Employ.Me Design

Sabrina Drammis, Grant Gunnison, Sam Edson, Danny Sanchez

Motivation

Description [Sam]

Employ.Me is a web application to help employers find candiates and students find internships and jobs. We have a unique tagging system that allows users to easily search for results that fit their needs. ….. more ?

Purpose [Sam] need to get rid of the stuff about experience since they don't have tags

1. Help students discover companies and positions that they may have interest in.

Students have a set of experience, and Employers have a set of experience they are looking for. If we can find the intersection to these two sets in a way that is easy to use for both parties, we have created something genuinely useful.

2. Help employers find qualified applicants for their company.

Employers want candidates who can do the job well. To aid in finding these candidates, Students will create profiles that accurately portray their strengths, as well as credibilities to these strengths in the way of Classes and Jobs. Employers can then search for candidates based on the job's criteria to find suitable matches, and make introductions to potentially interview or hire a candidate.
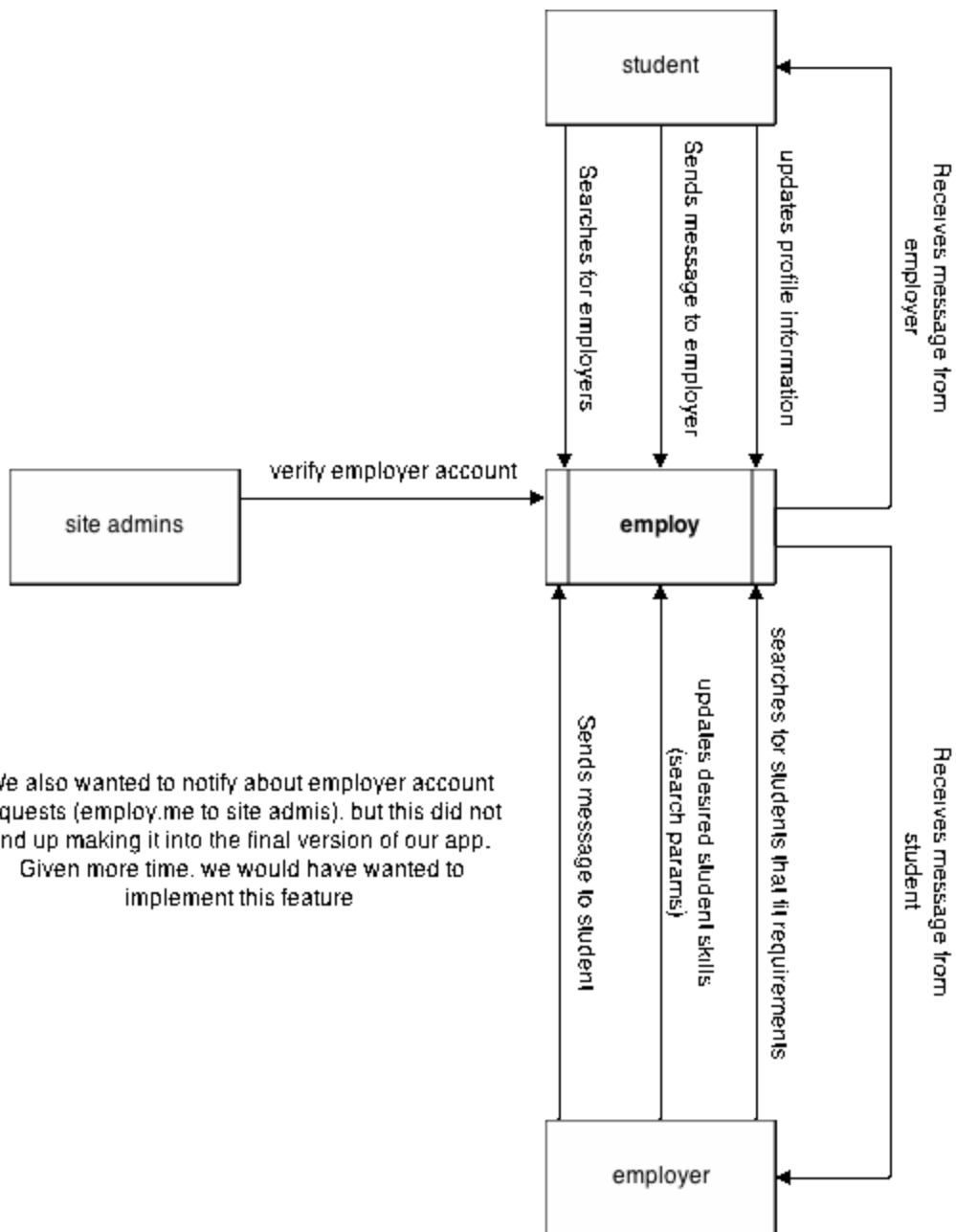
Existing Solutions
- Career Bridge
  - Deficiencies:
    - Interactions between candidates and employers are mainly placed under control of the candidate
    - Candidates may apply to a job opening thinking they have the right qualifications, but they may not be to the extent that they employer was looking for. Potential waste of time for both parties.
- Career Fair
  - Deficiencies:
    - Interactions between candidates and employers are mainly placed under control of the candidate
    - Candidates may visit a booth just because the company sounds cool, or because they want to see if they have a shot at a job with a company, without knowing much prior knowledge about the company beforehand.

This could potentially waste time for the both the employer, and the candidate.

- Candidates may not visit booths that they are good candidates for, either because they decided to leave the career fair, or they didn't know too much about the company and it didn't sound interesting, or whatever other reasons a candidate may have.

- Ann Hunter emails
  - Deficiencies:
    - Students filter emails because most of the employer related emails don't directly apply to them and it becomes a nuisance to sift through them all the time. Also, an employer must wait for responses from students. Therefore, they are probably missing out on students that would be a good match for their company and position they are looking to fill.
    - Employers can't send job information directly to students that they think fit their requirements.

Context Diagram [updated]

student

employ

site admins

verify employer account

Searches for employers

Sends message to employer

updates profile information

Receives message from employer

Sends message to student

updates desired student skills (search params)

searches for students that fit requirements

Receives message from student

employer

We also wanted to notify about employer account requests (employ.me to site admis). but this did not end up making it into the final version of our app. Given more time. we would have wanted to implement this feature

Concepts [sam]

- Tags
  - Each tag represents a skill that a candidate has. Tags can gain credibility through relevant classes taken. In this way, classes can relate to specific tags that the user has, so that the employer knows why the candidate has this tag, and how much the candidate was able to develop their proficiency with that tag.
  - Motivated by both purposes because both Students and Companies rely on the other having tags to use their functionality.
- Search
  - Allows Students to search for Companies matching their own Tags.
  - Allows Companies to find Students who have the Tags they are looking for.
  - Motivated by both purposes because both Students and Companies have a Search functionality.
- Listing
  - Motivated by both purposes 1 because it gives Students something to look for.
  - Without Listings, Students would search for larger Companies with several unrelated Tags that they are looking for. For example, GE hires just about any field of Engineering, so a Student from many fields would come to the same Company and have no basis for the Company's Tags.
- Profile
  - If an Employer is going through a bunch of Students that match its wanted Tags, the Employer would want to see some information about the applicant before it reaches out. We have found that Companies are reluctant to hand out interviews to people before they know any concrete details about them.
  - Motivated by purpose 2 because it gives Employers the power to manually filter the concentrated selection they get from searching.
- Translate Function
  - A way of determining a Tag based on a class. The user can input a class, and the translate function will determine the appropriate Tags associated with this class.
  - Motivated by purpose 2 because it allows Companies to verify Experience Tags.
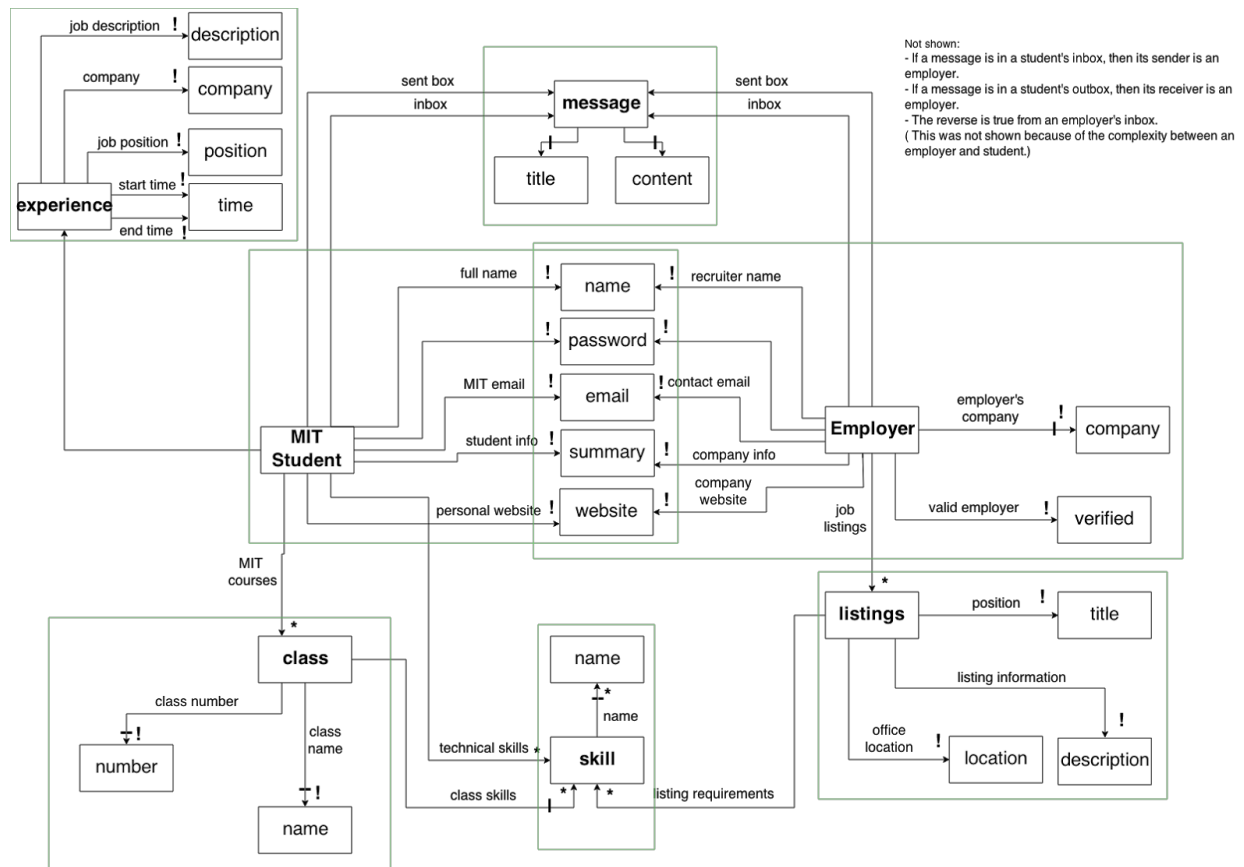
Data Model [updated]
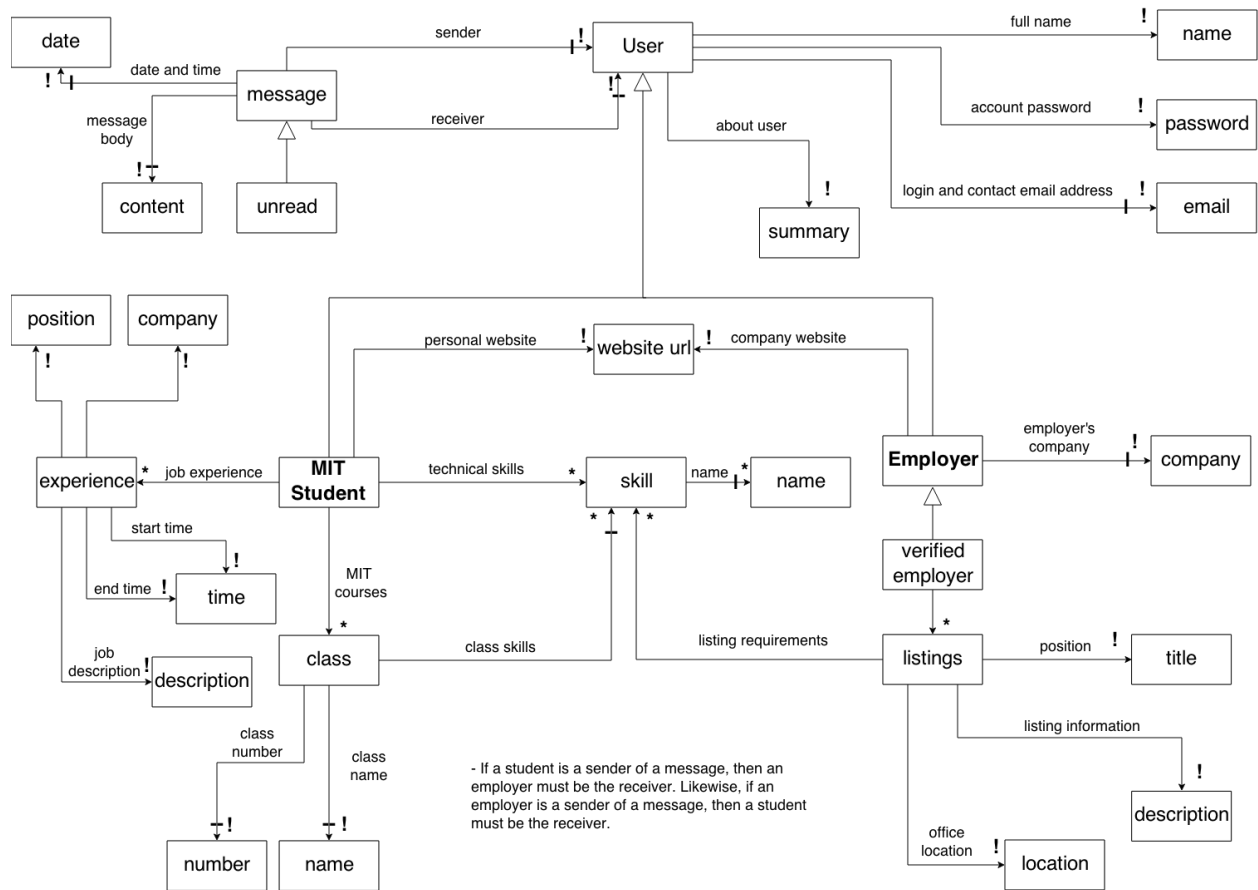
Pieces of our data model are not currently being used in our application. However, they are in place if we decided to work on this project further after 6.170.
For example, our application currently does not use employer verification, but this is something that would be necessary if we made our app live.

Data Design [updated]

experience

- job description **!** → description
- company **!** → company
- job position **!** → position
- start time **!** → time
- end time **!**

message

- sent box / inbox (left)
- sent box / inbox (right)
- title
- content

Not shown:
- If a message is in a student's inbox, then its sender is an employer.
- If a message is in a student's outbox, then its receiver is an employer.
- The reverse is true from an employer's inbox.
(This was not shown because of the complexity between an employer and student.)

MIT Student → Employer

- full name **!** → name **!** ← recruiter name
- **!** password **!**
- MIT email **!** → email **!** contact email
- student info **!** → summary ← company info
- personal website **!** → website **!** ← company website

- employer's company **!** → company
- job listings
- valid employer **!** → verified

class

- MIT courses → class *
- class number → ⊤ **!** number
- class name → ⊤ **!** name

skill

- name
  - ***** name ↑
- technical skills ***** → skill
- class skills ***** ↑ ***** listing requirements

listings *

- position **!** → title
- listing information → **!** description
- office location **!** → location

Security Concerns [updated]

- **Requirements:**
  - Only employers can search for students: In order to create and view a search for candidates, you must be logged in to the site under an employer account.
  - Only students can search for job listings: In order to create and view a search for job listings, you must be logged in to the site under a student account.
- **Standard attacks:**
  - XSS: Escape all relevant user inputs, just to be safe.
  - CSRF: Generate a secret CSRF token for each user session, and send as a hidden field with all forms. Be sure not to show token in URL. (we wanted to implement this, but we could not do it with our QUnit tests)
- **Threat model:** What the adversary can do:
  - If adversary can impersonate a company by acquiring an employer account, they can fool candidates into giving them more personal information
  - If adversary can view candidate profile, they might be able to send a message to the candidate (again impersonating an employer/recruiter), asking for personal information.
  - Adversary can script an attack on the site in an input field if not escaped properly
    - Student can write a script in a message to a company, and potentially edit the company profile
- **Content Security Policy:**
  - Scripts:
    - Allow from: code.jquery.com, 'self', ajax.googleapis.com
  - CSS:
    - Allow from: 'unsafe-inline', 'self', code.jquery.com, netdna.bootstrapcdn.com, fonts.googleapis.com

User Interface [updated]

Employer Login

Employ.Me

Employ.Me   Employers

Find the right undergrads   <u>Login</u>   <u>Sign up</u>

Search for students   Company Name
Find graduates tailored to you

Help students find you   Email
Show students how awesome you are

No more hassle, just results   Password
Greater Exposure. Streamlined search

                                                 Join Now

Student Login

Employ.Me

Employ.Me   Employers

Built by undergrads for   <u>Login</u>   <u>Sign up</u>
undergrads

Search for jobs   First Name   Last Name
Find positions tailored to you

Help companies find you   Email
Show employers how awesome you are

No more hassle, just results   Password
Exposure. Opportunities. $$$

                                                 Join Now

Student Profile View/Creation/Editing

Employ.Me   **Profile**   Search   Messages   Logout

Summary

Skills

Button	Button	Button	Button	Button	Button	Button
Button	Button	Button	Button	Button	Button	Button
Button	Button	Button	Button	Button	Button	Button

Experience

Position Name

Start Date

End Date

Company

Skill Tags

Description

Action
Action
Action
Action

Classes

Class Number

1. Class Number   Skill Tag   Skill Tag   Skill Tag   Skill Tag

2. Class Number   Skill Tag   Skill Tag

Save

Employer Profile View/Creation/Editing

Employ.Me

← → C ⌂ [   ] [🔍   ]

Employ.Me   **Profile**   Search   Messages   Logout

Summary

[   ]

Jobs

Position Name   Start Date   End Date
[   ]   [   ]   [   ]

Location   Skill Tags
[   ]   ┌─────────────────┐
                       │ Action   │
Description   │ Action   │
┌───────────────┐   │ Action   │
│   │   │ Action   │
│   │   └─────────────────┘
│   │
└───────────────┘

1.   Software Engineer   June - August 2015   San Francisco

      Description   [Tag 1]   [Tag 2]   [Tag 3]

General Info

Recruiter Name   Company Website
[   ]   [   ]

                                                    [Save]

Search

Employ.Me   Profile   Search   Messages   Logout

Input Label

Skills   🔍 Search

Required

[Button]
[Button]

Prefered

[   ]

[Button] [Button]
[Button] [Button]

Results

Joe Smith   Graduating: 2015

Skills:

[Software Development] [Web Development] [Databases]

Experience:

[Microsoft Software Design Intern] [Summer 2013]

[Send Message] [Go to Profile]

Patricia Tupel   Graduating: 2015

Skills:

[Computer Vision] [Java] [Concurrency]

Experience:

[CSAIL - Research] [Summer 2013]

[Computer Vision] [Summer 2012]

[Send Message] [Go to Profile]

Messages

Employ.Me   Profile   Search   Messages   Logout

Messages

Create
Inbox
Sent

Microsoft would like to interview you!
SunPower would like to offer ...
Thank you for applying to Google ....
Thank you for applying to Bain ...

Flow Chart [updated]
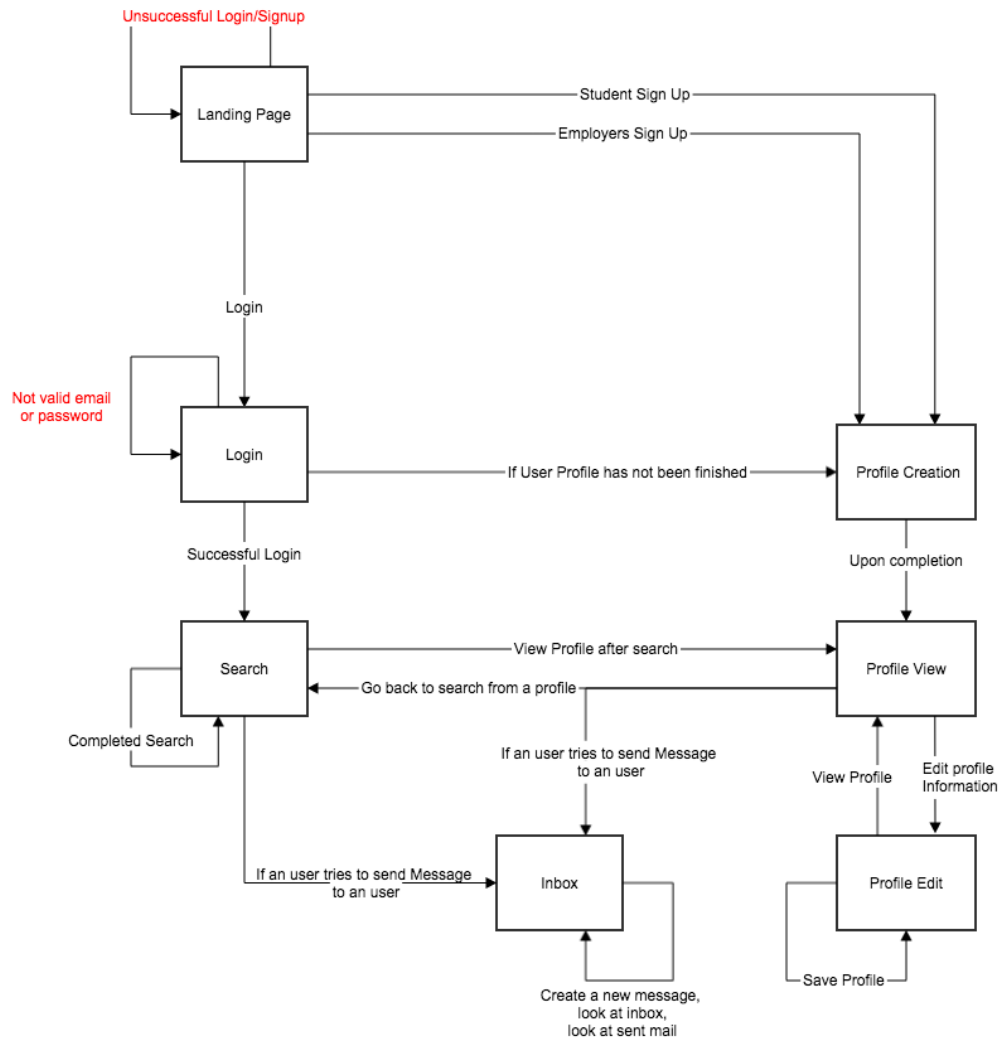
Design Challenges

Accurately Depict a Student's Abilities
Options:

     1. Require endorsements like LinkedIn does. Other students and employers would view your profile and endorse your skill tags so they are more reliable. The advantage to this is it is semi reliable if endorsements come from valid sources. Unfortunately, this approach can be gamed by people with fake accounts, friends, etc. Also, it will be difficult to implement, adds a lot of complexity to our app, and requires a larger community than we are going to be able to manage if we are restricting it to only MIT students.

     2. Use classes as proof of experience and trust that manually-added experience tags are valid because an interview would show the person was lying. The drawback to this is it could waste employers' times if a bunch of invalid accounts creep up. Later we could possible add a report feature.

     We choose option 2 for the reasons outlined in option 1.

Easy and Descriptive Profile
Options:

     1. Require the user to enter in Sports, Activities, Clubs, etc. Have that be the profile and use the tags for searching. The Employer can decide whether to contact the Student based on his profile that acts as a resume. This is not great because it takes so long to set up. Also, profiles are constantly changing.

     2. Have a general summary that the user can use to make a quick impression on the Employer. Keep sections for the most important attributes like previous jobs, which are also useful because they have Experience Tags associated with them. The drawback of this is that people with non-job related experience may feel strained for room. On the other hand, the app is meant to create introductions, so that type of information would be communicated after a formal communication has been established.

     We choose option 2 because it is more fitting with our app's purpose and makes for a better user experience.

Help Students Find Companies Based on Their Interests
Options:

     1. Don't let Students search for Companies. Only let them see the companies that have contacted them, making Students passive. This is bad because it removes a big incentive for Students to join.

     2. Have a sorted list of companies that users can look through. This isn't ideal either because it means we will need to sort the list based on some parameter. Most of the time this will favor some companies over others in some strategic manner, while we want Students to *match* with a company suitable to their skills.

     3. Have an Employer search like there is Student search with tags as well. Companies will set tags based on what experience they are *looking* for. That way, Students can use their

own tags, or search on other tags, to see what companies they match up with. The only drawback is that it requires more setup on a company's part.

We choose option 3 because it allows for the most flexibility for our users. It also gives an incentive for Students to interact with the site, where before they were just passive users that got contacted.

Validate Companies

Options:

1. Build a system for automatically validating a company. We couldn't come up with a feasible way to do with within the scope of this project.

2. Email a moderator when a Company signs up. Allow the company to work while it is unverified but shut it's account down if it fails the verification process. This requires little work.

We choose option 2 out of necessity. Building an automated system to do this would be extremely difficult without requiring a lot of work on the Company's part to prove itself.

Messages

Options:

1. Messages as a separate collection. A message will have a sender and receiver that will contain a user id. This approach results in a message being stored in only one location. However, retrieving messages by a given user id will take time.

2. A user has a list of sent message subdocuments and received message subdocuments. This means that there will be two copies of a message in the database, one for the receiver and one for the user. It will be quick and easy to get the received or sent messages for a given user.

We chose to use option 2 because space is not a big concern as of now. It is also the easiest approach as it does not require any joins or extra hits on the database.

Client side rendering

1. Use handlebars. Some of the team is familiar with handlebars and has used it in the previous project. However, no one is entirely and expert with using it.

2. Use jade. Some team members are familiar with jade. Rendering client side templates can easily be incorporated into a Browserify build using Jadeify.
   Because we are using Browserify to require node modules on the client side, it made sense to do client side templates with jade. It easily integrates with our build system, automatically compiling files. However, it requires that some group members learn jade that had not used it before. Likewise, if we chose option 1 there would still be a learning curve for other members of the team.

Search creation

1. Drag and drop without autocomplete: would not require us to pass any data from jade into the client side model, but since there are so many skills, the user would need to take more time to find skills to drag and drop

2. Use of angular search with drag and drop: Requires us to pass data from jade into client side model, but when we do, user can search for skills, and then drag and drop the results much quicker. So now instead of scrolling in search of skills, the user can simply search for them. Because there will be no sensitive data being passed into the client side code, this is the preferred option.

We have currently implemented option 1, but we have chosen to use option 2 for our final product.