CSC 648/848 Software Engineering – Spring 2021
Section 02

Milestone 1
Use Cases, High Level Requirements and Architecture

February 23, 2021

Team 5

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
February 23, 2021	Submitted for review
March 9, 2021	Date revised/finalized

1. Executive Summary

TalentFront aims to provide a marketplace for employers to gain deep insights into the prospects that San Francisco State University is producing, targeting roles inside of the technology space. It's founded to be an easily searchable marketplace of upcoming engineers, UI/UX designers, and program managers. We will provide value both for upcoming talent and employers, as we help talent navigate job openings, and help companies find the right candidate for the job.

One of our key advantages over the competition is our close ties with the faculty of San Francisco State University. Professors will be given specialized accounts where they can provide insights into the values that different students can bring to the table of any prospective company. They will be able to write current/former students recommendations or reviews. Students can also designate professors that they have taken classes with, which allows prospective companies the ability to request feedback as they make hiring decisions.

Students would be able to flesh out their profile page which will live as a working resume with the added functionality that these profiles will be standardized allowing for easy querying to suit any recruiters specific needs. They will be able to list off skills and experience in an organized fashion. These skills shall include technologies such as programming languages, frameworks and various other technologies that recruiters can look for. The prospects experience will include durations, and technologies used, allowing recruiters to specifically look for candidates with the appropriate qualifications.

At the other side of the hiring equation, we will also provide companies a robust job posting platform. Companies will be able to post open roles, the requirements they are looking for in a candidate, as well as location and salary ranges. All of this will then be easily searchable by job-seeking candidates, whether they have a profile on our platform or not.

Our team boasts a modern technology stack with experienced developers. We are using an industry standard frontend framework:Node,React. Heading up our frontend we have Battulga Tsogtgerel, who has professional experience working on the framework. Heading up the team as well as backend development is Austin Wimberley. He has chosen to use the Spring framework, which has become the industry standard for Java development. At the beginning of the company, we focussed heavily on maximizing developer productivity so that each developer is empowered to deploy code to production painfree.

2. Personas and main Use Cases

Main Personas:

- Students
    - Tech savvy, can figure stuff out as long as it's not frustratingly bad.
    - Motivated to find a well paying job.
    - Moderate to high WWW skills.
- Talent Acquisition Departments
    - Hired to find the best employees for a company.
    - Basic WWW skills.
- Non-profit Organizations
    - Wants to connect its members with job opportunities.
    - Basic WWW skills.
- Employee Resource Groups
    - Wants to help their company find employees.
    - Voluntary, employee led.
    - Basic WWW skills.
- Professors
    - Wants to either help really good students look good on their applications, or flame students they absolutely detest.
    - Basic WWW skills.

Use Cases:

- 1. Student
    - 1.1 First visit
        - Student want to look for potential jobs. He goes to our site, searches "Computer Science Jobs" and finds the job he likes based on a light description. He clicks on a link to that job's page, and finds a more detailed job description. He decides to apply, and is prompted to make an account, which sends him to a register page to fill out his information. After registering, he is sent to a page to answer any specific questions the company might want included in the application. Then, he is free to continue applying to jobs.
    - 1.2 Subsequent visits (standard)
        - Student wants to check on jobs they've applied to. They go to our website, and click a button to go to their dashboard. On the dashboard is the jobs they've applied to through the website, along with information related to their standing in the application, and tools to organize his applications.
- 2. Employer (Jeff)
    - 2.1 First Visit
        - Jeff wants to find employees for his company. He goes to our site, and clicks on a button that asks if he wants to set up a job posting, which sends him to a page detailing the benefits of posting a job on our website.

After reading, he clicks a button that sends him to a registration page to register his company. After registering his company, he is sent to a dashboard page which details the jobs his company has posted (none so far), current applicants for each job with their respective information (again, none so far), a button to create a new job posting, and some tools to manage his job postings.

    - 2.2 Subsequent Visits
        - Jeff wants to check on his job postings. He goes to our website, clicks a button to login, and after filling in his information, he is sent to a dashboard page which details the jobs his company has posted, current applicants for each job with their respective information, a button to create a new job posting, and some tools to manage his job postings.
- 3 Professor
    - 3.1 First Visit
        - Professor wants to give a rating to one of his students on our website. He goes to the site, and clicks a button that asks if a professor wants to rate a student. He is sent to a register page, and after filling out his information, is sent to a dashboard that details the students he's rated, and students he hasn't rated, but have listed him as their professor. From here, he selects a student with a button, and is prompted to give a rating and comment on them.
    - 3.2 Subsequent Visits
        - Professor wants to give another rating to a student. He goes to our site, clicks the login button, is sent to a login page, and after filling out his information, is sent to a dashboard that details the students he's rated, and students he hasn't rated, but have listed him as their professor. From here, he selects a student with a button, and is prompted to give a rating and comment on them.

3. List of main data items and entities

Tables

1. User (email/username, password_hash, user_type)
   This table will hold the attributes of the main users like their email, password, and what type of user it is whether it be student, teacher, non-profit organization, employee, research group, talent acquisition, etc.

2. user_information(user, type, description)
   It holds the important information of the user like what type of user and description that is displayed on their home page.

3. User_information_type(school/degree/experience/location/etc)
   It holds user information for SFSU students like what school they attended, what degree they have, where they are located, etc. This will be important for employers to see.

4. User_type (talent/professor/recruiter/company)
   It separates the user by talent, professor, recruiter, or company which are all the different types of users. People will have the ability to choose when they create their profile.

5. User_experience (job title, experience, date start, date end)
   Holds the job experience items like the job title, the day they started, and last day at the job. SFSU students will be able to post past job experience so it is important that we remember those data items.

6. User_relationship(user follower id, user followee id, relationship_type)
   Users will be able to follow each other so there must be some type of way to recognize the data items that represent those such as the id of the follower and followee and what type of relationship it is.

7. User_review(user reviewer id, user reviewee id, review_data)
   Users will have the ability to review other user's posts and such. We will keep track of their ids and their reviews.
8. Posting(poster id, description,employer name,salary range, experience required,location,date start-end)
   Employers will have the ability to do job posting so we need to keep track of the job description, who the employer is, and the start date of the job.

9. Recommendations(posting id, description, user recommended id, user recommender id)

    Users will have the ability to recommend other users with recommendations like job posting or other people's post. We need to keep track of whom the post belongs to and then we need to keep track of who is recommending the post and to whom he is recommending to.

10. Notification(sender id, receiver id, description/data, notification_type)

    Users will have the ability to send notifications to other users. We need to store the id of both the sender and receiver, what exactly the notification is, and what type of notification it might be whether it be an alert or message.

4. Initial list of functional requirements

- Talent
    1. Sign up - Create account. The site will require that users sign up in order to make a profile or apply to positions.
    2. Returning users should be able to log into previously created accounts.
    3. After logging in/registering session information should be stored in the client, so that state/information is persisted to their profile.
    4. Talent should be required to enter certain required information: name, school, major, and highest degree completed or expected graduation date.
    5. Talent can further expand on their skills by listing: technologies they are experienced with and evaluating their familiarity.
    6. Talent should be able to add personal information, such as pictures, a short bio, and job experiences.
    7. Search for job posting. Talent should be able to search by location, expected pay range, job title,
    8. Talent should also be able to apply for job postings, where they can upload their relevant experience, link to their profile and/or a resume.
    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.
    10. Follow other students & alumni.
    11. Distance approximation from their home.
    12. Talent can rate employers on a scale, and see an aggregate of everyone's ratings.
- Professors
    1. Professors should be able to create specialized accounts which will be verified by using a correct email's domain.

    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.
    3. Professors should be able to recommend students for a position that they think students are a good fit for.
- Employers
    1. Create job postings for their company.
    2. Create organizations that members can join/follow.
    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.
    4. Search through talent for prospects with appropriate qualifications.
    5. Administrator capabilities to trigger the matching alerts to the companies and alerts to the students to get ready for interviews
- Not registered users
    1. Users who don't want to register, should still be able to view talent and job listings.

5. List of non-functional requirements

- Passwords should be protected and not stored in the database in plain text. They should only be passed to the backend in secured POST messages, and saved/checked using salt and hashing standards.
- There must be a documented term of service that clearly defines how the user's personally identifiable information (PII) is stored and shared.
- Everything should be in English
- The site must make use of Google Maps and Google Analytics.
- Application must be optimized for desktop/laptop and must render correctly on the latest version of two major browsers.
- Application UI shall be easy to use and intuitive.
- Selected application functions must render well on mobile devices
- Application shall be developed, tested and deployed using tools and servers approved by class CTO and as agreed in Milestone 0.
- All data should be stored in the database on the team's deployment server.
- No more than 100 concurrent users shall be accessing the application at any time
- Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated in UI.
- Site security: basic best practices shall be applied (as covered in the class)
- Modern Software Engineering processes and practices covered in the class will be used, including collaborative and continuous development/deployment.

- The website shall prominently display the following exact text on all pages "SFSU Software Engineering Project CSC 648-848, Spring 2021. For Demonstration Only" at the top of the WWW page. (Important so not to confuse this with a real application).

6. Competitive analysis

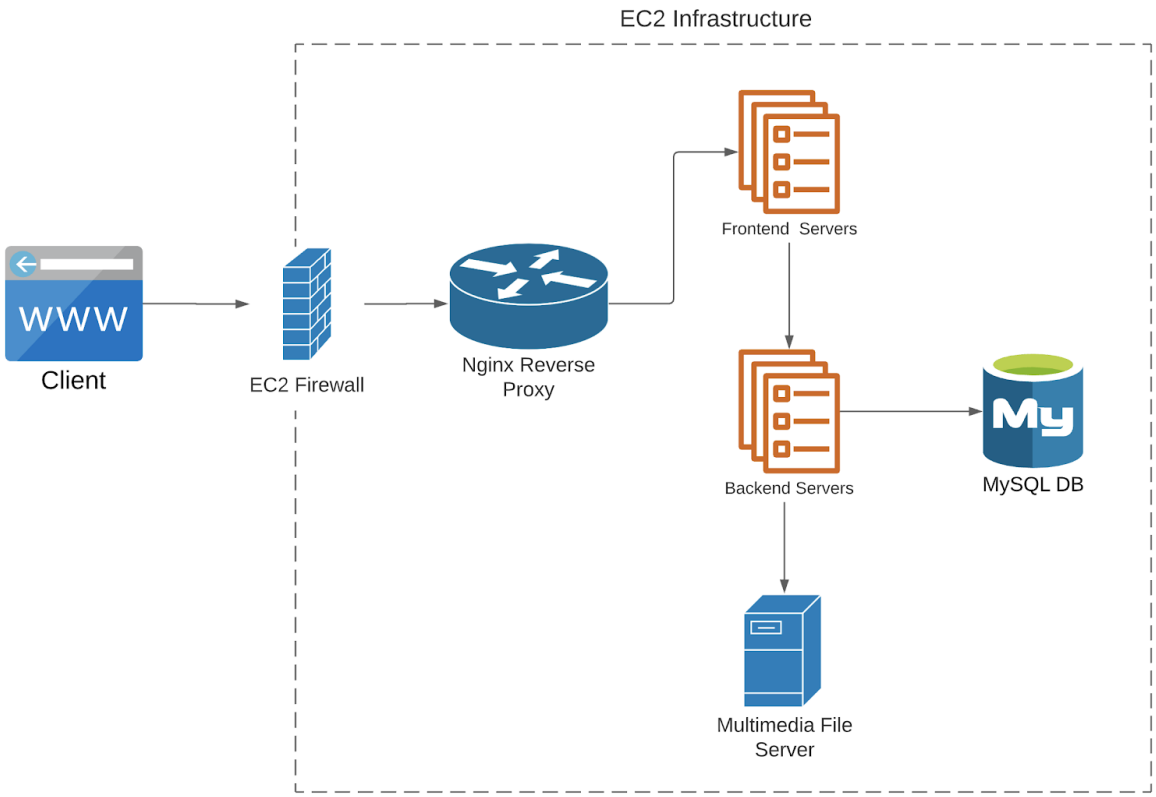
FEATURE SCORE MATRIX:

	FEATURE SCORE MATRIX:			
FEATURES:	**Linkedin**	**Indeed**	**Glassdoor**	**Our Company**
Ease Of Application (for candidate)	9	6	5	10
Diversity of filter (for candidate)	10	7	7	9
Strength	9	6	7	9
Weakness	2	5	4	2
Oppurtunities	1	4	4	10
Threats	5	9	9	8
Brand reputation	10	5	6	8
Ethnicity/Gender Filter	0	0	0	10
Ease Of Screening (for recruiter)	8	5	5	9
Price Of Product	10	9	9	0
Technology Used	8	7	7.5	9.5
RELIABILITY	9	4	5.5	8.5
USER EXPERIENCE	9	3	6	8.5
Recommendations	9.5	2	4.5	9

6.1) Thus all the profile evaluation and analytical data would be taken from the user at the time to sign up which would be stored in the database in respective data schema.(can be altered at later stage)

6.2) Talent Acquisition Departments Would be able to access a separate dashboard which refers to all the jobs they posted and the candidates that are well suited for the same.
6.3) The priority of the Talent Acquisition portal would be to sort out the best fit on the basis of seval search criterias such as distance , job-description, gender ,ethnicity of the candidate .(weighted ranking)
6.4) Applicants Would be able to access a separate dashboard which refers to all the jobs they applied for and the companies that are well suited for them.
6.3) The priority of the candidate portal would be to sort out the best job fit on the basis of seval search criterias such as distance , job-description, CTC ,popularity of the company. (weighted ranking)

7. High-level system architecture and technologies used



Our deployment platform focuses on harnessing Gitlab Actions in order to execute shell scripts and build docker images. On every push to master, we have scripts that log into EC2 instances, pull the latest docker images and use

docker-compose to orchestrate our production environment. Each of the services that we run, are built on docker images and also have deploy manifests in docker-compose to define their runtime environment. By abstracting everything to docker, and using deploy scripts, this should allow our developers to more quickly get our environment up and running locally, as well as deploy new code to our server quickly. Our server is an 8gb ram/2 cpus AWS EC2 instance.

For our backend we will be implementing RESTful API's in order to persist/query data for our frontend. As a persistent datastore we will be using MySQL instance hooked up to our backend webservers. For a session store, we are targeting the use of a Redis instance in order to validate user sessions and maintain a logged in state for the user. Our backend webservers will be programmed using the Spring Boot framework to quickly implement an API for our frontend to use. We will hook it up to the MySQL instance using spring-data and the respective mysql driver.

For our frontend we will be leveraging the React framework. We will be using Redux for state management. We will be targeting to support Google Chrome and Firefox browsers.

For our load balancing and routing we will be using an Nginx application to route all of our services. Depending on if it is required we plan on running two instances of Nginx. One for the internal routing of our microservices, then a second Nginx app to route external traffic to the proper frontend services.

8. Team and roles

a. Github Master - Austin Wimberley
b. Front-End Lead - Battulga Tsogtgerel
c. Backend Lead - Austin Wimberley
d. Front-End - Ezra Player
e. Backend - Alfonso Duarte-Sarabia
f. Fullstack - Utkrisht Sharma

9. Checklist

1. Define main use cases and personas - Ezra. (Done)
2. Go over functional and nonfunctional requirements - Austin, Ezra, Battulga (Done)
3. Go over data requirements and planned items - Austin, Alfonso (Done)
4. Flesh out the data items before documents is finalized - Alfonso (Done)
5. Complete competitive analysis for the project - Utkrist (Done)
6. Complete execute summary of the project - Austin (Done)

7. Elaborate functional/non-functional requirements in the Milestone 1. - Battulga (Done)
8. Format the Milestone 1 documentation after everybody completes their task - Alfonso. (Done)
9. Make requested changes from feedback on Milestone 1. - Austin, Utkrist (Done)
10. Complete high level architecture (done)