CS348 - Milestone 1

Aaron Choo, Jason Chu, Kallen Tu, Charles Zhang, Jack Zhang

February 6, 2020

Description

The application we want to build is a website named Bread that fuses the services of Tinder and WaterlooWorks. Students looking for internships, new graduates looking for entry-jobs, and job-hunters looking for full-time jobs postings are the users of Bread.

When the user first signs in, they will see job cards that describe the job tasks, the hiring company, the location of the job, and other data related to the job posting. Users will be able to swipe right if they are interested in applying to the job or swipe left if they are uninterested in the job posting.

We will keep track of what jobs the user has viewed already and what jobs they are interested in. This data will be used for interview matching with the employer. Employers belong to one or more companies. Employers will be able to post their own jobs under a company and do the same swiping process for interested candidates. Employers will not be able to see candidate profiles until the candidate has already swiped on that company.

Planned Features

We (tentatively) plan to have the following features.

1. Candidates can register and login with accounts and see the Candidate view
2. Employers can register and login with accounts and see the Employer view
3. Candidates can see Jobs postings one by one and swipe left or right to indicate if they like the job or not
4. Employers can post Jobs under a Company. After candidates swipe on this job, Employers can see who liked this job and swipe to indicate if they like the candidate

5. After both sides indicate they like each other, there is a match. Matches are posted to a dedicated page which Candidates and Employers can both see each other's contact info.
6. We plan to allow Candidates to filter by job location and tags.

Technical Design

Our database uses MySQL and is hosted on AWS. In addition to .sql files, we use python scripts to read csv datasets to populate our database tables.

Our backend is written in Go, which exposes the database to the client through HTTP REST APIs in JSON format.

Our frontend is an Android app written in Java, and clients will use this to interact with our data.

Plan for getting data

We will take job data sourced from Monster.com on kaggle to populate our jobs database. The CSV representation of this data can be found in data/monster-job-data.csv. We have a script (script/import_job_data.py) to read this csv data and insert into our database.

However, the Monster.com job dataset on kaggle does not contain company names. Therefore, we will use the list of Fortune 500 companies to populate our companies database, then randomly assign each job to one of the companies in the database.

For application specific data like user accounts, likes, matches etc we will manually create them.

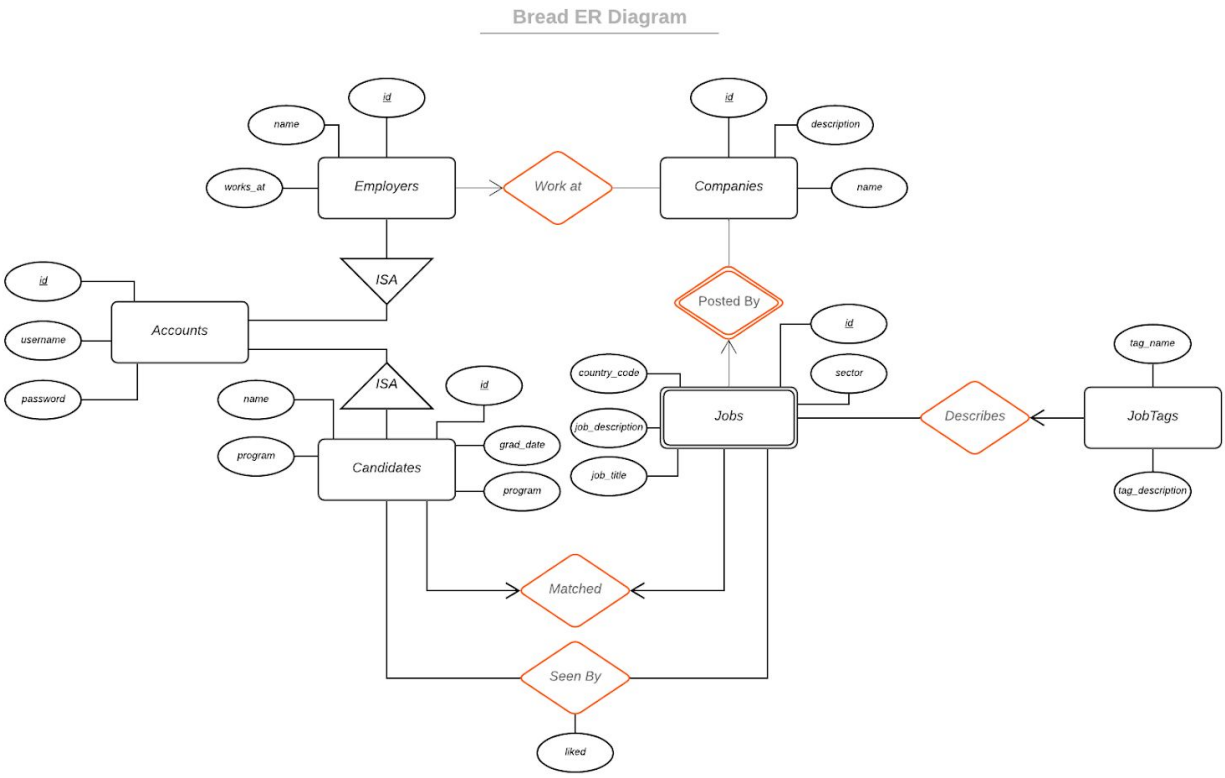
Assumptions

For the Jobs data which we imported from Kaggle, we are assuming that the id string which they provided is unique.

For candidate seen job (CSJ) and job seen candidate (JSC), we are assuming that a candidate always sees the job before the job (employer) sees the candidate. That is, a row must exist in CSJ before it can exist in JSC.

For a match, we are assuming that there are both CSJ and JSC entries before an entry can be written to the match table. That is, a match cannot exist until both the candidate and job (employer) has seen and swiped on each other.

E/R Diagram



List of Tables

Account

Column	Constraints
id	Primary key
username	
password	

Candidates

Column	Constraints
id	Primary key, also Foreign key to account.id
name	
program	
grad_date	
description	

Employers

Column	Constraints
id	Primary key, also foreign key to account.id
name	
works_at	Foreign key to companies

Companies

Column	Constraints
id	Primary key
name	
description	

Jobs

Column	Constraints
_id	Primary key
country_code	
job_description	
job_title	
sector	
company_id	Foreign key to Companies

Job Tag

Column	Constraints
tid	Primary key
tag_name	
tag_description	
owned_by	Foreign key to Jobs

Candidate Seen Job

Column	Constraints
uid	Foreign key to Candidate

jid	Foreign key to Jobs
liked	bool

Job Seen Candidate

Column	Constraints
jid	Foreign key to Jobs
uid	Foreign key to Candidate
liked	bool

Match

Column	Constraints
uid	Foreign key to Candidate
jid	Foreign key to Jobs