

TalentFront - Milestone 2

(Team 5)

CSC 648/848 Software Engineering – Spring 2021 Section 02

More Detailed Requirements, Specs, Architecture, UI
mock-ups and Vertical SW prototype

March 9, 2021

Austin Wimberley (awimberley@mail.sfsu.edu) - Team Lead & Github Master &
Backend Lead
Battulga Tsogtgerel - Frontend Lead
Ezra Player - Frontend
Utkrisht Sharma - Fullstack
Alfonso Duarte-Sarabia - Backend

History Table:

Date	Action
March 9, 2021	Submitted for review
NA	Date revised/finalized

1. Functional Requirements - Prioritized

Talent

- 1) Sign up - Create account. The site will require that users sign up in order to make a profile or apply to positions. The username, salt, and hashed password should all be saved to the database.
 - Priority: 1 - must have
- 2) Returning users should be able to log into previously created accounts. This will require the frontend to send the username and password, which will then be verified securely using the username and hashed and salted password.
 - Priority: 1 - must have
- 3) After logging in/registering session information should be stored in the client, so that state/information is persisted to their profile. The session credentials should only be valid during use of that session, and should require re-login after periods of not using the site,
 - Priority: 1 - must have
- 4) Talent should be requested to enter standard information: name, school, major, and highest degree completed or expected graduation date. This data is most important in order to allow for searching the applicants for a given position.
 - Priority: 1 - must have
- 5) Talent can further expand on their skills by listing: technologies they are experienced with and evaluating their familiarity. This data should also be queryable and searchable when looking for applicants for a given position.
 - Priority: 2 - desired
- 6) Talent should be able to add personal information, such as pictures, a short bio, and job experiences. This personal information should all be publicly displayed on the user's profile, so that recruiters can get a better sense of if the talent is suitable for their open positions.
 - Priority: 2 - desired
- 7) Talent should be able to search for job postings. Talent should be able to search by location, expected pay range, job title, and technologies required for the position.
 - Priority: 2 - desired
- 8) Talent should also be able to apply for job postings, where they can upload their relevant experience, link to their profile and/or send a resume. When applying to

a position it should send a message to the job poster with all of the info that the talent provided.

- Priority: 2 - desired
- 9) Talent should be able to search their peers and view their public profile information. This will allow them to get a better understanding of how others are making use of the service.
 - Priority: 2 - desired
- 10) Talent should be able to follow other users, whether they be talent, companies, recruiters in order to follow changes to their profiles, job statuses, and recruitment opportunities.
 - Priority: 3 - opportunistic
- 11) When a talent views a posting it should display the distance approximation from their home, which will more easily help the candidate make decisions which will affect their commute time.
 - Priority: 2 - desired
- 12) Talent should be able to rate employers that they work for on a scale anonymously, and see an aggregate of everyone's ratings. This will allow other talent to make more educated decisions when choosing which jobs to apply for.
 - Priority: 3 - opportunistic

Professors

- 1) Professors should be able to create specialized accounts which will be verified by using a correct email's domain. This will prevent people from signing up as professors with bad intent.
 - Priority: 1 - must have
- 2) Professors will be able to rate students in a scale from 1-5 fashion, being 5 being the highest ratings that implies knowledgeable, responsible, teamwork, leadership, committed to success, etc. and enter recommendations. This will give employers more information about students, which will help them to make hiring decisions.
 - Priority: 1 - must have
- 3) Professors should be able to recommend students for a position that they think students are a good fit for. This will help both employers and students to expedite the job/candidate searching process.
 - Priority: 3 - opportunistic

Employers

- 1) Create job postings for their company. Students then will be able to apply for their job postings.
 - Priority: 1 - must have

- 2) Create organizations that members can join/follow. This will allow employers to post the latest news about their company. This can be their new feature launch or news article mentions.
 - Priority: 2 - desired
- 3) Employers should be able to enter their company profile information: Product/Services they offer, company mission, number of employees, photos of office, office locations, brief introduction of company. This will give students a good understanding about their company.
 - Priority: 1 - must have
- 4) Search through talent for prospects with appropriate qualifications. When there are a lot of applicants, this feature will allow employers to find the right candidate in a faster and accurate way.
 - Priority: 1 - must have
- 5) Administrator capabilities to trigger the matching alerts to the companies and alerts to the students to get ready for interviews. This will allow companies to find potential good hires for their existing postings without missing them or losing time.
 - Priority: 3 -opportunistic

Not Registered Users

- 1) Users who don't want to register, should still be able to view talent and job listings. This will allow people to search jobs without being pushed to sign up first. This feature will allow the site to grow more.
 - Priority: 1 - must have

2. UI Mockups and Storyboards (high level only)

We created demos for all major use cases using Figma. All demo screenshots below have more screens in the Figma directory, which is accessible through this link:

<https://www.figma.com/file/waLxNwz0zWPmPPMJhFzBp0/Mock-up?node-id=0%3A1>

For the sake of simplicity and use of ease, we made design as minimalistic as possible. Terms used in the demo are from database tables and column names. This allows developers to know where the data is coming from.

Sign up demo:

Login Talent

Software Engineer, Marketing, Designer... Search Username Home Logout

School

First Name

Last Name

Degree

Location

Back Enter

Professor login demo:

Login Professor

Software Engineer, Marketing, Designer... Username

School

First Name

Last Name

Degree

Location

Login demo:

Login 1

Software Engineer, Marketing, Designer... Username

Email

First Name

Last Name

Password

Confirm Password

Indicate Type of User [employer, employee, etc]

Employer login demo:

Login Employer

Software Engineer, Marketing, Designer... Search Username Home Logout

Company Name


Location

Allready Have an account? Enter

User profile demo:

Talent Landing Page

Software Engineer, Marketing, Designer... Search Username Home Logout



User -> email/username
User_Type -> student/professor
Expected graduation date
Degree Program

Edit Profile

Skill set

- UI design
- Marketing
- C++ Programming
- Public Speaking
- React
- Python

User_Info -> description

User_Experience -> job_title User_Experience -> date_start User_Experience -> date_end

User_Experience -> experience

User_Experience -> job_title User_Experience -> date_start User_Experience -> date_end

User_Experience -> experience

Job search demo:

Job Postings Page

Software Engineer, Marketing, Designer...

Search

User -> first_name

Home

Logout

Sort By:

Salary Range

Date Start

Experience

Filter by word...

go

posting -> job_title

posting -> salary_range_bottom

posting -> salary_range_top

posting -> Description

posting -> employer_name

posting -> experience_required

...

posting -> created_dt

Apply Now

posting -> job_title

posting -> salary_range_bottom

posting -> salary_range_top

posting -> Description

posting -> employer_name

posting -> experience_required

...

posting -> created_dt

Apply Now

posting -> job_title

posting -> salary_range_bottom

posting -> salary_range_top

posting -> Description

posting -> employer_name

posting -> experience_required

...

posting -> created_dt

Apply Now

posting -> job_title

posting -> salary_range_bottom

posting -> salary_range_top

posting -> Description

posting -> experience_required

...

Apply Now

Employer job postings demo:

Employer Landing Page

Software Engineer, Marketing, Designer...

Search

Username

Home

Logout

New Posting

Filter with word

Sort by date

Sort by number of applicants

Sort by engagement

UX / UI Designer

Number of applicants: 234

Posted Date: 03/06/2021

Job Description:

Description Description Description Description
Description Description Description Description
Description Description Description ...

Legal Consultant

Number of applicants: 49

Posted Date: 01/23/2021

Job Description:

Description Description Description Description
Description Description Description Description
Description Description Description ...

Strategic Analyst

Number of applicants: 113

Posted Date: 03/06/2021

Job Description:

Software Engineer

Number of applicants: 123

Posted Date: 03/06/2021

Job Description:

Description Description Description Description
Description Description Description Description
Description Description Description ...

Product Manager

Number of applicants: 56

Posted Date: 02/13/2021

Job Description:

Description Description Description Description
Description Description Description Description
Description Description Description ...

Director of Sales


Number of applicants: 113

Posted Date: 03/06/2021

Job Description:

Talent/Professor profile demo:

Talent/Professor Edit Profile



Skill set

- UI design
- Marketing
- C++ Programming
- Public Speaking
- React
- Python

user_info -> description

User_Experience -> job_title

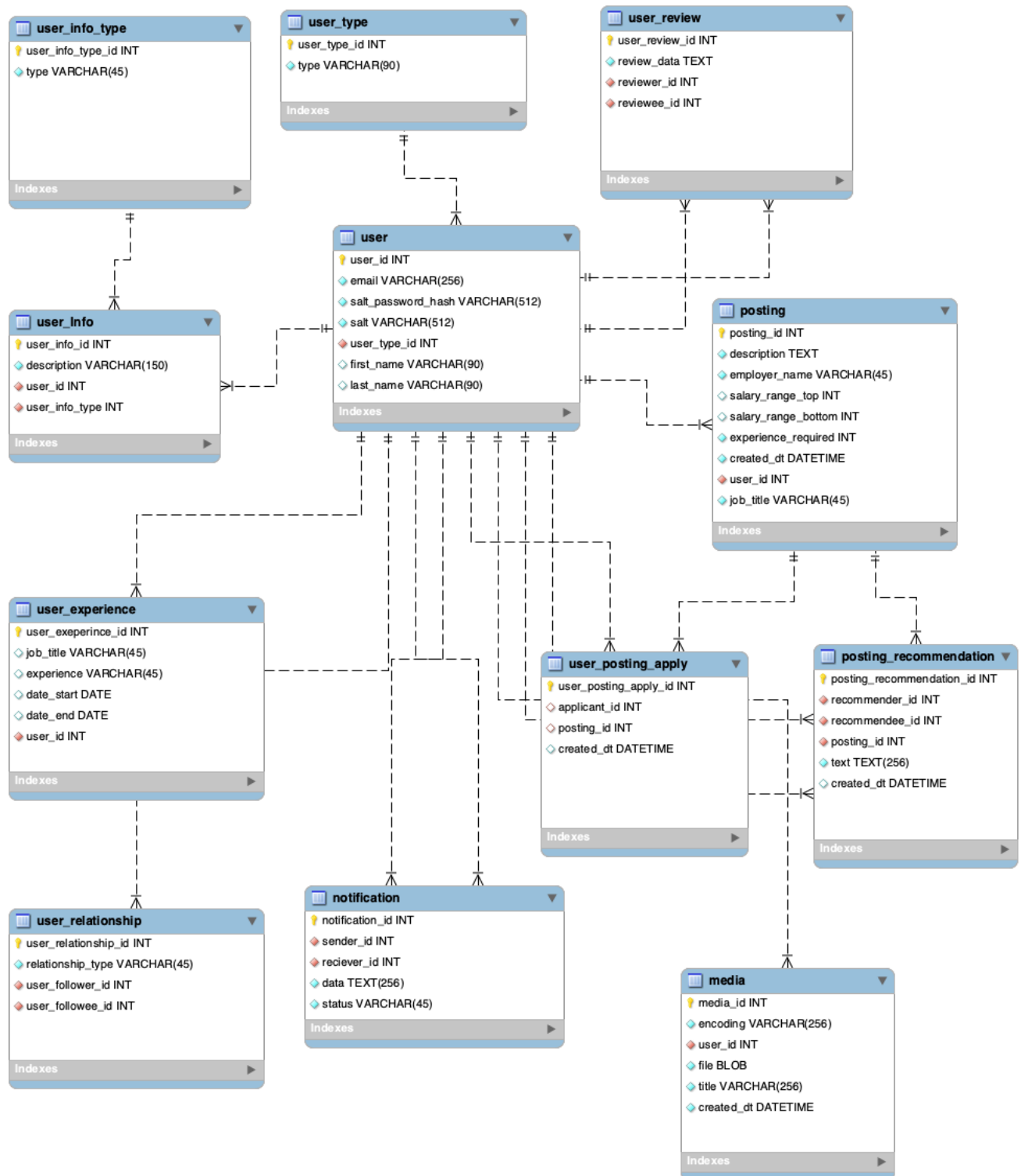
	Start Date	End Date
<ul style="list-style-type: none"> - Did a great job there. - Grew company's revenue by 10% within a year. 		

Job Title

	Start Date	End Date
<ul style="list-style-type: none"> - Did a great job there. - Grew company's revenue by 10% within a year. 		

3. High Level Architecture, Database Organization

Database Organization / Schema



Media Storage

We have decided to use the BLOB data type in MySQL in order to store our media files. This is new territory for us, so we might have to change the schema for this table, but hopefully with the encoding column it will tell us how to play/serve the media files stored here.

Maybe use BLOBs since we have admin and access control, has transaction security, and easy to manage access by many users. When storing image paths in a database column we can use relative paths and maybe use universal unique identifiers.

Search/filter architecture and implementation

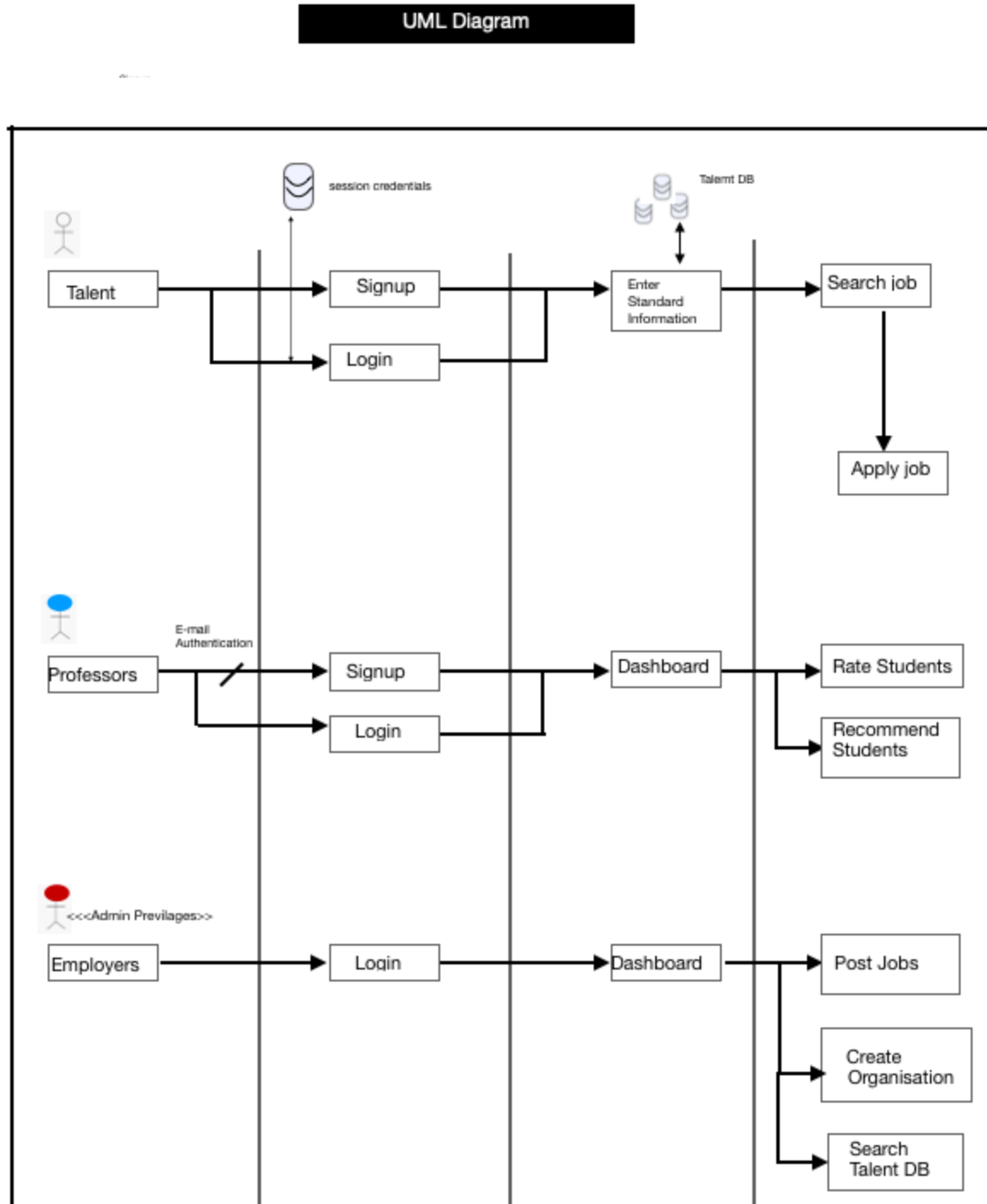
Using SQL and %like, we will implement a search/filter algorithm. The like operator is a logical operator where that tests whether a string contains a specific pattern. The search algorithm will use limits and seeks to paginate the results. It depends on the input of the client to define what tables will be searched. A global search will kick off multiple table scans, which include user_experience, user_info, user_review and posting. The client should be able to filter for only job postings, which will query the posting table. The client should also be able to filter for talent, which will filter what user_type to run a search on.

Our own APIs

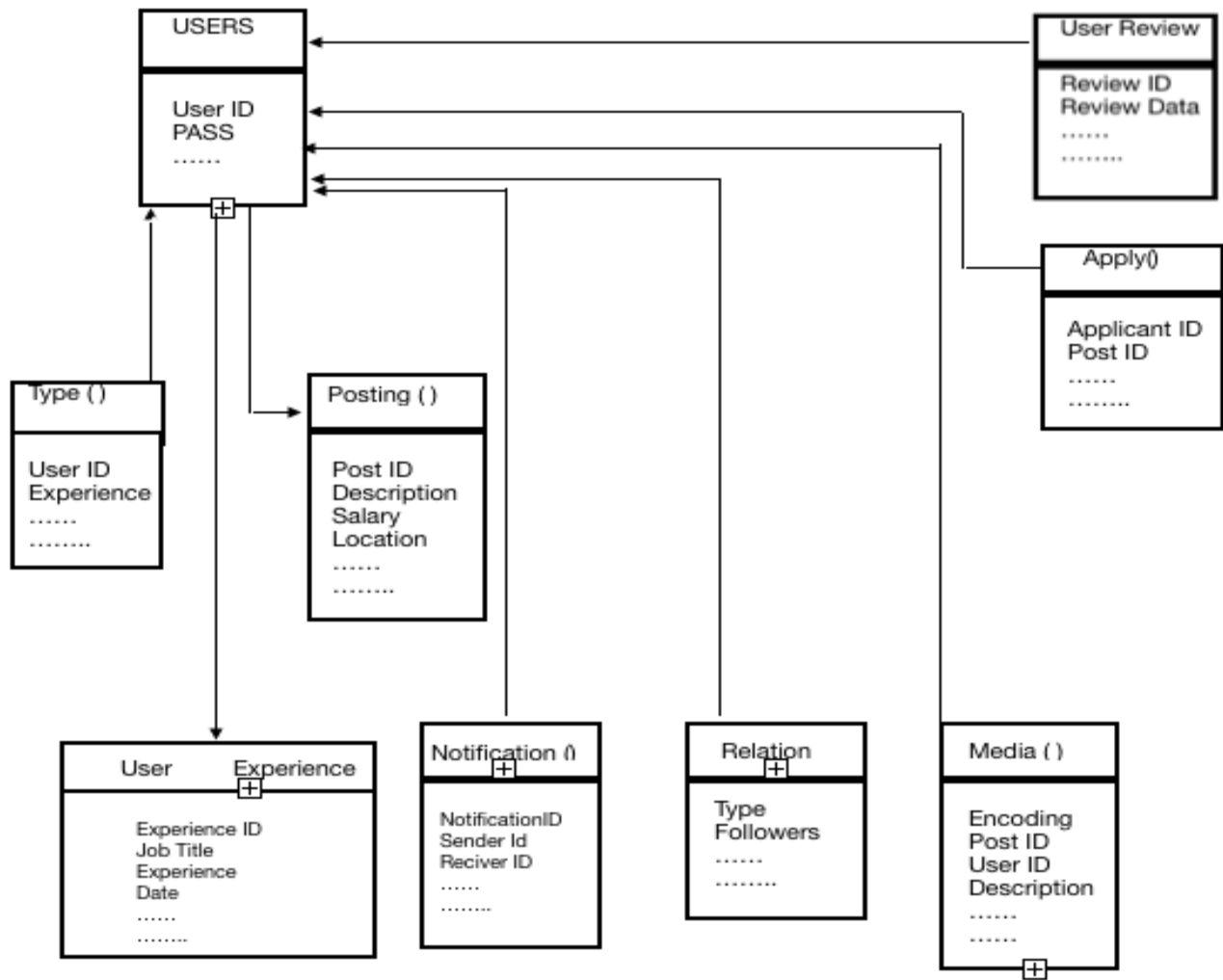
The search will be sent to the backend in a POST request. We will have to define the exact schema on the request, but it should include the search terms and the type of results the client wants to see, whether it be postings, talent, companies etc. The request should return a data array with up to a set limit of items, it should also return a url for the next page of results.

4. High Level UML Diagrams

Use case diagram:



Class Diagram



5. Key Risks

Skills risks:

There is a wide diversity in the risks of the team. There are a bunch of technologies that we will have to harness in order to get everything wired together and working.

- 1) Frontend: The only member with any significant React experience is Battulga. While other members such as Austin, have experience in modifying some React apps he his overall experience is pretty limited. The other members have also had brief experiences in standing up some minimalistic React servers, but there will be a big weight to carry as far as setting up the apps and creating the teams best practice in regards to how the apps should be built. In the development process, we plan to start React app as simple as possible and avoid all unnecessary complexities. As our overall project grows, we'll be bringing more complex packages and structures changes one at a time.
- 2) Backend: The only member with any experience in building Spring Boot applications is Austin. It will be up to him in order to create strong best practices that the other backend team members can use as templates for completing their work.
- 3) Nginx: Austin is the only member with experience configuring reverse proxies so a lot of the work will fall onto his shoulders in order to establish best routing procedures.
- 4) MySQL: While some members have taken database classes, setting up the database to work in a production environment is a bit different. There will be a bit of a learning curve in order to set up a deploy pipeline to properly spin up the MySQL image with applying the proper Flyway migrations.
- 5) Overall: I think if all the team members could spend enough time ramping up their skills in the areas that they need to learn more about, this project could be done properly. However, the concern is about how many engineering hours and mentoring can be devoted to this project in order to meet all the requirements.

Schedule risks:

This is something that we are a bit worried at as a team. While we do feel like as a team we have the engineering skills necessary to meet the requirements. Some of the team members have full time jobs and it is hard to dedicate solid chunks of focused

engineering team outside of these responsibilities. Also it is a bit rough to collaborate because we have team members on the other side of the world, so scheduling times to meet can be tricky. In order to solve these scheduling risks we will have to be extra diligent in communicating and working asynchronously, which has the negative of making knowledge sharing problematic. This team will have to be extra diligent in writing strong tickets and holding team members responsible for meeting deadlines.

Technical risks:

One of the big standouts that none of us have experience in implementing is having a working logged in state. While we understand the general concept of how to pass login credentials to the backend to validate, and protect passwords on the backend. We will need to learn about how to implement session credentials and validate them possibly with a Redis instance.

Another field that none of the team members have experience with is webmail requirement. We haven't included it in our requirements because there still needs to be a more in depth discussion on what we want to achieve by using webmail. I am confident that with the proper time dedicated to implementing this feature, we could get it live. The risk I see is that it is hard to estimate the amount of lift it will require to get this working since it is a new technology to everyone. A first step will be to do a spike investigation into the technology in order to better understand its value proposition, the amount of work required, and where would make the most sense in order to leverage its value.

Teamwork risks:

As the tech lead for the team, Austin feels hesitant about how much time he can devote to this project. While he has contributed more than his fair share in terms as an individual contributor. When it comes to the added work of writing tickets, distributing work, setting expectations and ensuring deadlines are met, Austin feels that the team might be better served with electing someone else to take on the Project Management responsibilities. We are currently discussing if anyone feels comfortable with taking those on. Followup: Alfonso

Legal risks:

We faced a dilemma in how we thought to address some of the given use cases allowing recruiters to filter candidates by protected classes. From the Equal Employment Opportunity they state:

Applicants, employees and former employees are protected from employment discrimination based on race, color, religion, sex (including pregnancy, sexual

orientation, or gender identity), national origin, age (40 or older), disability and genetic information (including family medical history).

From a legal standpoint, we feel that offering recruiters the ability to filter candidates by their protected classes violates EEOC's rules and would open our product up to lawsuits. From a moral standpoint, there are many questions as to whether we should be offering this functionality even if it was legal. You said this can be used in a right and wrong way. However, if looking at demographic breakdowns of software the most overrepresented group would actually be Asians. Would it be ethical to filter out Asian candidates based upon their representation in the field? For many reasons, our team made the decision to follow the law and will not be providing a service allowing employers to discriminate based on protected classes.

Content risks:

We will be using open source technologies that allow for the use in this educational purpose. React, NodeJs, OpenJDK, and SpringBoot are all open source, so we don't see possible risks with using these technologies.

As far as the multimedia aspect of the program, this is something that will require a bit more scrutiny. By providing a platform where users can upload content, we run the risk of users uploading copyrighted media. We do not have the engineering/human resources to be able to vet each media upload for possible copyrighted material. However, we should put it on our roadmap to build some sort of functionality to allow concerned parties to request copyrighted material be taken off of our platform.

6. Project Management

Overall (Alfonso, Project Manager)

For milestone 2, we separated the majority of the work between our backend and frontend teams. The assignments were as follows:

1. Functional Requirements -- Austin and Battulga
2. UI Mockups and Storyboards -- Ezra and Battulga
3. High level Architecture, Database Organization -- Austin & Alfonso
4. High Level UML Diagrams -- Utkrisht (In progress)
5. Key Risks -- Austin
6. Project Management: Assigned to leads/project manager
 - a. Alfonso -> write up overall
 - b. Austin -> write up backend process
 - c. Battulga -> write up frontend process

Battulga and Ezra focused on doing the UI mockups and storyboards. Austin and Battulga focused on prioritizing the functional requirements and Austin also focused on identifying the actual risks for the project. Alfonso and Austin managed the database organization and on how we were going to handle media storage. Utkrisht focused on the high level UML diagrams.

In order to efficiently finish all our tasks we used Trello for task management. We divided up the work and everyone easily saw what they needed to work on. We plan to use Trello in future milestones to help each team member understand what their tasks are, organize the flow of our project, and keep everyone in check. When checking the progress of each task, other members will be able to help those who have fallen behind or need extra assistance. It is important that both backend and frontend teams know where they are and what is needed of them. We will schedule weekly meetings through Zoom. Frontend and backend teams will meet and work on their respective parts of the project. When a deadline is approaching we will all have a meeting together to discuss what is needed to finish the milestone and discuss any issues or concerns. Austin, as a team leader, will be the one to schedule the meetings and the one that will keep us informed of any issues regarding our project. Alfonso will have a significant role in project management as well.

Back End (Austin, Back End Lead)

While working with the project manager, I think it is important to write up detailed tickets so that we have strong visibility into what others are working on. With respect to

the backend, having the API protocols defined as requirements for completion acceptance is important. While the frontend is going to be heading in a more microservice architecture direction, I envision the backend infrastructure to be more monolithic. By only having one backend service, there will be less time spent configuring the service and we can move quicker in instrumenting the endpoints required.

I (Austin) and Alfonso have worked closely together throughout M1 and M2 in order to build up the data requirements. This will have a large impact in streamlining the instrumentation process as we write tickets, and standup the backend and datastores.

Front End (Battulga, Front End Lead)

Frontend team will be building the application one feature at a time and the development process will have 4 steps. Developers will pick features they want to work on and if a feature is big in terms of coding or time consuming, two developers will work together on the feature.

1. **Designing:** Figma designs will be created and consulted with team members. Once finalized, a Trello card with screen shots of the feature will be created.
2. **Building UI:** Build UI React components in a new git branch and integrate it with existing app components such as Redux, React Router and CSS Transition/Overlay. Branch name, assigned developer and timeline(or deadline) will be added to the Trello card.
3. **Integrating Backend:** Add data validation and make API calls to the backend service. Communicate with the backend team regarding any questions during this phase.
4. **Test & Deployment:** Test the newly created feature. Distribute across team members and ask them to test. Once all testing is done merge the branch into the main branch. This feature will now be deployed to the users in the next scheduled update.