CS 407

Spring 2018

Design Document

January 26, 2018

MakeItHire

Team 23

Avi Rakesh
Ben Weis
Nik Suprunov
Rajat Srivastava
Zack Fernandez

1 Purpose

Job Search and Application process is arduous for university students and equally demanding for the company recruiters as well. That is why we are looking forward to design a web application that will streamline that process for both parties and make it much more effective and efficient. Our application will provide the students a platform to upload their resume, cover letter, and fill in their personal and professional information, only once, which will be then used as a formal job application which is acceptable by all the companies and their recruiters.

The purpose of our application is to present a better alternative to the extensive job application and simplify it. We plan to offer a solution which will eradicate the need for companies to host undergraduate student applications in their website. This will also save the trouble of the student because this will eliminate the inconsistency students face when being made to fill-in all of their personal and professional information for each application; which is not only tedious, but also time consuming. Our goal is to change the way company recruits university students, and enhance the way college students apply for jobs.

2 Design Outline

Our project is a system, available to all the users online, which provides the ability for students to search and apply for jobs seamlessly, as well as makes hiring easier for the recruiters. Our implementation of the application will be based on a Client-Server Model. The client-side application will interact with the server, which will have the access to send, retrieve and update the information in the database, and serve back the client accordingly, while maintaining a level of authentication to avoid database injection.

1. **Client** - Web Application and iOS Application Extension
    - Users will be able to interact with the application easily by accessing the website online in a Web browser, or by using the iOS Application Extension in their iPhone at own their convenience.
    - The client application will communicate with the server by using HTTP requests and passing data either in the form of JSON objects or query strings.
    - The secure data received from the server will be neatly presented in a user-readable format to their user in their respective client applications.
2. **Server** - NodeJS
    - The server will be hosted using NodeJS files, which will communicate and query with the database using the necessary security protocol.
    - The server will be responsible on sending back a secure response string to the user.
    - The server will also return error codes along with the response string to make it easier for the client to understand what the response means, and if there is an error condition.
3. **Database** - MySQL

- The database will be responsible for storing the data of all the users' information, the student profiles, company profiles, job postings, current applications, and the conversations.
- The database would require a user name and password to log in, and will not be accessed without it.

High-Level Overview

Both the web application client and the iOS App client offer a many-to-one client-server architecture so as to allow multiple users using the application and communication with the server at the same time without any issues. The usual flow of information would be that the the client would send or request some information, from the server, which in turn the server will query the database to fulfill. All information sent and received will be secured and all connections are will be authenticated.
Below is a high-level diagram of the full stack components of the application.

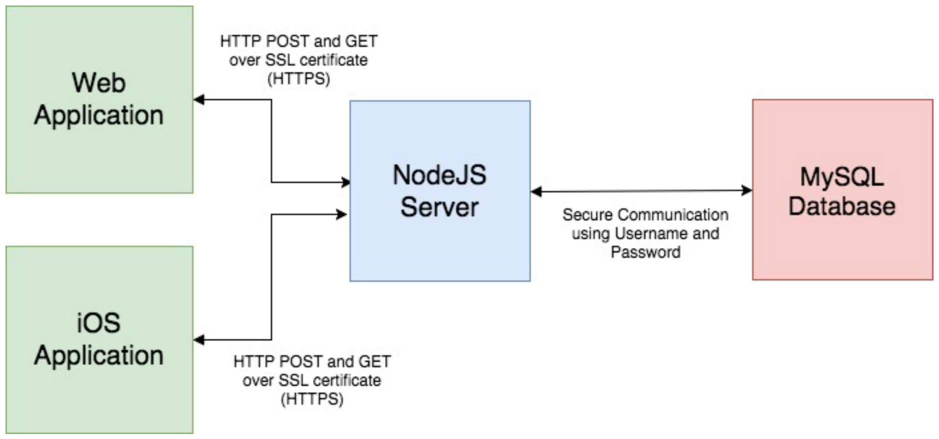

Web Application
HTTP POST and GET over SSL certificate (HTTPS)
NodeJS Server
Secure Communication using Username and Password
MySQL Database
iOS Application
HTTP POST and GET over SSL certificate (HTTPS)

Figure 1: Full Stack Diagram

The Front-End client will offer different types of services based different types of users and their privileges. Given below is the outline of the different types of users who will be using the application, and their privileges.

1. **Student**: Students will be able to edit their profiles, add resume, cover letter, and apply for as many available job positions as required. They will be able to browse all jobs and search for specific companies and job positions. They will be able to reply to a conversation started by a recruiter, and also look at the job positions they have applied to.
2. **Recruiter**: Recruiters will be able to join their company profile as a recruiter and will be able to post, edit or remove job positions for their respective companies. They will be able to look at all the applicants who have applied to the job positions for their company and start a conversation with them using the in-house messenger.
3. **Head Recruiter**: Head Recruiters will be responsible for creating a company's profile on the app for the first time. Beyond the abilities of a recruiter, they will be able to add a

company's logo and description. They will also be able to make some other recruiter under their company as the head recruiter in case necessity arises.

4. **Admin**: Admin is responsible for handling the company's technical issues. They will have the override access to remove any user, be it a student, recruiter, or a head recruiter, and delete a company's profile if needed. They will be able to add them into the system. The admin will also be able to monitor the statistics of the application like the number of students registered, or the number of applications submitted.

3 Design Issues

3.1 Functional Issues

1. How many types of account do we want?

   Option 1: Two - Students and Recruiters
   Option 2: Three - Students, Recruiters, Admins
   Option 3: Four - Students, Recruiters, Head Recruiter(Company), Admins

   Reason: We want a 'Head Recruiter' for a company who is responsible for the information about that company, requesting addition and deletion of a recruiter's account on the portal for his/her company. Admin(s) will be able to add or remove students, company and it's affiliates.

2. Do we want to allow companies/recruiters to add additional questions?

   Option 1: No, the companies should only be able to see students from what information that students have already filled up on the application about their experience.
   Option 2: Yes, the companies should be able to ask specific questions which will further enhance the candidates from one another. 2-5 questions at most.

   Reason: The companies want to be able to further assess a student than just their basic information about their experience. Allowing the option of asking more questions gives the companies that added piece of information that would help them to filter out the good candidates from not-so-good.

3. Should we allow the recruiters to be classified into subcategories?

   Option 1: Yes
   Option 2: No

   Reason: We want the recruiters to be allowed to be classified into different, pre-set, categories so that only recruiters of that category can modify the job posting of that category within that company.

4. How many tables do we want?

   Option 1: Three - Users, Company, Jobs
   Option 2: Four - Students, Recruiters, Company, Jobs
   Option 3: Five - User, Students, Company, Jobs, Application

   Reason: We want to be able to classify the users into Student and Company (Recruiters) with students and company having it's separate information while the common information will be stored in 'User' table. Jobs table will contain all the job posting available form different companies and the application table will contain the details of the students who have applied to that job posting. For more detail, please refer to the Figure 7 .

5. How do we set who is the 'Head Recruiter' for a company?

   Option 1: First to sign up for our system.
   Option 2: Assign a head recruiter once a company joins our service.

   Reason: Our idea is that the first person who signs up for our service will automatically become the 'Head Recruiter' for that company. If the company wants to change that, it can be requested and handled by the Admin.

6. How much of access do we want to give on the iOS application?

   Option 1: Full Access - Complete control of the account.
   Option 2: Read Only Access - Only content can be viewed, but not changed, such as status for on-going job application, deadlines, etc.
   Option 3: Message + Read Only Access - The user can only create and reply to a message on the application and the rest will still be a read only access.

   Reason: We wanted our application to be more of an application web-portal with partial mobile application access which allows to reply to recruiters' messages. This design decision was made by keeping in mind that students do most of the application work on a laptop or a desktop and only need to reply to messages instantly.

7. Do we want to use a resumé parser?

   Option 1: No
   Option 2: Yes

   Reason: We will not be using a word parser to parse information from the resume to the system. We want the students to fill up the basic information once and then those information can be used to automatically populate the job application.

3.2 Non-Functional Issues

1. What should be used as our backend hosting service for the Server and for the Database?

   Option 1: Heruko
   Option 2: AWS

   Reason: After much debate, we will be using a mixture of both the services. We will be using AWS RDS for the database structures and Heruko to host the server files (Node.JS).

2. Should we use HTTP or HTTPS?

   Option 1: HTTP
   Option 2: HTTPS

   Reason: HTTP is HyperText Transport Protocol which is the wat data gets passed back and forth between web servers and clients. HTTPS is the secure version of HTTP and the 'S' at the end of HTTPS stands for 'Secure'. With HTTPS, communications between your browser and the website is encrypted. If implemented in our app, private information such as log-in information and recordings will be much less likely to be stolen; for this reason, we chose to utilize HTTPS for our service communication.

3. What iOS Development language will we use for the app?

   Option 1: Swift
   Option 2: Objective-C

   Reason: Apple has now released Swift 4 which is now the standard for all iOS development and the application can be scaled to other Apple devices very easily, in the future.

4 Design Details

4.1 Database Design

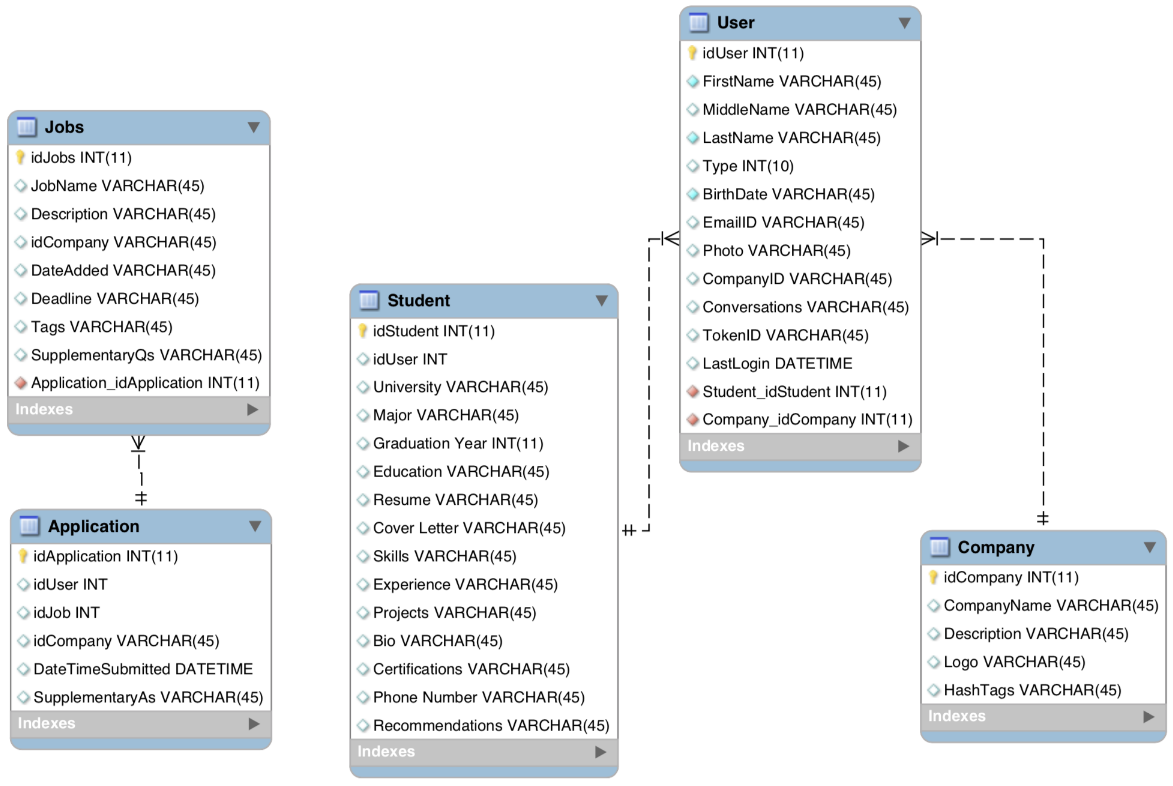


Figure 2: Database Schema Diagram

4.2 Description of Data Classes and Their Interactions

1. User

    - This is the highest level class for user accounts. Anytime someone registers with our service, an instance of User is made for them with the specified information.
    - We differentiate between Student, Recruiter, and Head Recruiter based on the Type field
    - Based on their account type there may also be an associated Student instance or Company instance as well. If a recruiter signs up and their company does not have an instance

in the Company table, then one will be made with a unique idCompany, and that User account will be given the type Head Recruiter.

- Recruiters and Head Recruiters will only have an instance in the User table as their information does not extend beyond this.

2. Company

   - An instance in the Company table represents a unique company that is registered with our service.
   - Given that each Company has their own company page, we will store this associated information in the table as well
   - Every recruiter will also store the idCompany as a field in their User profile as a way to reference which company each recruiter works for
   - Recruiters reference this with every job they post and students can search for these as well

3. Student

   - Any time a User account is created with the Type correlating to student, an associated instance will be made in the Student table.
   - Here will be stored all information for the student's profile (resume, bio, cover letter, etc.)

4. Jobs

   - Any time a recruiter posts a new job it will be stored in this table
   - Jobs will have tags that allow them to be easily filtered, as well as their associated companies
   - Any relevant information for a job posting (title, description, deadline, etc.) will be stored here
   - Job instances are only created by Recruiters/Head Recruiters

5. Application

   - This table is an extension of the Job table, and new instances are created every time a Student applies for a job
   - Each application has reference to the student who submitted it
   - Once an application is submitted, recruiters may then reach out to the associated student and begin messaging them through our application
   - The contents of their conversations will be stored in the User table

4.3 User Work-Flow Diagram

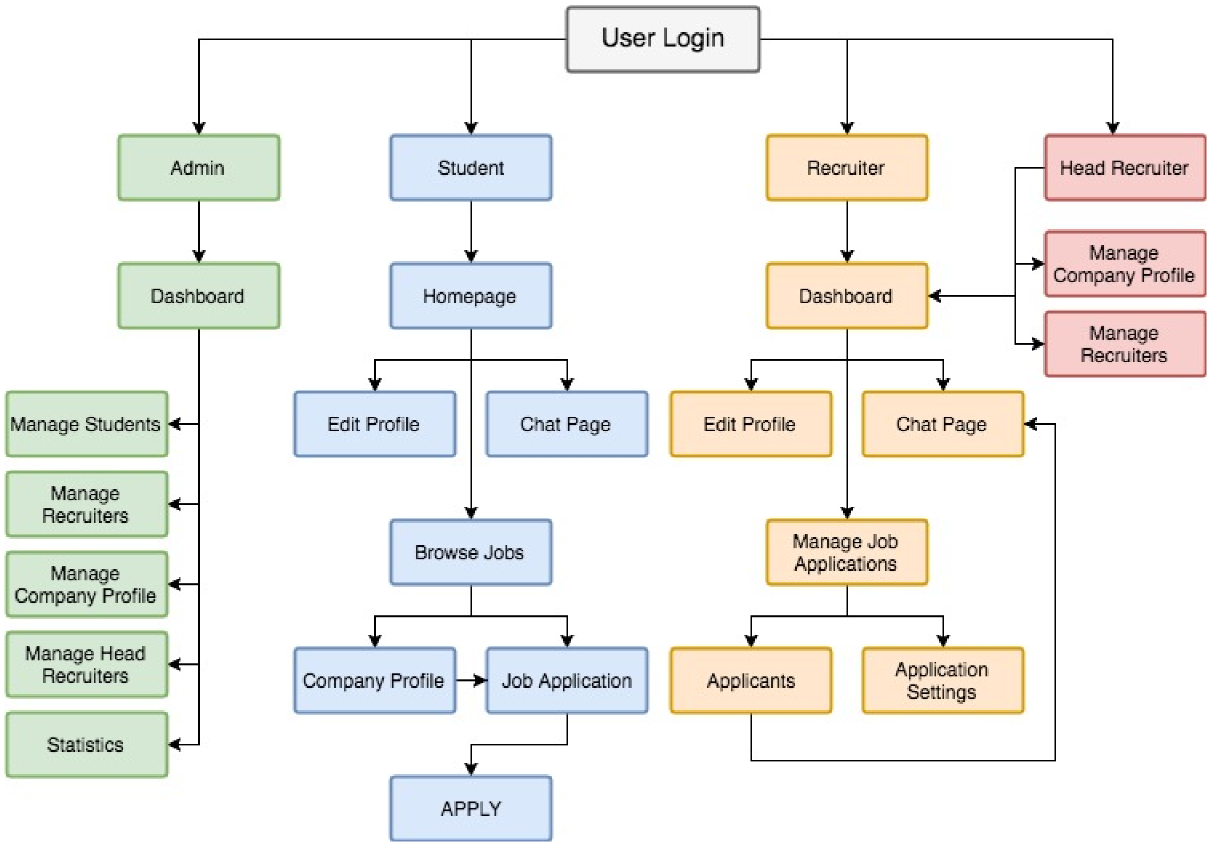
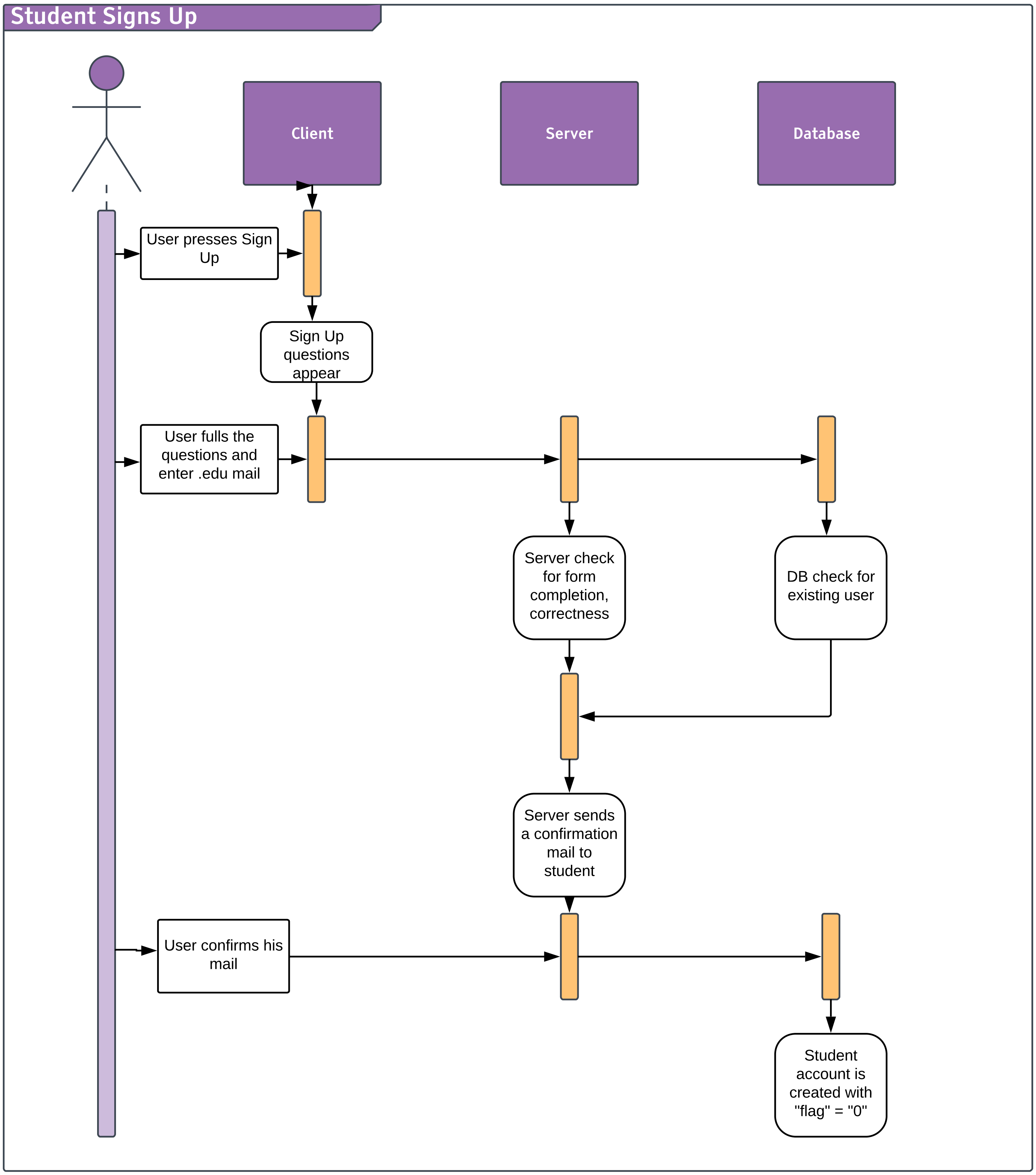
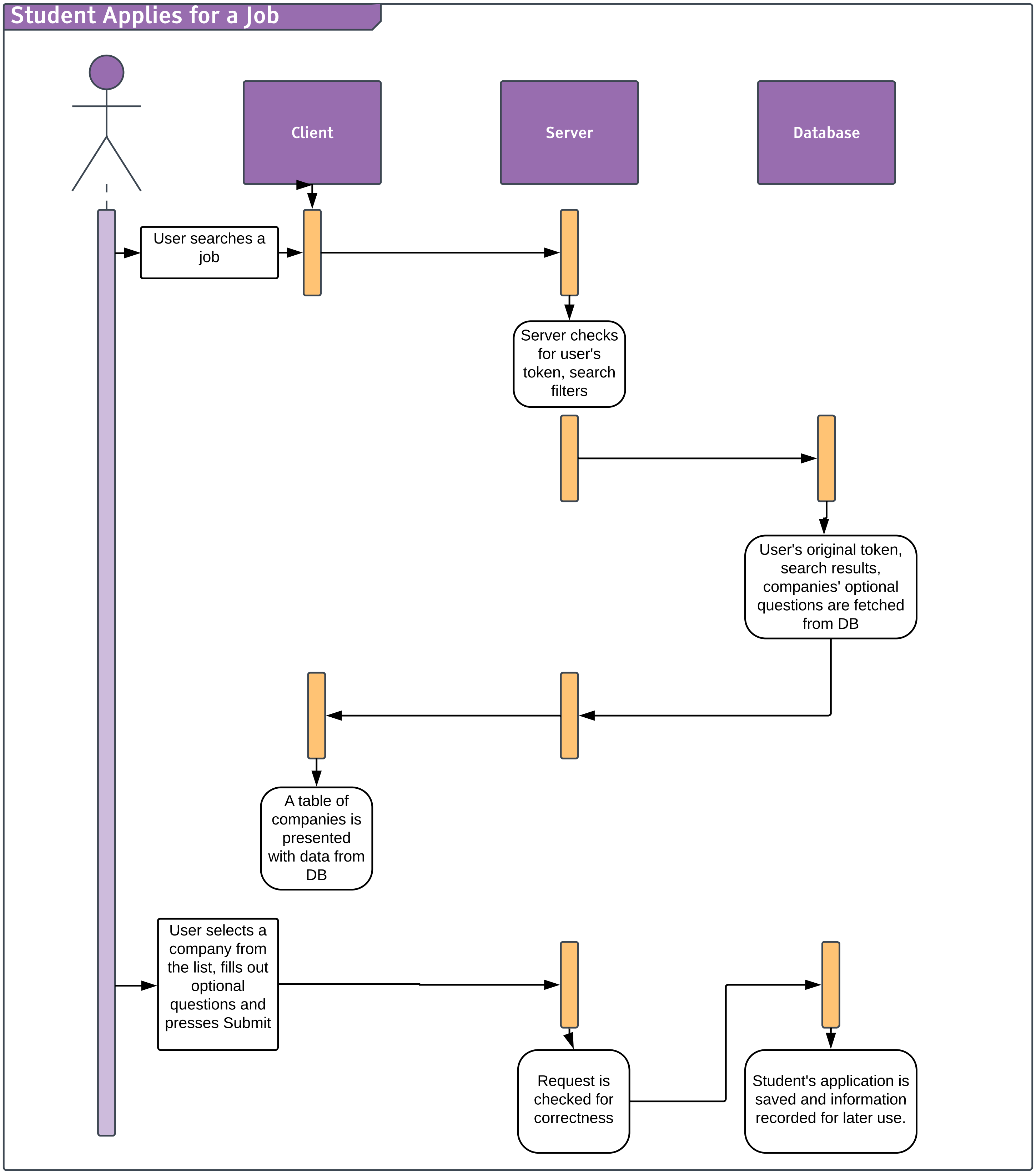
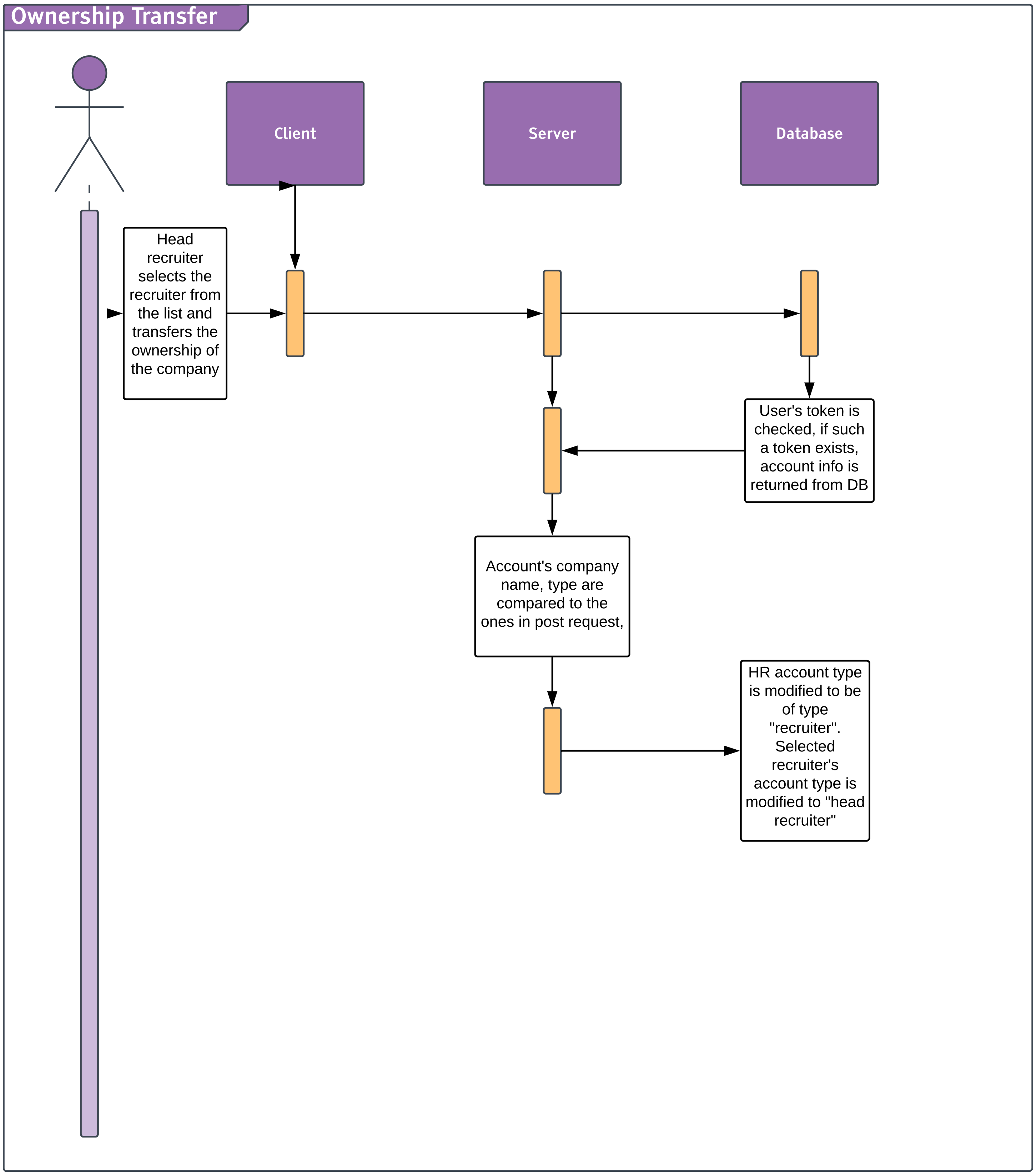


Figure 3: User Work Flow

4.4 Sequence Diagrams


Student Signs Up
Client
Server
Database
User presses Sign Up
Sign Up questions appear
User fulls the questions and enter .edu mail
Server check for form completion, correctness
DB check for existing user
Server sends a confirmation mail to student
User confirms his mail
Student account is created with "flag" = "0"


Student Applies for a Job
Client
Server
Database
User searches a job
Server checks for user's token, search filters
User's original token, search results, companies' optional questions are fetched from DB
A table of companies is presented with data from DB
User selects a company from the list, fills out optional questions and presses Submit
Request is checked for correctness
Student's application is saved and information recorded for later use.


Ownership Transfer
Client
Server
Database
Head recruiter selects the recruiter from the list and transfers the ownership of the company
User's token is checked, if such a token exists, account info is returned from DB
Account's company name, type are compared to the ones in post request,
HR account type is modified to be of type "recruiter". Selected recruiter's account type is modified to "head recruiter"

4.5 UI Mockups

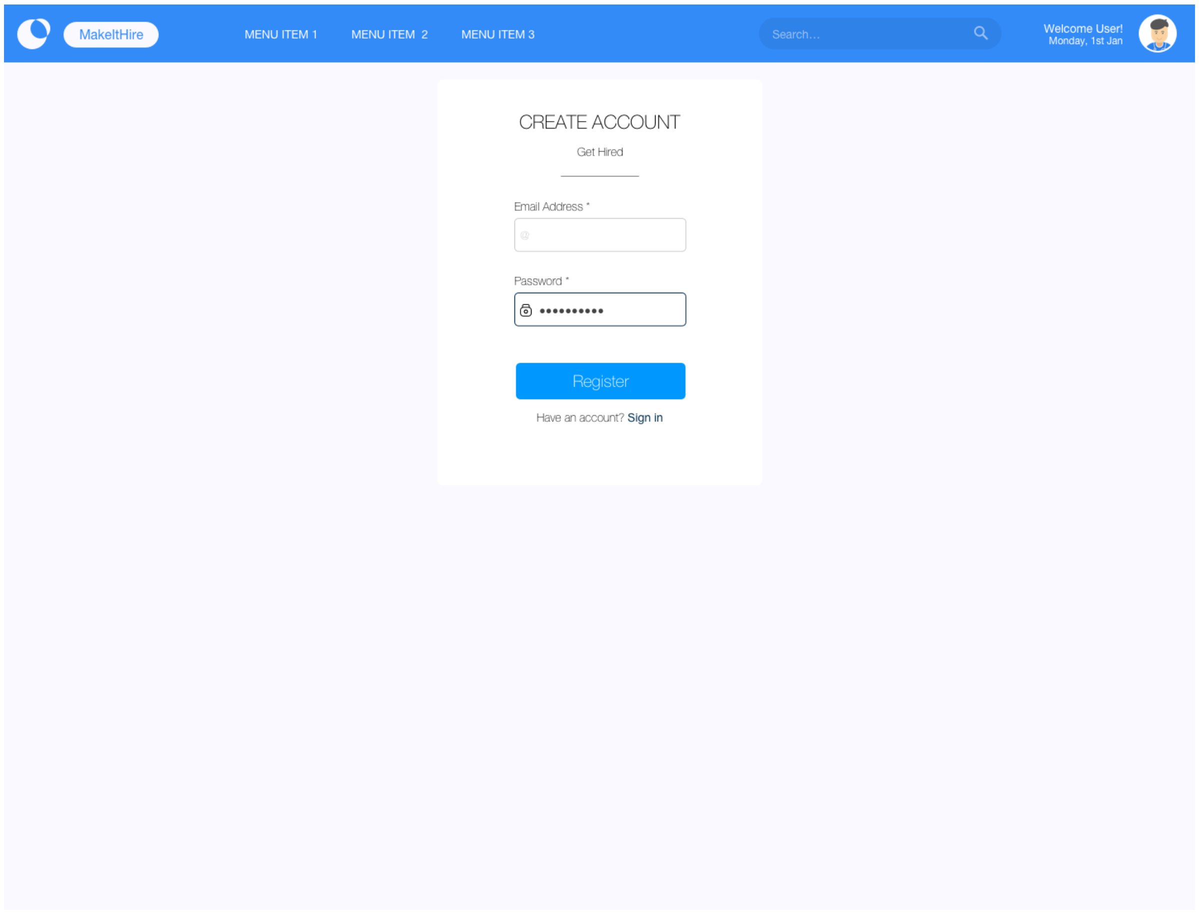


Figure 4: Register Page

- The register page will be for first time users who would like to use MakeItHire. Users will need to confirm their email address via a link sent to their email. Students will then be redirected to their profile page where they can set up their resume, job experience, and other details about themselves.

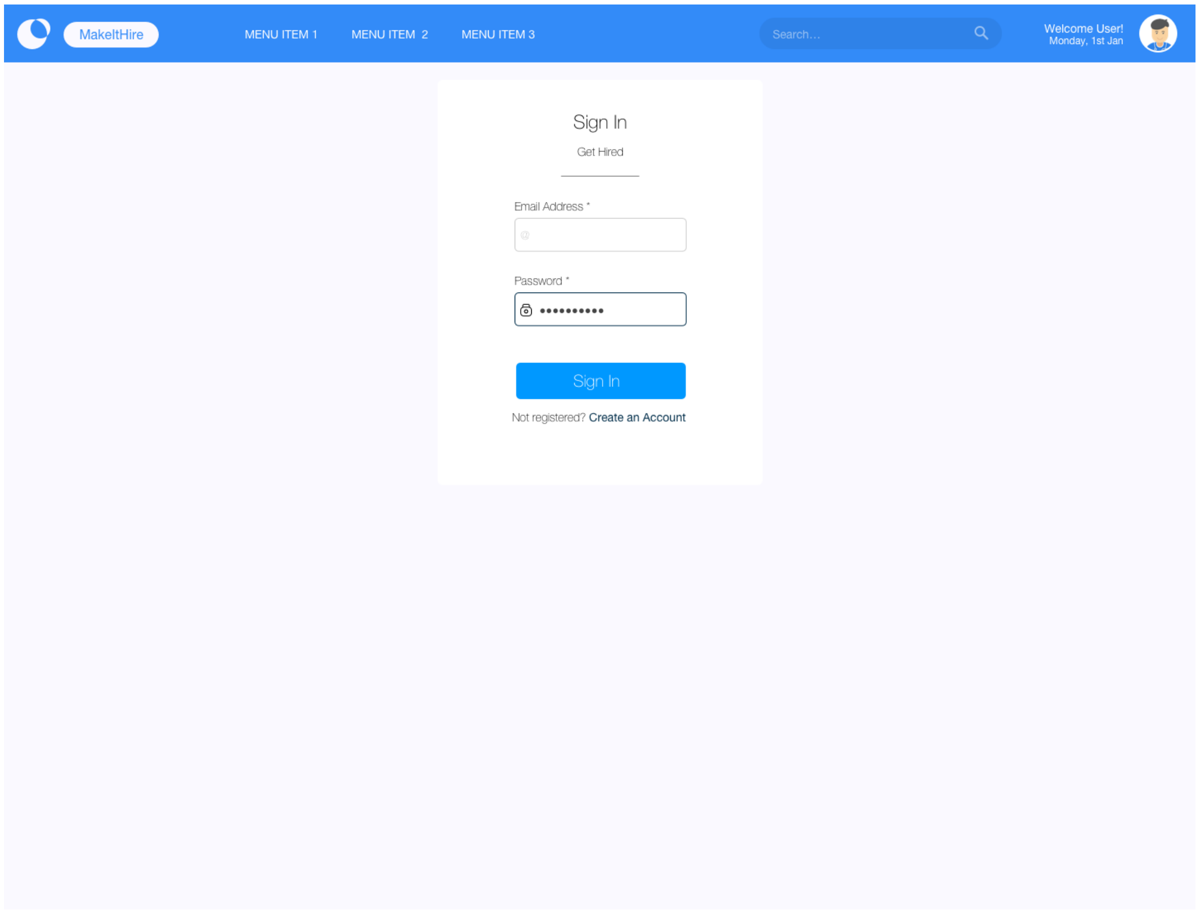


Figure 5: Sign In Page

- The sign in page is similar in form to the register page. This is the form that will verify a users credential and will direct an existing user to the job search page.

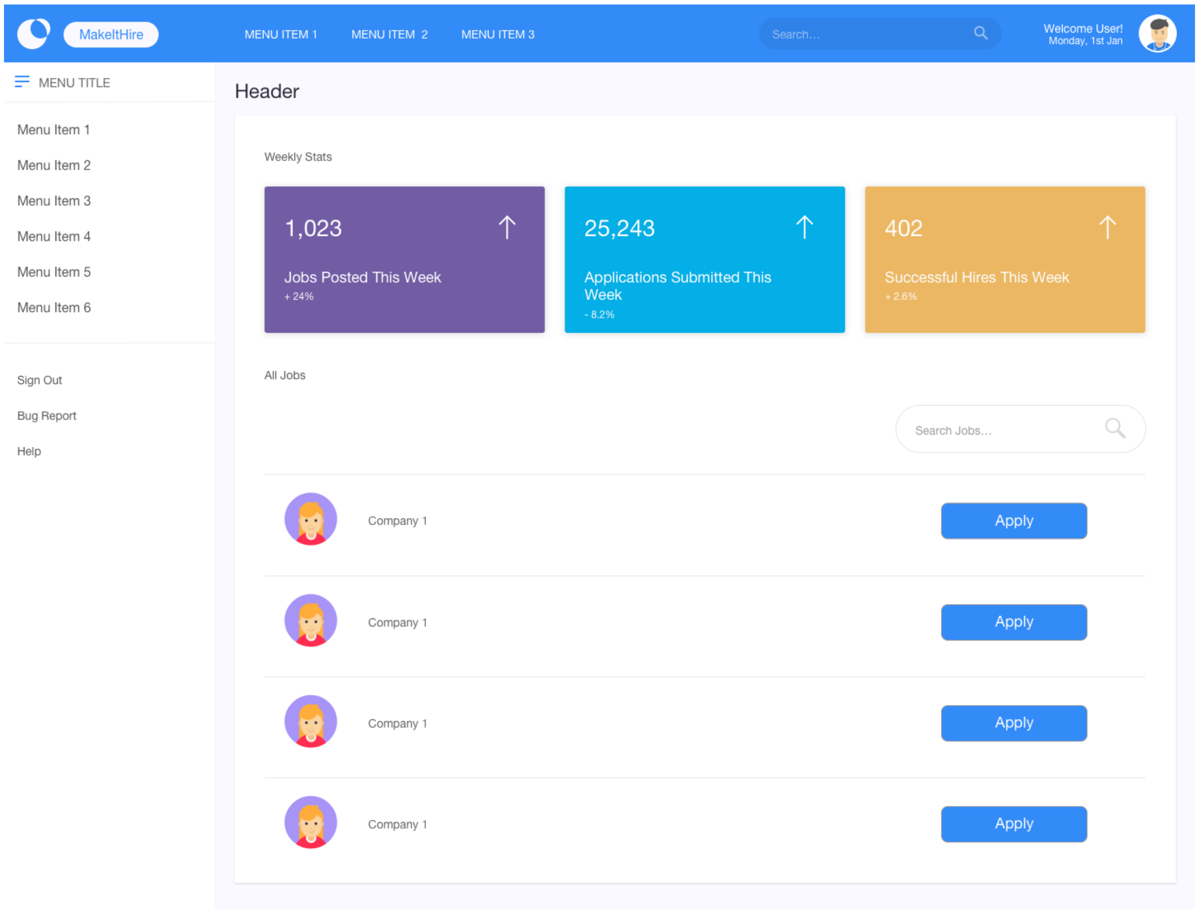
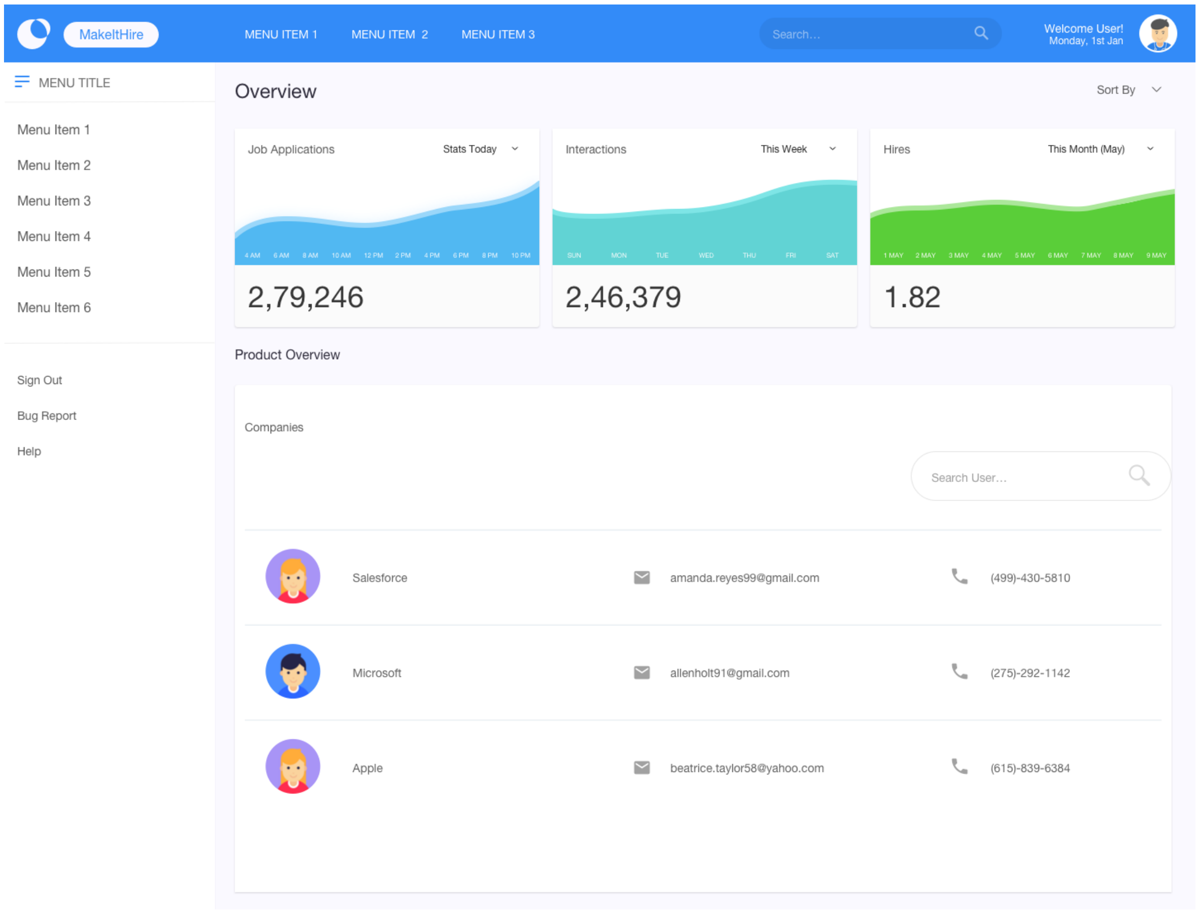


Figure 6: Job View Page

- This is where a user will be able to browse and apply to current job listings. Jobs will be search-able and the user will be presented with a row of statistics upon first visiting the page. Each job listing will expand to show more information and can direct the user to a company profile page.



Job Applications	Interactions	Hires
2,79,246	2,46,379	1.82

Salesforce	amanda.reyes99@gmail.com	(499)-430-5810
Microsoft	allenholt91@gmail.com	(275)-292-1142
Apple	beatrice.taylor58@yahoo.com	(615)-839-6384

Figure 7: Admin Page

- The administration page allows admins to view companies, and their main recruiter contacts. Upon viewing a company and admin will have the ability to change the head recruiter, remove accounts, and manage company pages.