CS 407

Spring 2018

Sprint 3 Planning Document

April 02, 2018

MakeItHire

Team 23

Avi Rakesh
Ben Weis
Nik Suprunov
Rajat Srivastava
Zack Fernandez

1 Sprint Overview

Overview

With Sprint 1 mostly focused on user-accounts, log-in and sign-up, with the the full-stack connection working, and Sprint 2 focused on the the functionality of the web-app, Sprint 3 will mostly be focused on developing message and application class designs which will enable a basic interaction between the students and recruiters through applications and message/chatting feature.

In terms of the functional requirements, the overview goal of this sprint is to finish up the remaining requirements for the project. We will allow recruiters to add supplemental questions to the job posting and make them view-able to all students. We will also allow students to respond to those questions. We will also allow the students to apply for multiple job postings with one click. We also plan to incorporate a messgaing/chatting feature between the recruiters and students. By the end of this sprint, we plan to have developed a working prototype of what our application is supposed to deliver.

In addition to the user-based focus of this sprint, we will also develop methods, in this sprint so that we as developers are able to step-in, override and remove companies which have signed up to be recruiters in our product.

Scrum Details:

Scrum Master: Rajat Srivastava

Scrum Meetings: MWF 4:30pm - 6pm

Risks and Challenges

Real-time chat

We want our application chat to be as responsive and fast as all modern messengers are. For that to be accomplished, we must decide on implementation design. One option is to use Socket.io with active listener that fetches data from server. Another is to put all messages in corresponding DB table and use Timer in Front End.

Apply

We want to split the CPU load between server and front-end, so to do that we need to decide where sorting and other heavy operations should be done. When it comes to pure DB fetch, it's usually done on back-end, however if we are updating the views and for that the data needs to be reformatted/rearranged, front-end should handle that (e.g. Jobs apply list).

Data concerns

All routes should be injection-free, meaning user won't be able to pass SQL statements to be executed on back-end. There will be a lot of different types of data being transferred from the server to the application which will furthermore be transferred to the companies' job applications.

These transfers of data needs to be protected that only the correct/appropriate users can access that data. Inappropriate data access could result in a data breach situation.

Teams

Frontend: Ben Weis, Nik Suprunov, Zack Fernandez

Backend & Database: Avi Rakesh, Rajat Srivastava

Workload Summary Table

Name	Total Work Hours
Avi	40
Ben	40
Nik	40
Rajat	40
Zack	40
Presentation Preparation	2
Total	**202**

2 Current Sprint Details

User Story 1: As a student, I should be able to answer supplemental questions on the jobs I have applied for if they requirement to do so.

#	Task Description	Time (hours)	Team	Owner
1	Create additional view for recruiters/head recruiters to add supplemental questions on job applications	8	Front-End	Zack
2	Allow for students to fill out supplemental questions on submission of job application	7	Front-End	Zack
3	Write database queries to post answers to the "Supplemental Questions"	6	Backend	Avi
4	Write HTTP API routes to handle the submission of supplemental questions for a student's application	6	Backend	Rajat

Testing/Unit Tests

1. Login as a student, and apply for a job with supplemental questions.
2. Login as a student, and apply for a job without supplemental questions.

Acceptance Criteria

- Job with supplemental question should accept my answers
- The answers should be view-able by the recruiters in their recruiter dashboard for that applicant
- The supplemental answers should be stored in the database, and view-able by developers.

User Story 2: As a student, I should be able to apply to multiple jobs at once, not limited to one company.

#	Task Description	Time (hours)	Team	Owner
1	Add support for applying to multiple jobs in a single view	7	FrontEnd	Nik
2	Ensure that the data is live updated to only display jobs that have not yet been applied to.	8	FrontEnd	Nik
3	Real time fetch request from the server with an option to update view.	5	FrontEnd	Ben
4	Write database queries to apply to multiple jobs, in one call	6	Backend	Avi
5	Write HTTP API routes to quickly handle the submission of job applications	6	Backend	Rajat

Testing/Unit Tests

1. Update page multiple times, check if displayed data is on par with Database
2. Double click on UI elements to see if "double-request" is handled properly

Acceptance Criteria

- Page autoreloads with no layout, visible delay issues (async)
- Only new/ not applied to jobs are displayed

User Story 3: As a student, I should be able to chat with recruiters who have contacted me.

#	Task Description	Time (hours)	Team	Owner
1	Create "chat bubbles"	5	Front-end	Nik
2	Populate "chat bubbles" with proper messages from Server	10	Front-end	Nik
3	Create chat view for students	6	Front-end	Zack
4	Populate chat notifications on profile page	6	Front-end	Zack
5	Create the ability to veiw a running list of all past recruiters who have contacted me	4	Front-end	Ben
6	Write database queries to save and access chat information	6	Backend	Avi
7	Write HTTP API routes to exchange chat information between students and recruiters	6	Backend	Rajat

Testing/Unit Tests

1. Send the message, check if it gets displayed instantly.
2. Send empty message

Acceptance Criteria

- Chat updates in real-time without user noticing it
- Only correct messages (from sender, user) are being displayed

User Story 4: As a Recruiter, I should be able to directly message students I am interested in

#	Task Description	Time (hours)	Team	Owner
1	Fetch data from server and update view accordingly	10	Front-end	Nik
2	Allow for recruiters to message applicants	5	Front-end	Zack
3	Create recruiter chat ui	8	Front-end	Zack
4	Recruiters should be able to search for users to chat with	8	Front-end	Ben
5	Write database queries to save and retrieve recruiter's message list	6	Backend	Avi
6	Create API routes to send messages to students as recruiters	6	Backend	Rajat

Testing/Unit Tests

1. Click on desired student to open the chat
2. Send the message to the student

Acceptance Criteria

- When user is selected, a related user window and chat history loads up
- All messages are stored properly in a chronological order

User Story 5: As a Recruiter, I should be able to add supplemental questions to my job postings

#	Task Description	Time (hours)	Team	Owner
1	Write database queries to post "Supplemental Questions" to a job posting	5	Backend	Avi
2	Create API routes for recruiters to add supplementary questions to a generic job posting	5	Backend	Rajat

Testing/Unit Tests

1. Posting desired Supplemental Questions into the text field.
2. Retrieving the same questions from the database

Acceptance Criteria

- Making the POST call to populate the Supplemental Questions should update the appropriate questions into the Database
- Retrieving the supplemental questions should be in the same order as originally posted

User Story 6: As a user of the application, I should only be authenticated to my relevant routes, and should not be able to use routes that were not designed for me.

#	Task Description	Time (hours)	Team	Owner
1	Security, authentication, prevent injection, and follow CRUD	19	Backend	Ben

Testing/Unit Tests

1. Try to execute admin only routes in postman, as a student
2. Try to execute head recruiter only routes as a recruiter

Acceptance Criteria

- The admin routes should not work with any other authentication level
- A recruiter should not be able to execute head recruiter routes

User Story 7: As an Admin, I should be able to delete a company profile

#	Task Description	Time (hours)	Team	Owner
1	Add button to company view that allows for the deletion of the company	6	Front-end	Ben
2	Handle proper redirects once page has been deleted	4	Front-end	Ben
3	Write database queries to allow an Admin to delete a company profile	6	Backend	Avi
4	Create API routes for the admin to delete a company profile, and all company related information in the database	6	Backend	Rajat

Testing/Unit Tests

1. Admins should be able to remove a company profile.
2. Admin should be redirected appropriately once deleting a profile.

Acceptance Criteria

- There should be no dangling resources once a company profile is deleted.
- Admin is properly redirected

User Story 8: As an Admin, I should be able to modify a company profile

#	Task Description	Time (hours)	Team	Owner
4	Write database queries to allow an Admin to modify a company profile	5	Backend	Avi
5	Create API routes to allow admins to visit a company profile and edit/remove information, as seen fit	5	Backend	Rajat

Testing/Unit Tests

1. Login as an admin, and go to a company profile to edit it's information.
2. Login as a student, and check if the information is changed in the company's profile.

Acceptance Criteria

- A success message should be displayed when admin changes a company information.
- The changes should be reflected as a student

3 Backlog

3.1 Functional Requirement

As a Student:

1. ~~I should be able to make a personal account on MakeItHire.~~
2. ~~I should be able to upload my resume to my account.~~
3. ~~I should be able to upload a cover letter to my account.~~
4. ~~I should be able to link my job-related accounts (e.g. Git).~~
5. ~~I should be able to answer supplemental questions on the jobs I have applied for if they require me to do so.~~
6. ~~I should be able to browse all open job offers.~~
7. ~~I should be able to browse all companies with available jobs.~~
8. ~~I should be able to apply to multiple jobs at once, not limited to one company.~~
9. ~~I should be able to chat with recruiters who have contacted me.~~

As a Head Recruiter:

10. ~~I should be able to create an account for my company~~
11. ~~I should be able to add a description of what my company does~~
12. ~~I should be able to add job positions available for the students to apply for~~
13. ~~I should be able to add recruiters to my company profile~~
14. ~~I should be able to request recruiter to be removed from my company profile~~
15. ~~I should be able to add a description for the jobs posted~~

As a Recruiter:

16. ~~I should be able to directly text students I am interested in~~
17. ~~I should be able to sign up as a recruiter using my company email address~~
18. ~~I should be able to post a job opening/open an application~~
19. ~~I should be able to add supplemental questions to my job postings~~
20. ~~I should be able to see all the applicants who have applied to one of my job postings~~
21. ~~I should be able to contact applicants through the app~~
22. ~~I should be able to filter applicants in the app~~

As an Admin:

23. ~~I should be able to approve a new company profile request~~
24. ~~I should be able to delete a company profile~~
25. ~~I should be able to modify a company profile~~
26. ~~I should be able to add recruiters to a company profile~~
27. ~~I should be able to remove recruiters from a company profile~~

As a Developer:

28. ~~I should be able to shut down the registration process~~
29. ~~I should be able to add/remove account for a student with a .edu account~~
30. ~~I should be able to add/remove accounts from our database when companies join/drop our services~~
31. ~~I should be able to add/remove account for a recruiter registered for the company~~

3.2 Non-Functional Requirements

Stress Testing

32. This includes creating numerous dummy users in the database, and making them perform activities. It includes the following:
    (i) Response Time: With a lot of users, that app should not become slow to respond.
    (ii) Scalability: A lot of users should be able to use the application at the same time.
33. Along with this our server should be able to handle a reasonably large amount of users and events
34. Servers must be up 24/7

Usability

35. It should be an easy to use application, an application should be intuitive and work smoothly.

Security

37. The application should keep the user's account secure using good authentication method attached to a server. It should also secure user's contact information, and share only what he/she prefers to share.

Performance

38. The app should not be slow to work with, and should not affect the other apps working on the device as well.
39. Response time must be minimal with updates occurring in real time when the application is open.

Size of the application

40. The application should not take too much space in the user's device. The app should limit its cache size to 32MB max.