeed, and replicating the homepage design of WeWorkRemotely, our end goal is to build an app capable of displaying different job postings to different users, and different job candidates to different employers.

There are therefore two kinds of user accounts on our site. Firstly, there are the job seeker user accounts. These are individuals looking for job opportunities in their industry. Once they have created an account and submitted the pertinent details such as the role, salary, and location criteria for their job search, their dashboard will allow them to view postings in paginated form. The results will also be combined with external APIs like google maps, which will make the job search more dynamic with graphical displays of location data. Job seekers will be able to apply to jobs that interest them, as well as read any responses employers have sent to their inbox.

Secondly, there are the employer accounts. These are companies with job opportunities on offer. When they log in to their account, they will not see pages of jobs but rather pages of candidates. Candidates who have directly applied to one of the employer’s job postings will be marked graphically and displayed more saliently than other candidates. But employer accounts can only see candidates who have experience relevant to the employer’s industry. If the employer wishes to change their company’s industry details, they can do so, and this will expand the pool of candidates available to them.

# Technologies and Libraries

* GitHub
* Node.js
  + Express, validator, MySQL date-fns, ejs, socket.io(maybe) (dev: nodemon, winston)
* Heroku
* MySQL workbench
* Javascript
  + JQuery, AJAX, lodash
* HTML/CSS
  + Bootstrap 5
* API: Google maps, GlassDoor, Linkedin, Lever…

# Special Features

* Display job postings by location visually using Google maps
* 2 types of user in addition to admin
* Connect with GlassDoor api for cross listing of posts
* Able to define a radius of interest
  + E.g. there are 4 postings for work in your field in a 20 km radius of your location
* Employers and Job-seekers able to exchange messages via internal mail system

# Extra Challenges

* Internal mail system
* Live chat ability? (requires websockets? Feasible?
* File transfer
  + E.g. job seeker uploads CV and transfers to employer account or employer sends out a written job offer

# URLS:

* /index.html
* /register
* /login
* /admin
* /dashboard

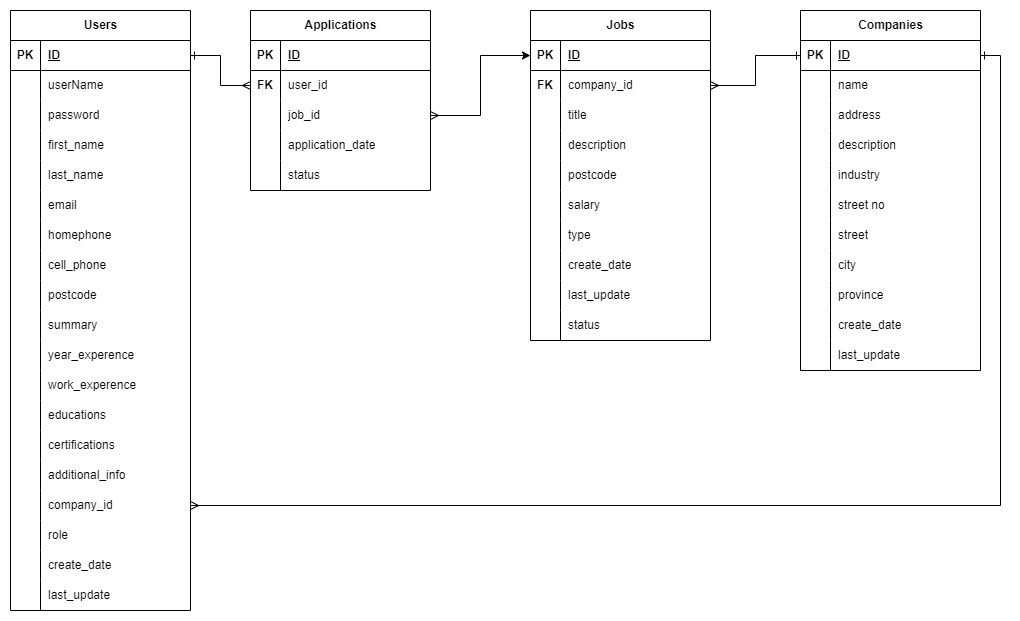
Figure - WorkRight Database ER Diagram

Figure 2- WorkRight Use Case Diagram

A picture containing text, diagram, screenshot, circle

Description automatically generated

Figure 3- mock-up of WorkRight job seeker page (after login)

A screenshot of a computer

Description automatically generated with medium confidence

Figure 4- a job seeker's individualized resultsA screenshot of a computer

Description automatically generated with medium confidence

Figure 5- a page of job postings organized by industry

A screenshot of a computer

Description automatically generated with medium confidence

Figure 6- mock-up of WorkRight job seeker page (after login)

A screenshot of a computer

Description automatically generated with medium confidence

# Questions

1. What about using a view engine like ejs? Generally, what is the advantage of creating dynamic content on the server side using something like ejs as opposed to having all dynamism be on the client side?
2. How do URLs for authenticated users work? How does the URL of my dashboard differ from another user’s dashboard? Potential infinity of users entails a potential infinity of URLs?
3. Seems like a chicken or egg problem as to whether we aim at job seekers or employers… job seekers will only use the site if employers make postings, yet employers only make postings if there are candidates. Which is primary actor?
4. How different/difficult is the implementation of direct messaging system vs live chat?
5. createServer vs createPool in node js.

# Feedback

-Project proposal needs to begin more directly, not *in media res* like a novel (*incorporated)*

-Location should be postal code (*incorporated)*

-Do not make multiple user ‘types’. Instead, one-to-many relationship between company and users, and employers are users with certain nullable fields. (*incorporated)*

-Use case diagram missing: Admin CRUD postings, new user-returning user logic can be more explicitly (special case, not a different type of actor) (*incorporated)*

-Map display: same approach as the airport implementation - Canada post API

-Question 1 – don’t use a view engine

-Question 2 – to be discussed later in class

-Question 3- Business logic – not pertinent to this project

-Question 4- Focus on what you have so far, don’t think about messaging

-Question 5- performance concern, not relevant to this project