

SW Engineering CSC

648/848 Fall 2022

The Hiring Guru

Team 2: Binary Brains

Farhan Haider (Team Lead/Backend Lead): shaider1@sfsu.edu

Kenny Leong (Frontend Lead/GitHub Master): kleong2@mail.sfsu.edu

Mamadou Bah (Frontend): mbah1@mail.sfsu.edu

Khushi Khanna (Backend) kkhanna@sfsu.edu

Eric Leow (Frontend): elow@mail.sfsu.edu

Milestone #4

Dec 1st, 2022

Row	Document Version	Review Date	Description
1	Version 1	11/14/2022	Milestone 4 was assigned, Document was first created with sections to be filled out
2	Version 2	11/16/2022	Sections were slowly being filled out with various work to be done on the application
3	Version 3	11/21/2022	Continue to work on the application while the document is being completed.
4	Version 4	11/28/2022	Final few meetings to discuss last things needed for the application as well what is left for the document
5	Version 5	11/30-12/1	Wrapping up the document for submission and anything left for backend work

Table of Contents

1 - Product Summary.....	3
2 - Usability Test Plan.....	6
3 - Quality Assurance Test Plan.....	14
4 - Code Review	21
5 - Self-Check for Best Practices for Security	44
6 - Adherence to original Non-Functional Requirement specs.....	48
7 - Team Contribution	52

Product Summary

Product name: Hiring Guru

All Major Committed Functional Requirements

Authorized User

All Committed Functions
A User is able to logout of the website anytime. Additionally, the User is able to log out from anywhere on the website.
A User is able to create a company. To create a company, the User needs to provide company name, company website, company email, industry type of company, a phone number of User, an email of the User, company size, and company type.
A User is able to create a new role. To create a role, the User needs to provide the role title, role expectations, and role benefits.
A User is able to update an existing role. To make this work, a User has to pick an existing role and update its properties.
A User should be able to see the list of all roles. For each role, they are able to see the role title, expectations of the role, and the benefits of the role benefits.
A User should be able to remove a role from the system. To do so, they have to go to the role page, and click on the section called actions, and from there the user is able to select delete which will allow the user to delete the role.
A User is able to create a new job. To create a job, the User needs to provide the job title, job workplace/location, job employment type, job location, and job description.
A User is able to close a job, if they don't wish to leave the job listing on the website.
A User should be able to see the list of all job. For each role, they are able to see the job title, job workplace/location, job employment type, job location, and job description, and change any or all of the requested job fields.
A User should be able to remove a job from the system. To do so, they have to go to the job page, and click on the section called actions, and from there the user is able to select remove which will allow the user to delete the role.
A User should be able to see the list of all jobs. For each job, they are able to see job title, job workplace/location, job employment type, job location, and job description,
A User should be able to add a referral for a specific position. For a user to be able to do this, the user has to include their first name, last name, user email, and a description of their chosen

candidate, and the applicant's resume.
A User is able to update an existing application. To make this work, a User has to pick an existing application and update its properties.
A User should be able to remove an application from the system. To do so, they have to go to the application page, and click on the section called actions, and from there the user is able to select remove which will allow the user to delete the application .
A User should be able to create the hiring process for a Role. For this, the User should be able to add different stages that are needed for the role hiring process.
A User should be able to add a new stage to the hiring process for a Role. They are able to add a new stage and provide a description of the new stage added.
A User should be able to see the hiring process for a role. The hiring process shows the different stages and a description of each of the stages.
A User should be able to move a stage backward in the hiring process. For a User to do this, they have to go to the recruitment process page and find the stage they wish to move backward and click on the arrow that allows them to move the stage backwards.
A User should be able to move a stage forward in the hiring process. For a User to do this, they have to go to the recruitment process page and find the stage they wish to move backward and click on the arrow that allows them to move the stage forward.
A User should be shown a confirmation before deleting a stage in the hiring process. This ensures that the action done by the User is done as expected, and a stage isn't deleted by accident.
A User should be able to move from page to page by navigating through the various links in the side. They are able to go from one page to another easily.
A User should be able to search for jobs using a query field
A User should be able to search for a job. A User should start off by selecting a job type(Full Time, Part Time, or All), and entering a keyword to match the job they are looking for.

General User

Requirement Statement
A General user should be able to sign up directly for an individual account. The General user would be prompted to select an email address and a password to be able to login.
A General User should be able to sign up using a Google account.
A General User should be able to login using a Google account. The General user should select

an existing google account and the matching password.
A General User should be able to login using Username and Password. As long as the General User enters an existing and valid username and password the user should be able to login and view the website.
A General user should be able to recover his/her account using email. If an account that matches the provided email exists, an email will be sent to the user that would allow them to change their password.
A General User should be able to go back to the home screen anytime by clicking on the HiringGuru logo.
A General User should be able to see the HiringGuru about page in the page footer.
A General User should be able to see all of the product features in the home page, and furthermore should be able to navigate through the entire website.
Any General User should be able to search for a job. A General User should start off by selecting a job type(Full Time, Part Time, or All), and entering a keyword to match the job they are looking for.
Any General User should be able to apply for a job. To do this, the General User needs to provide their name, phone number, email, and resume.

What is Unique about our Product

Can you think of a software application that allows organizations to not only create companies, add employees, and manage their teams, but also have the ability to post jobs, view various candidates that applied, manipulate their hiring process with ease, and have an easy way to contact them? No? Well not anymore. With the Hiring Guru application, you get the combined features of managing your entire company or a designated team, while also being able to view and oversee the candidates that apply to the jobs you post throughout their application process. With just a few buttons, anyone in your company can create any job that is desired and have the wonderful option of posting it to multiple job searching platforms such as LinkedIn. There, any candidate will have the option to apply and enter the hiring process, where the employee can easily view all candidates who apply and what stage they are at. It's all on one application and all made for ease. No longer will you have to use different applications for hiring people, such as posting your job on LinkedIn, and then going into a shared Excel Sheet and manually typing in the candidates you receive and updating it based on their status through the process. The Hiring Guru will do all that for you and more all on the application: Want to search for a specific job? We got that! Want to view the hiring process for every job posted? Already done! Want to

remove, edit, and submit a job referral on the same page?? Hiring Guru has got you covered! And keep in mind, all of this is for some sweet deals and payment plans to help even the smallest of startup companies hire the best employees for their companies. That sounds like an awesome deal to me! No one is doing it like Hiring Guru is and you should definitely check out our site below!

URL:

<https://hiring-guru.us-west-1.elasticbeanstalk.com/>

Usability Test Plan

Tests

1 - Add a new stage to the recruitment process

Test Objectives

In a company, the hiring process for a role can involve multiple stages including interviews and technical assessments. One of the main features of the HiringGuru app is that it allows users to design the hiring process for a particular role in the company. The purpose of this test is to understand how easy it is to add a new stage to the hiring process for a role, one of the most basic steps in the process of designing the hiring process. The usability of this feature is critical to the success of the feature because all other features related to designing the hiring process require the creation of the hiring process stages.

More specifically, the test has the following objectives

1. Check whether it's easier for the user to navigate the Recruitment Process Page
2. Check whether the user can easily locate the button for adding a new stage
3. Check whether the form for adding a new stage is easy to fill
4. Check whether the form validation errors are being communicated properly
5. Check how efficient the current user experience is
6. Measure the overall user satisfaction with the feature

Test Description

System setup

Since the application is accessible through the browser, the only setup required is a GUI system with a supported browser i.e. active version of Firefox or Chrome.

Starting point

This usability test starts when the user is on the Hiring Guru dashboard and wants to add a new stage to the hiring process for a particular role. Here're the steps needed to reach the starting point for this test

1. Go to the Hiring Guru URL
2. On the Landing Page, click on the Login button
3. Enter test credentials

Intended Users

The intended users for this feature are the HR executives in the company because it's their job to administer the recruitment process.

URL of the system

<https://hiring-guru.us-west-1.elasticbeanstalk.com/>

2 - Change position of a recruitment stage(Mamadou)

Test Objectives

There are multiple stages in a hiring process that can vary in a number of ways depending on the job. A more technical-focused job may have numerous coding assessments compared to a non-technical job that will be having a lot more behavioral interviews. With these various recruitment stages, Hiring Guru gives any company the ability to shift the different positions of the recruitment stage to accommodate for the skills they want to see from their candidate. An example would be shifting a technical interview before a behavioral interview for certain candidates, so their coding skills can be assessed even before hiring consideration. Features such as this are important for companies to make every recruitment process unique to every job posted.

More specifically, the test has the following objectives

- Check whether it's easy to find the buttons to move a recruitment stage
- Check whether the functions that move the recruitment stages are understandable
- Check how effectively the features respond to the user's actions
- Check how efficiently the current user experience is
- Measure the overall user satisfaction with the feature

Test Description

System setup

Since the application is accessible through the browser, the only setup required is a GUI system with a supported browser i.e. active version of Firefox or Chrome.

Starting point

This usability test starts when the user is on the Hiring Guru dashboard and wants to move a recruitment stage to a different position within the entire recruitment process.

1. Go to the Hiring Guru URL
2. On the Landing Page, click on the Login button
3. Enter test credentials
4. Click on *Hiring Pipeline* under *Recruitment* in Dashboard
5. Use the given features

Intended Users

The intended users for this feature are the HR executives in the company because it's their job to administer the recruitment process.

URL of the system

<https://hiring-guru.us-west-1.elasticbeanstalk.com/>

3 - Delete a recruitment stage(Mamadou)

Test Objectives

Now as time goes on, certain features within a company may need some modifications of sorts or just need a sense of general updating. This can apply to the recruitment process, where some stages within the process are no longer needed and should be removed entirely. The Hiring already has a feature like that implemented within the Hiring Pipeline. Not only can certain stages be created or moved, but they can also be entirely deleted if they are no longer needed. A good example of this can be a company no longer finding a need for certain technical roles to have a certain amount of behavioral tests. These companies have the ability to remove any or all the behavioral tests stages from their specific recruitment process of choice to then focus on the other stages of the process. These features continue to add more flexibility for the company for their recruitment process.

More specifically, the test has the following objectives

1. Check whether it's easy to find the button to remove a recruitment stage
2. Check whether the functions that remove the recruitment stages are understandable
3. Check how effectively the features respond to the users actions
4. Check how efficient the current user experience is
5. Measure the overall user satisfaction with the feature

Test Description

System setup

Since the application is accessible through the browser, the only setup required is a GUI system with a supported browser i.e. active version of Firefox or Chrome.

Starting point

This usability test starts when the user is on the Hiring Guru dashboard and wants to move a recruitment stage to a different position within the entire recruitment process.

1. Go to the Hiring Guru URL

2. On the Landing Page, click on the Login button
3. Enter test credentials
4. Click on *Hiring Pipeline* under *Recruitment* in Dashboard
5. Use the given features

Intended Users

The intended users for this feature are the HR executives in the company because it's their job to administer the recruitment process.

URL of the system

<https://hiring-guru.us-west-1.elasticbeanstalk.com/>

4 - See applications for a job

Test Objectives

When a company posts a job that candidates can apply for, they should have the ability to view all job application submissions. The objective of this test is to prove just that for the Hiring Guru. Once a candidate submits a job application, it's very important for employers to be able to see all of the applications for a job. Seeing all of the applicants, allows the employers to make wise decisions about selecting a candidate. The Hiring Guru application easily allows users to view all job applications submitted for any job posted.

More specifically, the test has the following objectives

1. Check to see if all job applications for the job are shown to the employer
2. Check to see if the employer is able to Reject/Deny applicant
3. Check to see if employer can view selected job application
4. Check how efficient the current user experience is
5. Measure the overall user satisfaction with the feature

Test Description

System setup

Since the application is accessible through the browser, the only setup required is a GUI system with a supported browser i.e. active version of Firefox or Chrome.

Starting point

This usability test starts when the user is on the Hiring Guru dashboard and wants to move to viewing all job applications

1. Go to the Hiring Guru URL
2. On the Landing Page, click on the Login button
3. Go to the Dashboard
4. Click on Job Applications

5 - Add referral for a job

Test Objectives

When employers network with other candidates, they should have the ability to submit referrals to certain candidates that they feel would excel at the job. The Hiring Guru makes this process swift and easy for the user, allowing them to submit a referral for any job posted. This is extremely important, because it allows for an Employer to select an applicant for a job referral. To create a job referral, the employer has to list the first and last name of the applicant, applicant's email, reasoning for choosing the applicant, and the applicant's resume.

More specifically, the test has the following objectives

1. Check to see if an Employer is able to fill in all of the requested fields
2. Make sure that the Job referral can't be submitted unless all of the requested fields are filled
3. Check to see if Employer can easily find job referral section
4. Check how efficient the current user experience is
5. Measure the overall user satisfaction with the feature

Test Description

System setup

Since the application is accessible through the browser, the only setup required is a GUI system with a supported browser i.e. active version of Firefox or Chrome.

Starting point

This usability test starts when the user is on the Hiring Guru dashboard and wants to move to adding a job referral.

1. Go to the Hiring Guru URL
2. On the Landing Page, click on the Login button
3. Go to the Dashboard
4. Go to Job Referral and fill out all of the fields

Test Results

Effectiveness

Testcase	Completion	Errors	Comments
Add a new stage to the recruitment process	PASS	None	N/A
Change position of a recruitment stage	PASS	None	Very efficient, users like the one click
Delete a recruitment stage	PASS	None	N/A
See applications for a job	PASS	None	N/A
Add referral for a job	PASS	None	Job Referral had slowest time to use, might look into in the future

Efficiency

Usecase	Benchmark - Average time to complete	Benchmark - Average number of clicks	Benchmark - Average screen navigations
Add a new stage to the recruitment process	Within 30 Second, including time to create stage	Approximately 8 Clicks	No screen navigation, all on the same page
Change position of a recruitment stage	A few seconds	Approximately 3 clicks	No screen navigation, all on the same page
Delete a recruitment stage	Within 10 seconds	Approximately 2 clicks	No screen navigation, all on the same page
See applications for a job	Within 30 seconds	Approximately 5 clicks	No screen navigation, all on the same page
Add referral for a job	Under a minute	Approximately 10 clicks	2 screen navigation

User Satisfaction

Statement	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
It was easy to navigate to the recruitment process page					
It was clear where I had to click to add a new recruitment step stage					
The sequence of stages in the recruitment process was easy to understand					
It was easy to add a new stage					
I think I would use the system frequently					
It was easy to change the sequence of recruitment stages					
I found the system unnecessarily complex					
I thought the system was easy to use					
I think I would need the support of a technical person					

to use this software					
I found these functions were well integrated					
There were too many inconsistencies in the system					
I would be easy for most people to use this system					
I felt very confident using this system					
It was clear where I had to navigate to see all jobs					
It was easy to add a job referral					

QA Test Plan

Tests

1- The backend server shall support TLS 1.1, 1.2 and 1.3

Test Objectives

The objective of this test is to make sure that our backend server supports all variations of TLS for the application. TLS represents the current standard for communication privacy so having our backend server supporting these features is crucial to the quality of our application

Hardware and Software Setup

There is no specific Hardware and Software Setup for this test. Simply submit a report through this link(<https://www.ssllabs.com/ssltest>) and view the results from the report to view with the application that supports TLS.

Features to be Tested

1. Hiring Guru server supports TLS 1.1
2. Hiring Guru server supports TLS 1.2
3. Hiring Guru server supports TLS 1.3

QA Test Plan

Title	Description	Test Input	Expected Output	PASS or FAIL
TLS Test 1	Test to see if the backend server supports TLS 1.1	Submit requested link to https://www.ssllabs.com/ssltest for report	TLS 1.1 is supported and shown in reports	PASS
TLS Test 2	Test to see if the backend server supports TLS 1.3	Submit requested link to https://www.ssllabs.com/ssltest for report	TLS 1.2 is supported and shown in reports	PASS
TLS Test 3	Test to see if the backend server supports TLS 1.3	Submit requested link to https://www.ssllabs.com/ssltest for report	TLS 1.3 is supported and shown in reports	FAIL

2- The application shall be usable in all supported browsers

Test Objectives

The primary objective of this test is to make sure our application is supported in all major browsers such as Chrome, Firefox, or Edge for example. We understand that our users can utilize many different browsers for their personal preferences, so the purpose of this test it to make sure the application is fully functional throughout every browser and no one browser is lacking in quality.

Hardware and Software Setup

There is no specific Hardware and Software Setup. The only setup necessary is to have access to all testing browsers and to utilize the application (<https://hiring-guru.us-west-1.elasticbeanstalk.com/>) in these respected browsers.

Features to be Tested

1. All functions are still operational in every browser

2. All features are in their designated alignment in every browser
3. Every application paged is aligned well to various screen sizes

QA Test Plan

Title	Description	Test Input	Expected Output	PASS or FAIL
Chrome Test	Test to see if application is functional in Chrome	Open the Hiring Guru in Chrome	Hiring Guru fully operational	PASS
Firefox Test	Test to see if application is functional in Firefox	Open the Hiring Guru in Firefox	Hiring Guru fully operational	PASS
Microsoft Edge Test	Test to see if application is functional in Microsoft Edge	Open the Hiring Guru in Microsoft Edge	Hiring Guru fully operational	PASS

3- The application shall be usable on all supported devices

Test Objectives

The primary objective of this test is to make sure our application is supported in all major devices such as Android, IOS, or Microsoft for example. We understand that our users own many different smartphone devices for their personal preferences, so the purpose of this test is to make sure the application is fully functional throughout every device and no one device is lacking in quality. Another major objective within this test is to make sure our application is able to adjust to any given screen size, so that the application still looks neat and presentable on any device.

Hardware and Software Setup

There is no specific Hardware and Software Setup. The only setup necessary is to have access to all testing devices and to utilize the application

(<https://hiring-guru.us-west-1.elasticbeanstalk.com/>) in these respected devices.

Features to be Tested

1. Application Page is adjustable to various screen sizes
2. All features all still functional on various devices
3. Hiring Guru si fully functional on every major smartphone

QA Test Plan

Title	Description	Test Input	Expected Output	PASS or FAIL
IOS Test	Test to see if application is functional on an IOS device	Open the Hiring Guru on an Iphone	Hiring Guru fully operational	FAIL
Samsung Test	Test to see if application is functional on a Samsung Device	Open the Hiring Guru on a Samsung	Hiring Guru fully operational	PASS
Google Test	Test to see if application is functional on a Google Device	Open the Hiring Guru on a Google Pixel	Hiring Guru fully operational	PASS

4- All team member pull requests shall be approved by the team lead

Test Objectives

All team members pull requests(PR for short) must be approved by the team lead before merging with the master branch. The purpose of this test is to make sure all team members' PR's are operational and functional before they are merged in with the designated prototype. This allows for not only an extra set of eyes to look at the work done in the PR, but prevents any major bugs from appearing on the master branch. By having the person who is not only the most experienced team member but understands the project best, reviews every PR, we only improve with every PR pushed to the master branch and do not run any risks with massive failures.

Hardware and Software Setup

There is no specific Hardware and Software Setup. The only setup necessary is to have access to all Github Repositories

(<https://github.com/sfsu-joseo/csc648-848-05-sw-engineering-fall-2022-02>) and view the pull requests section that has all the team members PRs.

Features to be Tested

1. Every PR cannot be merged by PR owner
2. All PR's are approved by team lead before merging
3. Any PR not approved is given comments for correction

QA Test Plan

Title	Description	Test Input	Expected Output	PASS or FAIL
Khushi PR Test	Test to see if Khushi's PR have either approval by team lead or comments for correction	Open Github and view PR section	Either approved PR's or commented PRs for future approval	PASS
Kenny PR Test	Test to see if Kenny's PR have either approval by team lead or comments for correction	Open Github and view PR section	Either approved PR's or commented PRs for future approval	PASS
Mamadou PR Test	Test to see if Mamadou's PR have either approval by team lead or comments for correction	Open Github and view PR section	Either approved PR's or commented PRs for future approval	PASS

5- Application shall not be prone to SQL injection attacks

Test Objectives

The purpose of this objective is to make sure our database server and backend server is well protected and users with malicious intent do not have access to this information. When users create their companies on the Hiring Application and post job applications for candidates to apply to, an abundance of information is being put into the database server. This can range from addresses, to emails, to phone numbers, to even payment information. All of this needs to be protected from SQL injection attacks for not only the safety of the application but the safety of its users.

Hardware and Software Setup

There is no specific Hardware or Software Setup for this test. Certain tests just need to be run on either the application (<https://hiring-guru.us-west-1.elasticbeanstalk.com/>) or the database query for the tes,

Features to be Tested

1. Check whether basic users have access to restricted information
2. Check how functional is our backend server

3. Check how the application responds to unexpected SQL inputs.

QA Test Plan

Title	Description	Test Input	Expected Output	PASS or FAIL
SQL Injection Test 1	To see if the search field will fetch all records	Entering * in a search field	Should not fetch all records	PASS
SQL Injection Test 2	Writing a delete query in the search field	Entering a delete query	Does not Delete Records	PASS
SQL Injection Test 3	Writing a delete query in the post field	Entering a delete query	Does not Delete Records	PASS

Code Review

Selected Code Styling Practices

1. Use PascalCase class names
2. Use all-caps SNAKE_CASE for constants
3. Two line spaces before module-level functions and classes
4. 4-space Tabs
5. Max line character length of 120
6. Name variables, functions, constants, and classes must be easy to understand
7. Large functions must be broken down into smaller ones
8. Instead of duplicating a piece multiple times, define it once and re-use
9. Follow the best practices for Spring Boot and React

10. Separate modules for logically separate features

Code Reviews

File1 - ManageJobs.js

Description

This is the code for one of the most crucial pages on the frontend side. It contains a number of private React components that are used internally by this page. The only component that is exported is the page component itself. As obvious by the name, this page is for managing jobs for a particular role in the company. It has the functionality to create, update and delete a job as well as see a list of all jobs.

Code Review

Review by Team 03 - Alexander Bjeldanes

This is my code review from another team's code. Without much context from what is happening in other files, this is my initial assessment/review of the "JobEditDialog" function in the ManageJobs file.

- Good to have an error check
- Good placeholder information to guide the user
 - Maybe change the "job title" and "workplace" placeholder, but it's not that important
- Clear division of each section of the function
 - Each section has a designated and unique purpose
- Everything else looks good and is well planned out

```
function JobEditDialog(props) {
  return (
    <Dialog
      show={props.show}
      title={props.title}
      actions={props.actions}
    >
      <div>
        <div className={"recruitment-step-errors"}>
          {
            props.errors.map((error, index) => {
              return (
                <div key={`create-rect-step-error-${index}`}
                  className="alert alert-danger" role="alert">
                    {error}
                  </div>
                )
              }
            )
          }
        </div>
      </div>
    </Dialog>
  )
}
```

```

        </div>
    )
    })
}
</div>
<div className="mb-3">
    <label htmlFor="recruitmentStageNameInput"
className="form-label">
        Job Title
    </label>
    <input className="form-control"
id="recruitmentStageNameInput"
        placeholder="Enter job title"
        value={props.name}
        onChange={props.onNameChange}
    />
</div>
<div className="mb-3">
    <label htmlFor="recruitmentStageNameInput"
className="form-label">
        Workplace
    </label>
    <input className="form-control"
id="recruitmentStageNameInput"
        placeholder="San Francisco, CA"
        value={props.workplace}
        onChange={props.onWorkplaceChange}
    />
</div>
<div className="mb-3">
    <label htmlFor="recruitmentStageNameInput"
className="form-label">
        Employment Type
    </label>
    <input className="form-control"
id="recruitmentStageNameInput"
        placeholder="Full-Time, Part-Time, Internship"
        value={props.employment}
        onChange={props.onEmploymentChange}

```

```

        />
      </div>
      <div className="mb-3">
        <label htmlFor="recruitmentStageNameInput"
className="form-label">
          Location
        </label>
        <input className="form-control"
id="recruitmentStageNameInput"
          placeholder="San Francisco, CA"
          value={props.location}
          onChange={props.onLocationChange}
        />
      </div>
      <div className="mb-3">
        <label htmlFor="recruitmentStageDescriptionInput"
className="form-label">
          Description
        </label>
        <textarea className="form-control"
id="recruitmentStageDescriptionInput"
          rows="5" placeholder="Enter description"
          value={props.description}
          onChange={props.onDescriptionChange}
        >
        </textarea>
      </div>
    </div>
  </Dialog>
)
}

```

File2 - RecruitmentProcess.js

Description

This is another file from our frontend application. It contains code for the components related to managing the hiring process for a particular role. Specifically, this page has the functionality to add new stages to the hiring process, modify the sequence of stages and update individual stages. This is one of the core components and unique features of our application.

Code Review

Review by Team 03 - Seth Pavlicek

Overall, this file has good indentation and consistency. It is missing a header and it needs more comments to describe what is happening in each method, component, or constant. One thing that should be considered is to not exceed a line length. There are many lines that are over 100 characters long. However, conforming to this rule may break some of the indentation consistencies. There are more comments below in the code.

This naming scheme is a little confusing. Is the ui supposed to display 'Software Engineer' or is it for 'Software Engineer';

```
const jobPositions = [
  {
    ui: 'Software Engineer',
    server: 'SOFTWARE_ENGINEER'
  },
  {
    ui: 'CEO',
    server: 'CEO'
  },
]

const RecruitmentStepType = {
  Interview: {
    ui: 'Interview',
    server: 'INTERVIEW'
  },
  ProgrammingTest: {
    ui: 'Programming Test',
    server: 'PROGRAMMING_TEST'
  },
}
```

The naming scheme for your recruitment pipelines is fine. Try to include comments. This file is massive and without some serious digging, I have no clue where these constants are being used and why.

```
const RecruitmentPipelineSoftwareEngineer = [
  {
    type: RecruitmentStepType.Interview,
    title: "First Technical Interview",
    detail: "First interview should be with a Software Engineer. It should cover basic" +
      " concepts like OOP, DS and Problem Solving."
  },
  {
    type: RecruitmentStepType.Interview,
    title: "Second Technical Interview",
    detail: "First interview should be with a Software Engineer. It should cover
advanced" +
      " aspects like Algorithms, Design and Networks."
  },
]
```

```

{
  type: RecruitmentStepType.ProgrammingTest,
  title: "HackerRank Test",
  detail: "First interview should be with a Software Engineer. It should cover all" +
    " the technical aspects like OOP, DB, DS, Algorithms and Networks."
},
{
  type: RecruitmentStepType.Interview,
  title: "HR Interview",
  detail: "HR Interview should cover discussion about personality and salary package."
},
]

const RecruitmentPipelineCEO = [
  {
    type: RecruitmentStepType.Interview,
    title: "Interview Evaluation",
    detail: "Interview with the founder to discuss past experience and assess personality traits."
  },
  {
    type: RecruitmentStepType.Interview,
    title: "HR Interview",
    detail: "HR Interview should cover discussion about personality and salary package."
  },
]

const RecruitmentPipelinePositionMap = {
  SOFTWARE_ENGINEER: RecruitmentPipelineSoftwareEngineer,
  CEO: RecruitmentPipelineCEO
}

```

Good comment at the top describing what this is. You may want to include a comment above “props.errors.map” and “Object.keys(RecruitmentStepType).map” and describe what it is supposed to do.

```

function RecruitmentStepDialog(props) {
  // dialog component for adding a new step to the recruitment process
  return (
    <Dialog
      show={props.show}
      title={"Add a new stage to the recruitment process"}
    />
  )
}

```



```

    actions={props.actions}
  >
  <div>
    <div className={"recruitment-step-errors"}>
      {
        props.errors.map((error, index) => {
          return (
            <div key={`create-rect-step-error-${index}`} className="alert
              alert-danger" position="alert">
              {error}
            </div>
          )
        })
      }
    </div>
    <div className="mb-3">
      <label htmlFor="recruitmentStageTitleInput" className="form-label">
        Title
      </label>
      <input className="form-control" id="recruitmentStageTitleInput"
        placeholder="Enter title ..."
        value={props.stepTitle}
        onChange={props.onTitleChange}
      />
    </div>
    <div className="mb-3">
      <label htmlFor="recruitmentStageDescriptionInput" className="form-label">
        Description
      </label>
      <textarea className="form-control" id="recruitmentStageDescriptionInput"
        rows="5" placeholder="Enter description ..."
        value={props.stepDescription}
        onChange={props.onDescriptionChange}
      >
      </textarea>
    </div>
    <select className="form-select" aria-label="Default select example"
      value={
        props.selectedStageType ||

```

```

        "Select the type of this recruitment stage"
      }
      onChange={props.onStageTypeChange}
    >
    <option disabled>Select the type of this recruitment stage</option>
    {
      Object.keys(RecruitmentStepType).map((k) => {
        return <option
          key={`recruitment-stage-type-select-${k}`}
          value={k}
          >{RecruitmentStepType[k].ui}</option>
        })
      }
    </select>
  </div>
</Dialog>
)
}

```

Other than the main component, I have no clue what these constants are keeping track of.

```

function RecruitmentProcess() {
  // main recruitment process page
  const appContext = useContext(ApplicationContext);
  const [recruitmentProcessState, setRecruitmentProcessState] = useState({
    selectedjobPosition: jobPositions.All,
    recruitmentProcess: undefined
  })
  const [createRecruitmentStepFormState, setCreateRecruitmentStepFormState] = useState({
    showCreationDialog: false,
    showModificationDialog: false,
    title: "",
    description: "",
    type: undefined,
    errors: [],

  })
  const [modifyRecruitmentStepFormState, setModifyRecruitmentStepFormState] = useState({
    showModificationDialog: false,
    title: "",
    description: "",

```

```

    type: undefined,
    errors: [],
    index: undefined
  })

const moveStepUp = (index) => {
  // move the step identified by the index one step upward
  let process = recruitmentProcessState.recruitmentProcess
  const saveState = process[index - 1]
  process[index - 1] = process[index]
  process[index] = saveState
  setRecruitmentProcessState({
    ...recruitmentProcessState,
    recruitmentProcess: process
  })
}

const moveStepDown = (index) => {
  // move the step identified by the index one step downward
  let process = recruitmentProcessState.recruitmentProcess
  const saveState = process[index + 1]
  process[index + 1] = process[index]
  process[index] = saveState
  setRecruitmentProcessState({
    ...recruitmentProcessState,
    ...recruitmentProcessState,
    recruitmentProcess: process
  })
}

const modifyRecruitmentStep = () => {
  // update a recruitment step
  let newSteps = []
  for (let i = 0; i < recruitmentProcessState.recruitmentProcess.length; i++) {
    if (i === modifyRecruitmentStepFormState.index) {
      newSteps.push({
        type: RecruitmentStepType[modifyRecruitmentStepFormState.type],
        title: modifyRecruitmentStepFormState.title,
        detail: modifyRecruitmentStepFormState.description
      })
    }
  }
}

```

```

    }
    else {
      newSteps.push(recruitmentProcessState.recruitmentProcess[i])
    }
  }
  setModifyRecruitmentStepFormState({
    ...modifyRecruitmentStepFormState,
    showModificationDialog: false
  })
  setRecruitmentProcessState({
    ...recruitmentProcessState,
    recruitmentProcess: newSteps
  })
}

const createRecruitmentStep = () => {
  // validate recruitment step info and create a new step
  let errors = []
  if (!createRecruitmentStepFormState.title ||
createRecruitmentStepFormState.title.length === 0) {
    errors.push("Please provide a title for this stage")
  }
  if (!createRecruitmentStepFormState.description ||
createRecruitmentStepFormState.description.length === 0) {
    errors.push("Please provide a description for this stage")
  }
  if (!createRecruitmentStepFormState.type ||
createRecruitmentStepFormState.type.length === 0) {
    errors.push("Please select the type of this stage")
  }
  if (errors.length > 0) {
    setCreateRecruitmentStepFormState({
      ...createRecruitmentStepFormState,
      showCreationDialog: true,
      errors: errors,
    })
  }
  else {
    setCreateRecruitmentStepFormState({

```

```

        ...createRecruitmentStepFormState,
        showCreationDialog: false
    })
    setRecruitmentProcessState({
        ...recruitmentProcessState,
        recruitmentProcess: [
            ...recruitmentProcessState.recruitmentProcess,
            {
                type: RecruitmentStepType[createRecruitmentStepFormState.type],
                title: createRecruitmentStepFormState.title,
                detail: createRecruitmentStepFormState.description
            }
        ]
    })
}

const removeRecruitmentStep = (index) => {
    // delete recruitment step identified by the index
    let newSteps = []
    for (let i = 0; i < recruitmentProcessState.recruitmentProcess.length; i++) {
        i !== index && newSteps.push(recruitmentProcessState.recruitmentProcess[i])
    }
    appContext.closeDialog()
    setRecruitmentProcessState({
        ...recruitmentProcessState,
        recruitmentProcess: newSteps
    })
}

```

I think your naming scheme is doing a good job at describing what is being displayed in this page but it may still need a few comments.

```

return (
    <div>
        <RecruitmentStepDialog
            show={createRecruitmentStepFormState.showCreationDialog}
            title={"Add a new stage to the recruitment process"}
            actions={[
                {
                    title: "Close",

```

```

        handler: () => {
            setCreateRecruitmentStepFormState({
                ...createRecruitmentStepFormState,
                showCreationDialog: false
            })
        },
        variant: "secondary"
    },
    {
        title: "Create Step",
        handler: createRecruitmentStep,
        variant: "primary"
    }
]}
errors={createRecruitmentStepFormState.errors}
onTitleChange={(e) => {
    setCreateRecruitmentStepFormState({
        ...createRecruitmentStepFormState,
        title: e.target.value
    })
}}
onDescriptionChange={(e) => {
    setCreateRecruitmentStepFormState({
        ...createRecruitmentStepFormState,
        description: e.target.value
    })
}}
onStageTypeChange={(e) => {
    setCreateRecruitmentStepFormState({
        ...createRecruitmentStepFormState,
        type: e.target.value
    })
}}
selectedStageType={createRecruitmentStepFormState.type}
stepTitle={createRecruitmentStepFormState.title}
stepDescription={createRecruitmentStepFormState.description}
/>
<RecruitmentStepDialog
    show={modifyRecruitmentStepFormState.showModificationDialog}

```

```
title={"Add a new stage to the recruitment process"}
actions=[
  {
    title: "Close",
    handler: () => {
      setModifyRecruitmentStepFormState({
        ...modifyRecruitmentStepFormState,
        showModificationDialog: false
      })
    },
    variant: "secondary"
  },
  {
    title: "Modify stage",
    handler: modifyRecruitmentStep,
    variant: "primary"
  }
]
errors={modifyRecruitmentStepFormState.errors}
onTitleChange={(e) => {
  setModifyRecruitmentStepFormState({
    ...modifyRecruitmentStepFormState,
    title: e.target.value
  })
}}
onDescriptionChange={(e) => {
  setModifyRecruitmentStepFormState({
    ...modifyRecruitmentStepFormState,
    description: e.target.value
  })
}}
onStageTypeChange={(e) => {
  setModifyRecruitmentStepFormState({
    ...modifyRecruitmentStepFormState,
    type: e.target.value
  })
}}
selectedStageType={modifyRecruitmentStepFormState.type}
stepTitle={modifyRecruitmentStepFormState.title}
```

```

        stepDescription={modifyRecruitmentStepFormState.description}
      />
    <div className={"page-container"}>
      <Breadcrumbs>
        <Breadcrumbs.Item href="/dashboard/home">Dashboard</Breadcrumbs.Item>
        <Breadcrumbs.Item active>Recruitment: Hiring Pipeline</Breadcrumbs.Item>
      </Breadcrumbs>
      <h1>Design Hiring Pipeline</h1>
      <div className="position-selection-control">
        <div className={"position-selection-header"}>
          <h5>Select a position to configure its Recruitment Process:</h5>
        </div>
        <div className={"position-selection-dropdown"}>
          <div className="input-group input-group-sm">
            <Dropdown className={"input-group-text"}>
              <Dropdown.Toggle id="dropdown-basic">
                <Filter /> Select Job Position
              </Dropdown.Toggle>
              <Dropdown.Menu>
                {
                  jobPositions.map((jobPosition) => {
                    return (
                      <Dropdown.Item key={jobPosition.ui}
onClick={
(e) => {
                      setRecruitmentProcessState({
                        ...recruitmentProcessState,
                        recruitmentProcess:
RecruitmentPipelinePositionMap[jobPosition.server],
                        selectedjobPosition: jobPosition
                      })
                    }
                  })
                }
              </Dropdown.Item>
            )
          })
        }
      </Dropdown.Menu>
    </Dropdown>

```



```

        </div>
      </div>
    </div>
    {
      recruitmentProcessState.recruitmentProcess !== undefined &&
      <div className={"recruitment-pipeline-detail-container"}>
        <div className={"recruitment-pipeline-controls
container-vcenter-hright"}>
          <Button
            variant="primary"
            onClick={() => {
              setCreateRecruitmentStepFormState({
                ...createRecruitmentStepFormState,
                showCreationDialog: true
              })
            }}
          >
            Add a new stage
          </Button>
        </div>
        <div className={"recruitment-pipeline-detail"}>
          {
            recruitmentProcessState.recruitmentProcess.map((step, index)
=> {
              const lenSteps =
recruitmentProcessState.recruitmentProcess.length
              return (
                <div key={`index-${step.title}`}
className={"recruitment-pipeline-step row"}>
                  <div className={"step-number-container
container-all-center col-1"}>
                    <Button disabled={true}
className={"step-number btn btn-circle btn-sm"}
                      variant="primary">{index +
1}</Button>
                  </div>
                  <div className={"step-detail-container col-9"}>
                    <div className={"step-detail"}>

```

```

<div className={'step-title
h5'}>{step.title}</div>

<div
className={'step-type'}>{step.type.ui}</div>

<div
className={'step-description'}>{step.detail}</div>
</div>
</div>
<div className={"step-actions-container
container-all-center col-2"}>

  <div className={"step-actions"}>
    <div className={"icon-container"}>
      <OverlayTrigger
        placement="bottom"
        overlay={
          <Tooltip>Update
stage</Tooltip>
        }
      >
      <button type="button"
        className="btn btn-circle
btn-sm btn-success"

        onClick={() => {
          let stepType
          Object.keys(
RecruitmentStepType).map((k) => {
            if
(RecruitmentStepType[k].ui === step.type.ui) {
              stepType = k
            }
          })
        }}
      >

setModifyRecruitmentStepFormState({
  ...modifyRecruitmentStepFormState,
  title: step.title,
  description:
step.detail,

```

```

                                type: stepType,
showModificationDialog: true,
                                index: index
                            })
                        }}
                    >
                        <PencilFill></PencilFill>
                    </button>
                </OverlayTrigger>
            </div>
            <div className={"icon-container"}>
                <OverlayTrigger
                    placement="bottom"
                    overlay={
                        <Tooltip>Delete
                    }
                >
                    <button type="button"
                        className="btn btn-circle
                            btn-sm btn-danger"
                        onClick={() =>
                            appContext.openDialog(
                                "Are you sure?",
                                [
                                    {
                                        title: "Close",
                                        handler:
                                            appContext.closeDialog,
                                        variant:
                                            "secondary"
                                    },
                                    {
                                        title: "Remove
stage",
                                        handler: () =>
removeRecruitmentStep(index),

```

```

                                variant:
"primary"
                                }
                                ],
                                "Once deleted, this can't
be undone. " +
                                "Are you sure you want to
proceed?"
                                })
                                >
                                <Trash3Fill></Trash3Fill>
                                </button>
                                </OverlayTrigger>
                                </div>
                                {
                                index !== 0 && <div
className={"icon-container"}>
                                <OverlayTrigger
                                placement="bottom"
                                overlay={
                                <Tooltip>Move stage one
step up</Tooltip>
                                }
                                >
                                <button type="button"
                                className="btn
btn-circle btn-sm btn-warning">
                                <ArrowBarUp></ArrowBarUp>
                                </button>
                                </OverlayTrigger>
                                </div>
                                }
                                {
                                index !== lenSteps - 1 && <div
className={"icon-container"}>
                                <OverlayTrigger
                                placement="bottom"
                                overlay={

```



```
export default RecruitmentProcess;
```

Internal Code Reviews

Description

The following snapshots are examples of the internal code review process within our team. Whenever a PR is created, our team lead spends time reviewing the files to make sure the code is satisfactory and able to be merged in the master branch.

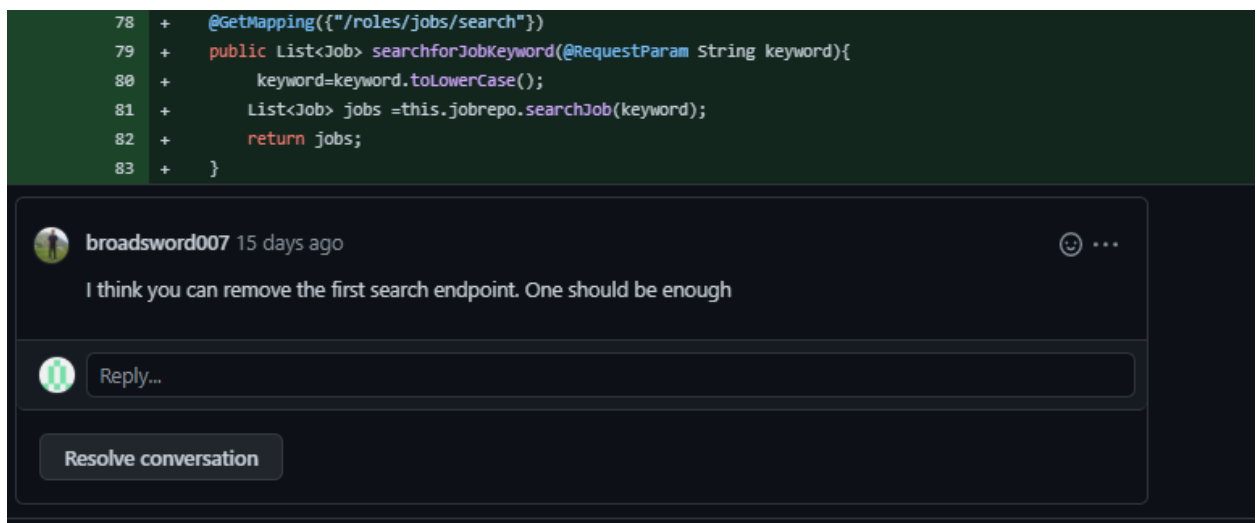
Code Review Snapshots

1.





The screenshot shows a code review interface. At the top, a code snippet is visible: `17 + public class hiring_process_stage {` and `18 + @Id`. Below the code, a comment box contains the text "Comment on lines +17 to +18". The comment itself is from user "broadsword007" 7 days ago, stating: "Please add an additional property to this model. Name it `stageNumber`". There is a "Reply..." input field and a "Resolve conversation" button at the bottom.

2.




The screenshot shows a code review interface. At the top, a code snippet is visible: `78 + @GetMapping("/{roles/jobs/search"})`, `79 + public List<Job> searchforJobKeyword(@RequestParam String keyword){`, `80 + keyword=keyword.toLowerCase();`, `81 + List<Job> jobs =this.jobrepo.searchJob(keyword);`, `82 + return jobs;`, and `83 + }`. Below the code, a comment box contains the text "I think you can remove the first search endpoint. One should be enough". There is a "Reply..." input field and a "Resolve conversation" button at the bottom.

```
16 + )
17 + public class hiring_process {
```

 **broadsword007** 8 days ago 

Please follow the naming conventions we are using across our codebase





Resolve conversation


3.

```
...backend/hiring_guru_be/src/main/java/com/hiringguru/hiring_guru_be/models/HiringProcess.java

14 + @Table(
15 +     name = "hiring_process"
16 + )
17 + public class hiring_process {
```

 **broadsword007** 8 days ago 

Please follow the naming conventions we are using across our codebase





Resolve conversation

4.


```
...nd/hiring_guru_be/src/main/java/com/hiringguru/hiring_guru_be/models/HiringProcessStage.java

Comment on lines +45 to +52

45 + public Role(String title, String expectations, String benefits, Company company) {
46 +     this.id = this.id;
47 +     this.title = title;
48 +     this.type = type;
49 +     this.description = description;
50 +     this.hiring_process = hiring_process;
51 +
52 + }
```

 **broadsword007** 8 days ago 

What is the purpose of adding this function?



Resolve conversation

5.



6.

Self-check on Best Practices for Security

Following are the major assets that we are protecting

1. Database
2. AWS account and Infrastructure
3. Code
4. Documentation
5. In-flight data

Following is the detail about the security of these items

Database

Our DB is provisioned in AWS. We have tried to ensure maximum security for our database as described below

1. We are using the latest version of Postgres to make sure we are using the stable and most secure build
2. All of our AWS users have read-only permission on the database. They cannot change critical settings on the database
3. We have enforced a max provision limit on the size of the DB. The DB size cannot increase by 20GB. So even if someone gets access to our DB the limit of damage is limited.
4. We are planning to limit DB access to our production server only.

AWS Account and Infrastructure

We are using AWS for all of our infrastructures. The security of our AWS account is very important. We have taken the following steps

1. Enabled two-factor authentication on the root AWS account
2. Limited the usage of the root account.
3. Created IAM users with minimum permission required, primarily with read-only access

4. Using AWS security groups as a line of defense
5. Using AWS-provided application load balancer to mitigate network attacks.

Code

Code is one of the most critical assets and we have taken steps to ensure it's secure

1. The code repository is private
2. Only the team members have access to repository
3. The master branch is protected against direct commits.
4. All PRs must get approved by the Team lead to get merged
5. Continuous integration pipeline that makes sure that the code builds correctly

Documentation

We are using Google for saving our documents. All of our Google docs are shared only amongst the team members over email. At the end of a milestone, the documents are converted to PDFs and submitted. A version of the documents is also kept in the code repo which is also secured.

In-flight data

We understand the importance of keeping the inflight data secure.

1. All of our inflight data is encrypted.
2. We provide support for TLS 1.1, 1.2, and 1.3 with strong encryption algorithms
3. Our front end is server over SSL
4. The backend is also server over SSL
5. We support strong encryption algorithms

The security report for our servers is attached below

Certificate #1: RSA 2048 bits (SHA384withRSA)



Server Key and Certificate #1



Subject	hiring-guru-backend-prod.us-west-1.elasticbeanstalk.com Fingerprint SHA256: 0216f5c846e1e9fa9d12affbc1dcc58a2238ed1df6ffa6509d8f835521a0ca50 Pin SHA256: Ilg2iXA2TbFzLG3U/ysrWwz5A6X+ByUmgnGUzuWn3Ls=
Common names	hiring-guru-backend-prod.us-west-1.elasticbeanstalk.com
Alternative names	hiring-guru-backend-prod.us-west-1.elasticbeanstalk.com
Serial Number	00b380926a474a8058dabef3f90294581c
Valid from	Thu, 10 Nov 2022 00:00:00 UTC
Valid until	Wed, 08 Feb 2023 23:59:59 UTC (expires in 2 months and 7 days)
Key	RSA 2048 bits (e 65537)
Weak key (Debian)	No
Issuer	ZeroSSL RSA Domain Secure Site CA AIA: http://zerossl.crt.sectigo.com/ZeroSSLRSADomainSecureSiteCA.crt
Signature algorithm	SHA384withRSA
Extended Validation	No
Certificate Transparency	Yes (certificate)
OCSP Must Staple	No
Revocation information	OCSP OCSP: http://zerossl.ocsp.sectigo.com
Revocation status	Good (not revoked)
DNS CAA	No (more info)
Trusted	Yes Mozilla Apple Android Java Windows

Configuration



Protocols

TLS 1.3	No
TLS 1.2	Yes
TLS 1.1	Yes
TLS 1.0	Yes
SSL 3	No
SSL 2	No



Cipher Suites

# TLS 1.2 (suites in server-preferred order)			
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)	ECDH secp256r1 (eq. 3072 bits RSA) FS		128
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xc027)	ECDH secp256r1 (eq. 3072 bits RSA) FS	WEAK	128
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)	ECDH secp256r1 (eq. 3072 bits RSA) FS	WEAK	128
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)	ECDH secp256r1 (eq. 3072 bits RSA) FS		256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028)	ECDH secp256r1 (eq. 3072 bits RSA) FS	WEAK	256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)	ECDH secp256r1 (eq. 3072 bits RSA) FS	WEAK	256
TLS_RSA_WITH_AES_128_GCM_SHA256 (0x9c)		WEAK	128
TLS_RSA_WITH_AES_128_CBC_SHA256 (0x3c)		WEAK	128
TLS_RSA_WITH_AES_128_CBC_SHA (0x2f)		WEAK	128
TLS_RSA_WITH_AES_256_GCM_SHA384 (0x9d)		WEAK	256
TLS_RSA_WITH_AES_256_CBC_SHA256 (0x3d)		WEAK	256
TLS_RSA_WITH_AES_256_CBC_SHA (0x35)		WEAK	256



Protocol Details

DROWN	No, server keys and hostname not seen elsewhere with SSLv2	
	(1) For a better understanding of this test, please read this longer explanation	
	(2) Key usage data kindly provided by the Censys network search engine; original DROWN website here	
	(3) Censys data is only indicative of possible key and certificate reuse; possibly out-of-date and not complete	
Secure Renegotiation	Supported	
Secure Client-Initiated Renegotiation	No	
Insecure Client-Initiated Renegotiation	No	
BEAST attack	Not mitigated server-side (more info) TLS 1.0: 0xc013	
POODLE (SSLv3)	No, SSL 3 not supported (more info)	
POODLE (TLS)	No (more info)	
Zombie POODLE	No (more info) TLS 1.2: 0xc027	
GOLDENDOODLE	No (more info) TLS 1.2: 0xc027	
OpenSSL 0-Length	No (more info) TLS 1.2: 0xc027	
Sleeping POODLE	No (more info) TLS 1.2: 0xc027	
Downgrade attack prevention	Yes, TLS_FALLBACK_SCSV supported (more info)	
SSL/TLS compression	No	
RC4	No	
Heartbeat (extension)	No	
Heartbleed (vulnerability)	No (more info)	
Ticketbleed (vulnerability)	No (more info)	
OpenSSL CCS vuln. (CVE-2014-0224)	No (more info)	
OpenSSL Padding Oracle vuln. (CVE-2016-2107)	No (more info)	
ROBOT (vulnerability)	No (more info)	
Forward Secrecy	With modern browsers (more info)	
ALPN	Yes h2 http/1.1	
NPN	Yes h2 http/1.1	
Session resumption (caching)	Yes	
Session resumption (tickets)	Yes	
OCSP stapling	No	
Strict Transport Security (HSTS)	No	
HSTS Preloading	Not in: Chrome Edge Firefox IE	
Public Key Pinning (HPKP)	No (more info)	
Public Key Pinning Report-Only	No	
Public Key Pinning (Static)	No (more info)	
Long handshake intolerance	No	
TLS extension intolerance	No	
TLS version intolerance	No	
Incorrect SNI alerts	No	
Uses common DH primes	No, DHE suites not supported	
DH public server param (Ys) reuse	No, DHE suites not supported	
ECDH public server param reuse	No	
Supported Named Groups	secp256r1, secp521r1, brainpoolP512r1, brainpoolP384r1, secp384r1, brainpoolP256r1, secp256k1, sect571r1, sect571k1, sect409k1, sect409r1, sect283k1, sect283r1 (server preferred order)	
SSL 2 handshake compatibility	Yes	

Self-check: Adherence to Original Non-functional Specs

1. The application shall have efficient response time **DONE**
2. The application shall have an easy to use platform **DONE**
3. The application shall have sufficient space for all data **DONE**
4. The application should have an option for extra space **DONE**
5. General User shall be able to create an account **DONE**
6. General User should be able to mark required regions in account registration **ON TRACK**
7. General User shall be able to add required information to account **DONE**
8. General User shall be able to add a resume to account **DONE**
9. General User shall be able to have other accounts linked to this account **ON TRACK**
10. General User should be able to save jobs to a folder **ON TRACK**
11. General User shall be able to search for jobs **DONE**
12. General User should be able to sort jobs by date applied **ON TRACK**
13. General User shall be able to view necessary information underneath job **DONE**
14. General User shall be able to add necessary information underneath job **DONE**
15. The application should have an easy to follow design **DONE**
16. The application should have a unique and Colorful Design **DONE**
17. The application should have no unnecessary information on the page **DONE**
18. The application shall have adequate Connection to Wi-Fi **DONE**
19. The application shall have see Wi-Fi connection status **DONE**
20. General User shall be able to see visibility status with other companies **ON TRACK**
21. General User shall be able to customize profile picture for account **DONE**
22. The application should have a career history to be added to account **DONE**
23. General User should be able to an academic history to be added to account **ON TRACK**
24. General User should be able to any achievements to be added to account **ON TRACK**
25. General User shall be able to add custom bio to account **DONE**
26. The application shall have strong security **DONE**
27. General User should be able to sign in using 2 Factor Authentication **DONE**

28. General User should be able to answer security questions in case of account
lockout **ON TRACK**
29. General User shall be able to hide password when typing **DONE**
30. General User shall be able to view password when typing **DONE**
31. General User shall be able to update password **DONE**
32. General User shall be able to choose username that's not taken yet **DONE**
33. General User shall be able to choose email that's not taken yet **DONE**
34. General User should be able to choose a secure password with specific security
requirements **DONE**
35. General User shall be able to define ideal location preferences **DONE**
36. General User shall be able to define ideal job preferences **DONE**
37. General User shall be able to recover lost account **DONE**
38. General User shall be able to notify users of any updates/changes **ON TRACK**
39. General User should be able to deliver reminders to users **ON TRACK**
40. General User should be able to customize reminder time **ON TRACK**
41. General User should be able to customize reminder date **ON TRACK**
42. General User shall be able to see all future schedules **ON TRACK**
43. General User shall be able to contact users easily **DONE**
44. General User should be able to make your account private or public **ON TRACK**
45. General User shall be able to see past completed interviews **ON TRACK**
46. General User should be able to see previous emails sent **ON TRACK**
47. General User should be able to see points of contact underneath jobs (if
applicable) **DONE**
48. General User should be able to see the location of job **DONE**
49. General User shall be able to easily see what the job is requiring **DONE**
50. General User should be able to be informed if this job is a good match **DONE**
51. General User should be able to be informed if the job is a bad match **DONE**
52. General User shall be able to able to assign an evaluation score to a candidate from
an interview **DONE**
53. General User should be able to know the evaluation score of an interview **ON
TRACK**
54. General User should be able to know the name and job title of your interviewer
DONE
55. General User shall be able to have the link sent to them for interview **ON TRACK**
56. General User shall be able to see what jobs they have previously viewed **ON
TRACK**

- 57. General User shall be able to see what jobs they have already applied for **ON TRACK**
- 58. The application shall be should to able to indicate job preferences **DONE**
- 59. The application shall be should to have potential jobs sent to them daily **DONE**
- 60. The application shall be should to have potential jobs sent to them weekly **DONE**
- 61. The application shall be should to have no preferences sent at all **ON TRACK**
- 62. General User shall be able to request updates for job postings of a specific company **DONE**
- 63. General User shall be able to login easily into account **DONE**
- 64. General User shall be able to log out easily of account **DONE**
- 65. General User shall be able to option to have account sign in remembered **DONE**
- 66. General User shall be able to able to delete account **DONE**
- 67. General User shall be able to able to deactivate account **DONE**
- 68. General User should be able to able to receive notifications for scheduled interviews **ON TRACK**
- 69. General User should be able to customize notifications for scheduled interviews **ON TRACK**
- 70. General User should be able to choose between light and dark theme for application **ON TRACK**
- 71. General User shall be able to cancel interviews **ON TRACK**
- 72. General User should be able to request different interview times **ON TRACK**
- 73. General User should be able to see who will be attending the Interviews **ON TRACK**
- 74. General User should be able to RSVP for an Interview **ON TRACK**
- 75. General User shall be able to deny an Interview denial **ON TRACK**
- 76. General User shall be able to remind attendees for Interview 15 minutes prior **ON TRACK**
- 77. General User shall be able to gives users options to change reminder time **ON TRACK**
- 78. General User should be able to add description within scheduled interview **ON TRACK**
- 79. General User should be able to request to create any non-interview meetings for questions **ON TRACK**
- 80. Authorized User shall be able to to see which exact candidate is applying **DONE**
- 81. Authorized User shall be able to to view resume attached to candidate profile **DONE**
- 82. Authorized User should be able to contact candidate freely **DONE**

83. Authorized User shall be able to schedule meeting with candidate freely **DONE**
84. Authorized User shall be able to score candidate accordingly **DONE**
85. Authorized User shall be able to see score/interview notes freely **DONE**
86. Authorized User shall be able to see available contact information for candidate **DONE**
87. Authorized User should be able to customize profile based on employment within company **DONE**
88. Authorized User should be able to be verifiable for better perks **DONE**
89. Authorized User should be able to see other verified accounts **DONE**
90. General User should be able to follow other users on the site **DONE**
91. General User should be able to see other users who are on the site **DONE**
92. Authorized User shall be able to see who you follow and what jobs they have applied for **DONE**
93. The Application should be able to recommend usernames and passwords for registering users **ON TRACK**
94. The Application should be able to recommend passwords for users non-registering users **ON TRACK**
95. The Application should be able to recommend usernames for users non-registering users **ON TRACK**
96. The Application should be able to have sufficient space for all users on platform **DONE**
97. The Application should be able to have a limit for resume size **DONE**
98. The Application should be able to have a preferences in how resume should be submitted **DONE**
99. The Application should be able to ask user if they are a robot **DONE**
100. The Application shall be able to deactivate user if they are a robot **DONE**

Team Contributions

Contributions

Farhan Haider

- Implemented the Create, Read, Update, Delete functionality for Employee
- Helped Kenny fix the manage jobs page
- Did internal code reviews - reviewed PRs from team members
- Did code review for Team 3
- Worked on Self-Check on Best practices for Security
- Created the basic skeleton for Usability tests and completed the first usability test
- Finalized the non-functional requirements for QA test
- Helped Khushi and Eric with the backend work
- Organized team meetings and planned the overall work

Kenny Leong

- Frontend implementation of backend APIs
 - Employees
 - Jobs
 - Job Roles
- UI Bug Fixes/UI improvements

Khushi Khanna

- Worked on the Backend
 - Created Company, Role, Job Models and CRUDs
 - Added a Job Search API
- Helped with the document
 - Sections 1&2

Mamadou Bah

- Worked on the backend
 - Created Role, HiringProcess, and HiringProcessStage Models
 - Helped with Job Application CRUD
- Helped with the document
 - Section 1-3
 - Organization and formatting

Eric Leow

- Backend
 - Helped Create models Role, HiringProcess and HiringProcessStage
 - Hiring Pipeline CRUD
 - Created models Job, JobApplication and Referral

- Tested Job, Role and Company API. To see if any fixes were required.
- Wireframe
 - Login and Signup Page
 - Apply for a job
 - Job Referral

Scores

- Kenny Leong - 8/10
- Mamadou Bah - 6/10
- Eric Leow - 6.5/10
- Khushi Khanna - 6.5/10