Technical Solution Design

JobsAustralia.tech

Group Members: Ozlem Kirmizi (s3491115@student.rmit.edu.au)
Kim Luu (s3536578@student.rmit.edu.au)
Aaron Horler (s3481341@student.rmit.edu.au)
Melissa Nguyen (s3476694@student.rmit.edu.au)
Dennis Mihalache (s3434719@student.rmit.edu.au)

Supervisor: Amir Homayoon Ashrafzadeh

Project Name: JobsAustralia.tech

Commercial in confidence

Document Control

Distribution

Version	Issued	Recipient	Position
1.0	16/10/2017	Homy Ashrafzadeh	Product owner
1.1	05/11/2017	Homy Ashrafzadeh	Product owner

Staff or Entities Consulted

NAME	Position / Organization
Homy Ashrafzadeh	Product owner

Related Documents

Name	Author	Description
User Stories	Team	A list of all user stories implemented.
User Manual	Team	Instructions on using the system.
Test Plan	Team	Outline of the project testing methodology.
Statement of Contributions	Team	Breakdown of workload by team members.
Risk Register	Team	Analysis of potential risks to the project.
Proposed Assessment Formula	Team	Agreed formula of project assessment.
Project Charter	Team	Outline of the project charter.
Peer Reviews	Team	Detailed assessment of team contribution.
Meeting Minutes	Team	Log a all team meetings.
Learning Outcomes	Team	Analysis of knowledge expansion.
Issue Register	Team	Log of all issues experienced in the project.
Final Schedule Plan	Team	Final outline of task completion schedule.
Development Guide	Team	Instructions on configuring development environment.

Preface

The purpose of this document is to outline the technical design of the system and to provide an overview for its implementation.

Table of Contents

1 INTRODUCTION

2 TECHNICAL ENVIRONMENT

3 OVERALL ARCHITECTURE

4 SYSTEM ARCHITECTURE

4.1 FUNCTIONALITIES/FEATURES

5 DATABASE ARCHITECTURE

6 IMPLEMENTATION INSTRUCTIONS

7 NON-FUNCTIONAL SPECIFICATIONS

8 SUMMARY OF TEST RESULTS

9 KNOWN ISSUES & RISKS

10 REFERENCES

11 APPENDICES

1 Introduction

This project, *JobsAustralia.tech*, is the creation of a job matchmaking website that aims to serve job seekers, and employers.

By utilising a matchmaking algorithm, the website allows employers to list the skills, experience, and level of education they require in a potential employee. Job seekers can register their skills, education, and experience. The system then matches job seekers with jobs accordingly.

Conversely, the system matches job seekers who have applied to a job (applicants) to the employer.

Simplified, the matchmaking algorithm calculates a percentage match based on a comparison of a job seeker's skills and a job's skill requirements. Education and experience are also taken into consideration when matching applicants to employers. See "4.1.1 Perform matchmaking on jobs to job seekers" and "4.1.2 Perform matchmaking on employers to applicants" for more information.

The system also integrates with popular social programming, and Git repository hosting, website GitHub. On registration, job seekers are asked for their GitHub username - which the system uses to autofill various supported skills, and report matching repositories owned by the applicant to employers. See "4.1.3 Integrate with GitHub" for more information.

The technical environment of the project includes the use of various tools and technologies. Laravel was used as a PHP framework, Apache as an HTTP web server, MySQL for database management, and GitHub for Git version control and collaboration. See *"2. Technical Environment"* for more information.

The estimated level of complexity of the project out of a score of 5 is 3.0 (moderate). The integration of familiar technologies indicated that some of the team members were equipped with the skills needed to complete the functions of the project. However, the team members who did not have experience in using these technologies required some level of support.

The project aims to benefit job seekers and employers. **Job seekers** are benefitted by the system utilising matchmaking to display jobs suited to their abilities, and, in the case of GitHub integration, reporting their open source work to employers. **Employers** are equally benefitted by matchmaking, as they experience reductions in the time spent manually sorting applicants to job positions advertised via traditional means.

2 Technical Environment

The team developed the project on different platforms. The table below details this.

Team member	Development platform
Ozlem Kirmizi	Windows 7
Kim Luu	MacOS 10
Aaron Horler	Ubuntu 17.04 & 17.10
Melissa Nguyen	Windows 10
Dennis Mihalache	Windows 10

The team made decisions to use various technologies based on *documentation*, *support*, *recommendations*, and *prior knowledge*.

Documentation refers to the availability of documents used to learn the technology. *Support* refers to the continued support of the technology by its developers. *Recommendations* refers to information we have received from peers. *Prior knowledge* refers to knowledge we already possessed - either as a result of formal education, personal education, or past experience.

In detail, technologies used include the following.

PHP

Version 7.1.11

Documentation	Support	Recommendations	Prior knowledge
Very good	Very good	None	Very good

PHP 7.1.11 was released on the 26th of October 2017. The language is very well documented, and, although the team had not received any specific recommendation for PHP, group members have had experience via Web Programming, Secure Electronic Commerce, Building IT Systems, and personal projects.

PHP-FPM (FastCGI Process Manager) was used on the production server for compatibility with the Apache event mpm - which was required for HTTP/2 support.

In the team’s development environments, a combination of PHP 7.1 and PHP 5.6 were used. This was required due to compatibility issues, although, it caused no issues.

Laravel

Version 5.4/5.5

Documentation	Support	Recommendations	Prior knowledge
Very good	Very good	Very good	Poor

Laravel 5.5 was released on the 30th of August 2017. The framework is very well documented, and, although the team had no prior experience with Laravel, the group had received multiple positive recommendations.

In the project's GitHub repositories (see "GitHub" under this section for more information), the *master* branch uses Laravel 5.4, and the *laravel-5.5* branch uses Laravel 5.5. Deployment to the production server is from the *laravel-5.5* branch, and development is primarily in completed the *master* branch. All Laravel 5.4 code in the project is forward compatible with Laravel 5.5.

MySQL

Version 5.7.20

Documentation	Support	Recommendations	Prior knowledge
Very good	Very good	None	Good

MySQL 5.7.20 was released on the 16th of October 2017. The database management system is very well documented, and, although the team had not received any specific recommendation for MySQL, group members have had experience via Web Programming, and personal projects.

MySQL as used with both XAMPP (in development environments), and directly via the *mysql-server* package in Ubuntu (on the production server and in development environments).

Apache

Version 2.4.29

Documentation	Support	Recommendations	Prior knowledge
Very good	Very good	None	Very good

Apache 2.4.29 was released on the 23rd of October 2017. The web server is very well documented, and, although the team had not received any specific recommendation for Apache, group members have had experience via Web Programming, Web Servers and Web Technologies, and personal projects.

Apache was used both with XAMPP (in development environments), and directly via the *apache2* package in Ubuntu (on the production server and in development environments).

Ubuntu

Version 17.04

Documentation	Support	Recommendations	Prior knowledge
Very good	Very good	Good	Very good

Ubuntu 17.04 was released on the 13th of April 2017. The operating system is extremely well documented, the team had received positive recommendations, and group members have had experience via Unix Essentials for System Administrators, Unix Systems Administration and Programming, Building IT Systems, Web Servers and Web Technology, and personal projects.

Ubuntu 17.04 was installed on the production server. See "Implementation" under 6. Implementation Instructions and "Vultr" under this section for more information on the production server.

XAMPP

Version 7.1.10

Documentation	Support	Recommendations	Prior knowledge
Good	Good	None	Good

XAMPP 7.1.10 was released on the 19th of October, 2017. The software is fairly well documented, and, although the team had not received any specific recommendation for XAMPP, group members have had prior experience via Web Programming. XAMPP is also a reasonable, and well-used, cross-platform web development environment solution.

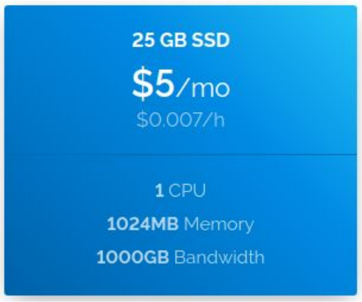
XAMPP was used as a development environment tool on the Windows operating system. It is packaged with Apache, PHP, and MySQL.

Vultr

Documentation	Support	Recommendations	Prior knowledge
Good	Good	None	Good

Vultr is an SSD-based VPS provider, with servers in Sydney, Australia. The team deployed a 25GB SSD server, costing $5 USD a month - which is referred to as our *production* server.

Read more about the production server configuration in "Implementation" under 6. Implementation Instructions.


25 GB SSD
$5/mo
$0.007/h
1 CPU
1024MB Memory
1000GB Bandwidth

GitHub

https://github.com/jobsaustralia

The team was required to use GitHub. It is a useful tool for collaboration, version control, and code management.

Git 2.14.3, specifically, was released on the 23rd of October 2017. Git is a very well known, and respected, version control system.

Git was used both locally in development environments, and directly via the *git* package on Ubuntu on the production server and in development environments.

Some team members used GitHub Desktop (version 1.0.3) as a front-end for Git with GitHub.

The team has published three repositories in the project's GitHub organisation. The following table details the purpose of each repository.

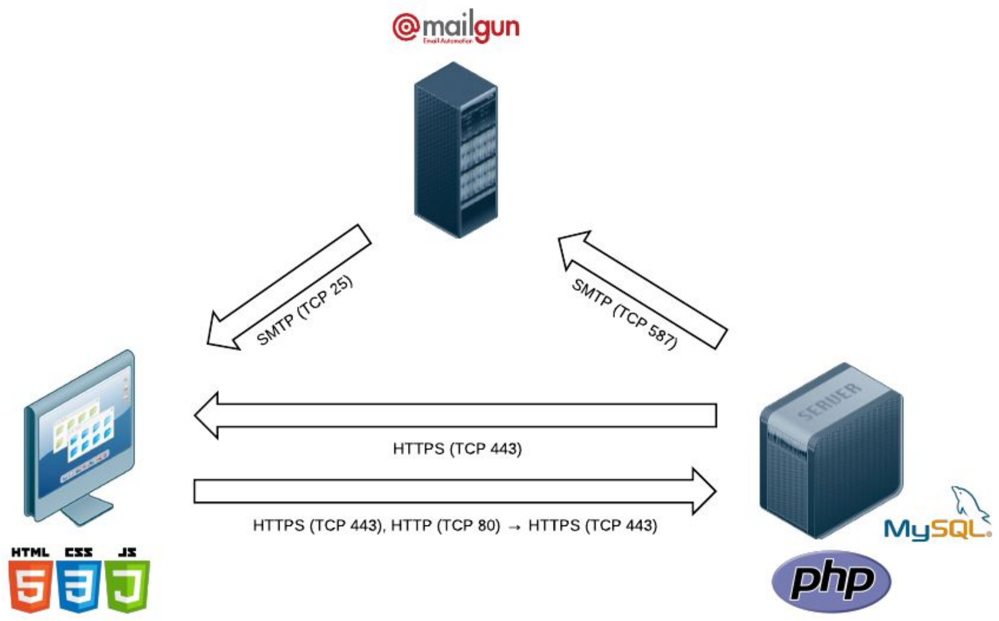
Repository name	Purpose
jobsaustralia/jobsaustralia.tech	Sub-project for job seekers.
jobsaustralia/employ.jobsaustralia.tech	Sub-project for employers.
jobsaustralia/scripts-conf-and-docs	Scripts (deployment, maintenance, and configuration), configuration files, and documentation.

As mentioned in "Laravel" of this section, the team maintains two official branches. The *master* branch is for work on Laravel 5.4, and the *laravel-5.5* branch is for work on Laravel 5.5. These branches are present in both sub-project repositories. All other branches are, and were, for feature introduction, or experimentation.

Further technologies used in development and production include those in the following table.

Name	Version	Description
Bootstrap	3.3.6	Front-end component library.
Composer	1.5.2	Dependency manager for PHP.
CSS	3	Style sheet language.
FontAwesome	4.7.0	Package of iconic fonts.
GitHub API	3	Public REST API to integrate with GitHub.
HTML	5.1	Markup language for documents.
JQuery	3.2.1	JavaScript library, that we primarily used to download JSON data.
Notepad++	7.5.1	Text editor for Windows operating systems.
PHPUnit	6.0	Unit testing framework for PHP.
Sublime Text	3.0	Text editor.
IcoMoon	N/A	Tool to create iconic fonts from vector graphics.
Let's Encrypt	N/A	Free and automated certificate authority.
Mailgun	N/A	Service for sending outgoing mail.

3 Overall Architecture



The system interacts with outside parties over various protocols, and for various purposes.

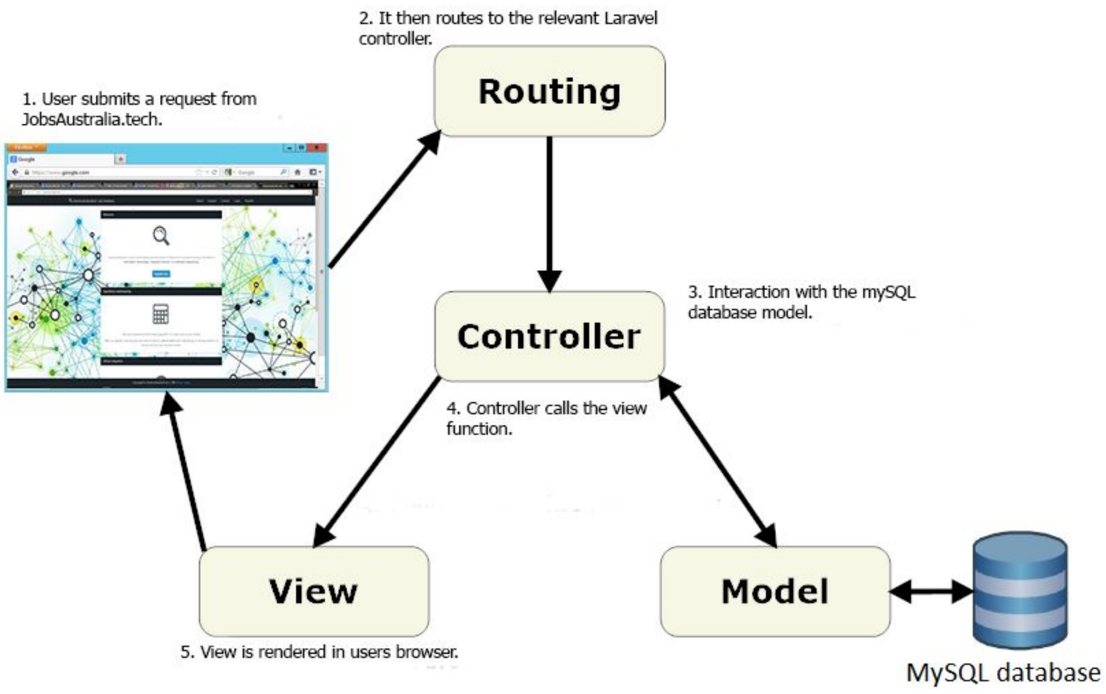
The client interacts with the server over HTTPS (TCP port 443), and, only initially, via HTTP (TCP port 80) - because of HTTP to HTTPS redirection, and HTTP Strict Transport Security (HSTS) (See "Connection security" under Security under 7. Non-functional requirements for more information on HSTS).

On the client-side, HTML5, CSS3, and JavaScript are processed and run by the browser. This includes the processing of retrived JSON data for matchmaking (See "4.1.1 Perform matchmaking on jobs to job seekers" and "4.1.2 Perform matchmaking on employers to applicants" for detailed information on matchmaking).

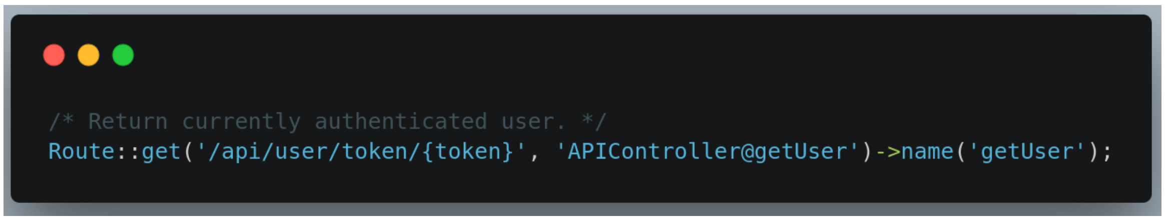
On the server-side, PHP (including Laravel) processes PHP scripts (See "4. System Architecture" for more information on the architecture of Laravel), and MySQL performs database management operations.

Mailgun is used to send outgoing mail (SMTP - TCP port 587) from the server to a client's email address (SMTP - typically over TCP port 25). This process is secure in transport.

4 System Architecture


2. It then routes to the relevant Laravel controller.
Routing
1. User submits a request from JobsAustralia.tech.
Controller
3. Interaction with the mySQL database model.
4. Controller calls the view function.
View
Model
5. View is rendered in users browser.
MySQL database

Laravel segments its main functionality into *routes*, *controllers*, *views*, and *models*.



```
/* Return currently authenticated user. */
Route::get('/api/user/token/{token}', 'APIController@getUser')->name('getUser');
```

Routes are used to register the GET and POST resources that combine to form the system. In them, the method (GET or POST), the resource (with any parameters), the controller, and the function are specified. When a user navigates to a resource registered in a route, the registered function in the registered controller is called - with any data (from GET parameters, or received via POST).

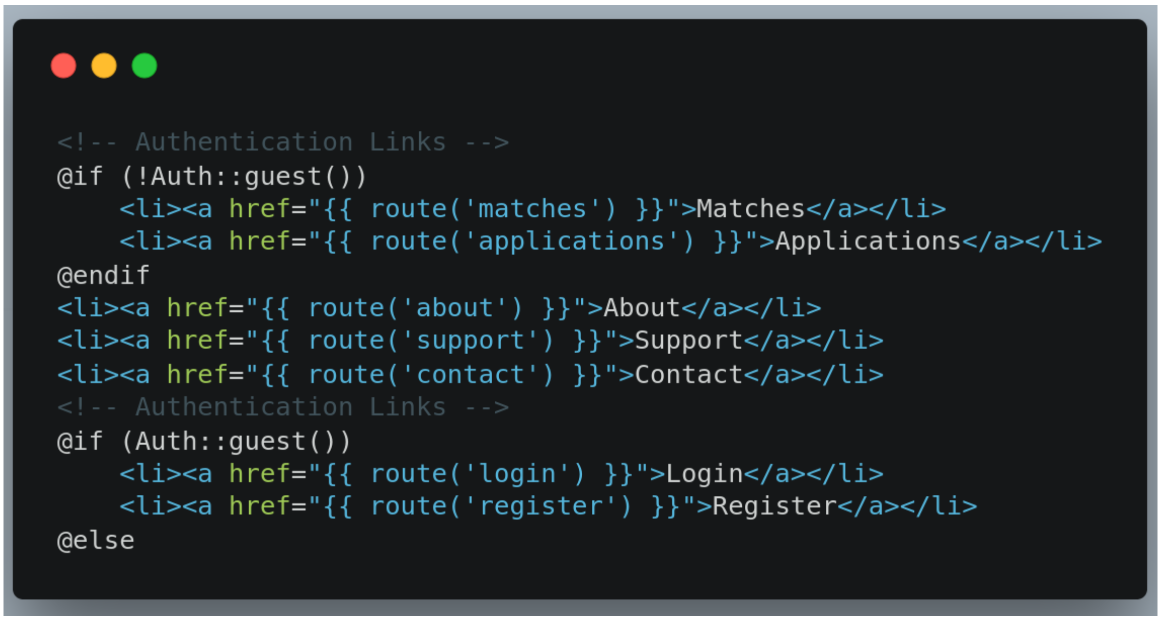
Controllers are classes containing functions. They are used for function segmentation, coding standards, and other features (like use of middlewares). The controllers used in the project, and their description, include the following.

Controller name	Functionality
ForgotPasswordController	Handling of forgotten passwords. This function uses the *guest* middleware to ensure a user must be logged out before accessing its functionality.
LoginController	User authentication. This function uses the *guest* middleware to ensure a user must be logged out before accessing its functionality.
RegisterController	User registration, and authentication. This function uses the *guest* middleware to ensure a user must be logged out before accessing its functionality.
ResetPasswordController	Handling of password resets. This function uses the *guest* middleware to ensure a user must be logged out before accessing its functionality.
APIController	A collection of API functions. This function uses the *auth* middleware to ensure a user must be logged in before accessing its functionality.
ApplicationController	Handling of job application creation, editing, deletion, and viewing. This function uses the *auth* middleware to ensure a user must be logged in before accessing its functionality.
ContactController	Handling of mailing team members contact form submissions.
JobController	Handling of job creation, editing, deletion, and viewing. This function uses the *auth* middleware to ensure a user must be logged in before accessing its functionality.
ProfileController	Handling of user editing, deletion, and viewing. This function uses the *auth* middleware to ensure a user must be logged in before accessing its functionality.
ResumeController	Handling of resume uploading, deletion, and viewing. This function uses the *auth* middleware to ensure a user must be logged in before accessing its functionality.

Controllers are stored in the *app/Http/Controllers* directory of the sub-project repositories.

A model a similar to an object in most programming languages. Models have attributes (see “Database table schemas” under 11. Appendix for all attributes), and an associated table. The models used in the project, and their description, include the following.

Model name	Description
JobSeeker (also User in jobsaustralia.tech)	User who is registered with jobsaustralia.tech A user who is seeking a job.
Employer (also User in employ.jobsaustralia.tech)	User who is registered with employ.jobsaustralia.tech A user who is advertising a job.
Application	A submitted application to a job.
Job	A posted job.



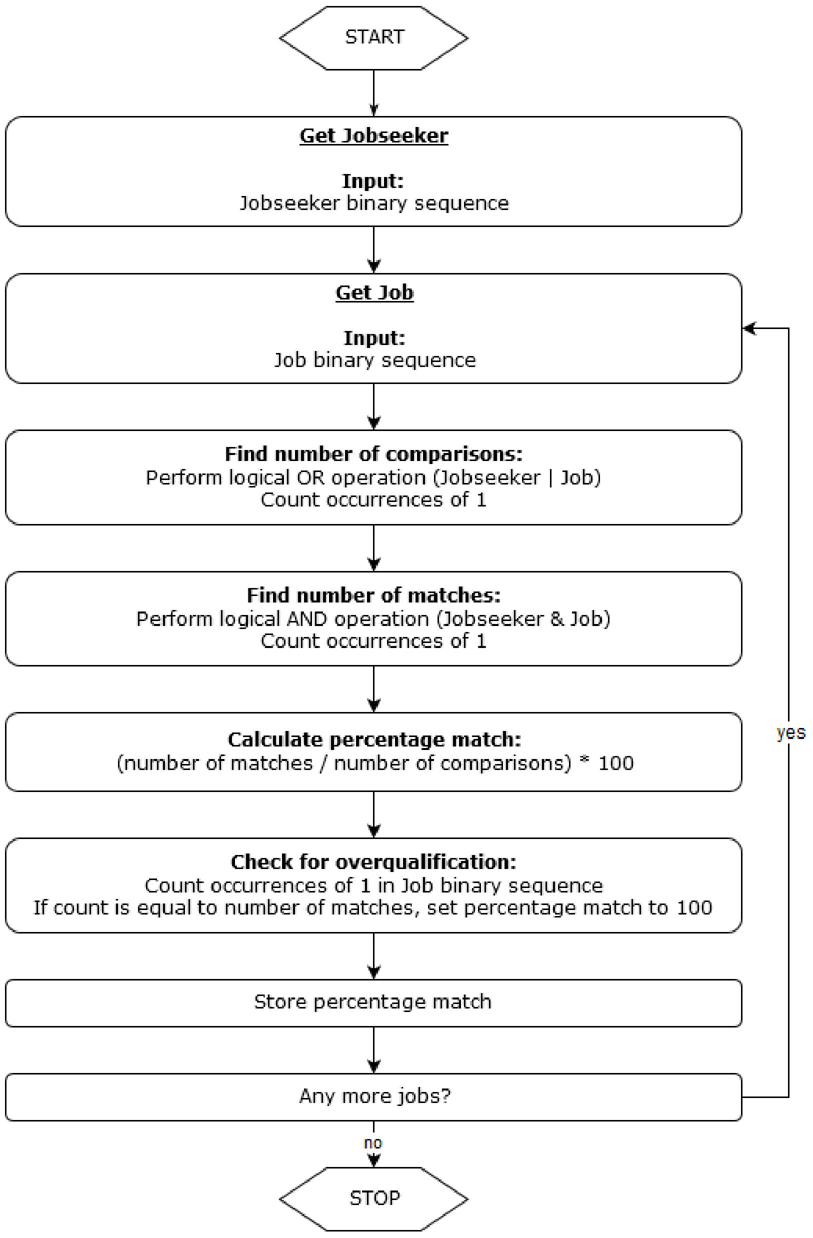
```
<!-- Authentication Links -->
@if (!Auth::guest())
    <li><a href="{{ route('matches') }}">Matches</a></li>
    <li><a href="{{ route('applications') }}">Applications</a></li>
@endif
<li><a href="{{ route('about') }}">About</a></li>
<li><a href="{{ route('support') }}">Support</a></li>
<li><a href="{{ route('contact') }}">Contact</a></li>
<!-- Authentication Links -->
@if (Auth::guest())
    <li><a href="{{ route('login') }}">Login</a></li>
    <li><a href="{{ route('register') }}">Register</a></li>
@else
```

A view is an HTML document, that can contain PHP variables and limited logic (for example, conditionals and loops) to alter and build the document before it is sent to the browser.

4.1 FUNCTIONALITIES/FEATURES

This section details the individual functionalities of the project.

4.1.1 Perform matchmaking on jobs to job seekers.


START
Get Jobseeker
Input:
Jobseeker binary sequence
Get Job
Input:
Job binary sequence
Find number of comparisons:
Perform logical OR operation (Jobseeker | Job)
Count occurrences of 1
Find number of matches:
Perform logical AND operation (Jobseeker & Job)
Count occurrences of 1
Calculate percentage match:
(number of matches / number of comparisons) * 100
Check for overqualification:
Count occurrences of 1 in Job binary sequence
If count is equal to number of matches, set percentage match to 100
Store percentage match
Any more jobs?
yes
no
STOP

To minimise execution time for the comparison of a large number of variables (which in the current implementation is over 50), the skills of the jobseeker/job are represented as a binary sequence where 1 corresponds to a possessed/required skill, and 0 if not. In doing this, finding the number of matches can be accomplished by simply using bitwise logical operations instead of a series of manual counting by iterating through each job for each variable.

It was found that an exact sequence match was not ideal for the purpose of the website, as 0 to 0 matches would artificially inflate the resulting percentage match. To remove this effect, a logical OR operation is performed on the two binary sequences; the resulting binary sequence would reflect where 0 to 0 matches occur with a 0, and 1 otherwise. By counting the number of 1s in this sequence, an updated number of comparisons (that is, the total number of skills minus the number of skills which both jobseeker and job lack) is obtained.

A logical AND operation is performed on the two binary sequences to find 1 to 1 matches; the resulting binary sequence would reflect these with a 1, and 0 otherwise. By counting the number of 1s in this sequence, the number of skills that the jobseeker possesses that the job requires is obtained. This value, along with the one obtained from the logical OR operation, is used to calculate the percentage match.

The algorithm was later amended to account for instances in which a jobseeker may be overqualified for a job, that is, they possess additional skills to that of all those desired by a job. Without the amendment, the resulting percentage match would be lower due to the occurrences of 1 to 0 between the jobseeker and job binary sequences. It was decided then to adjust the percentage match to 100 by default if the jobseeker at the very least possessed all the skills required for the given job.

In testing, it was found that JavaScript does not handle bitwise operations on sequences exceeding 32 bits. The maximum length of bit sequences handled in the current project implementation is 51 (the number of skills).

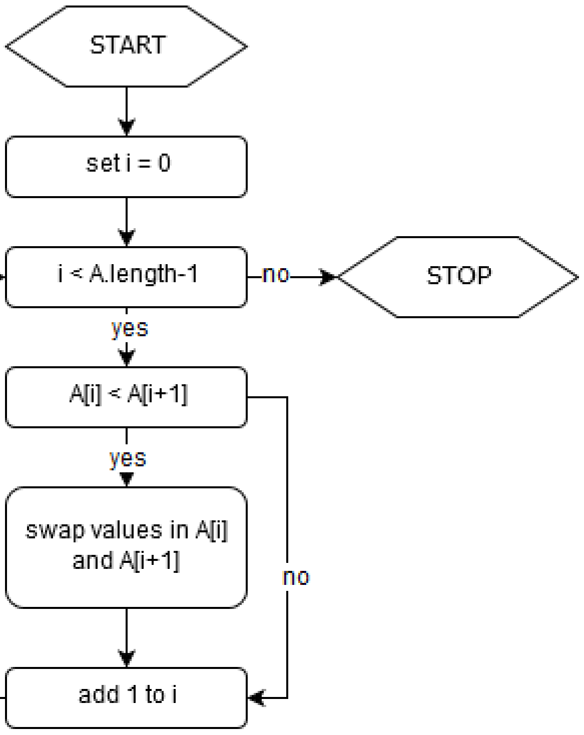
$$u = 011011010110010101101100011010010111001101110011000$$
$$j = 011000010110000101110010011011110110111001101000000$$

$$u[0] = 01101101011001010110110001101001$$
$$u[1] = 0111001101110011000$$
$$j[0] = 01100001011000010111001001101111$$
$$j[1] = 0110111001101000000$$

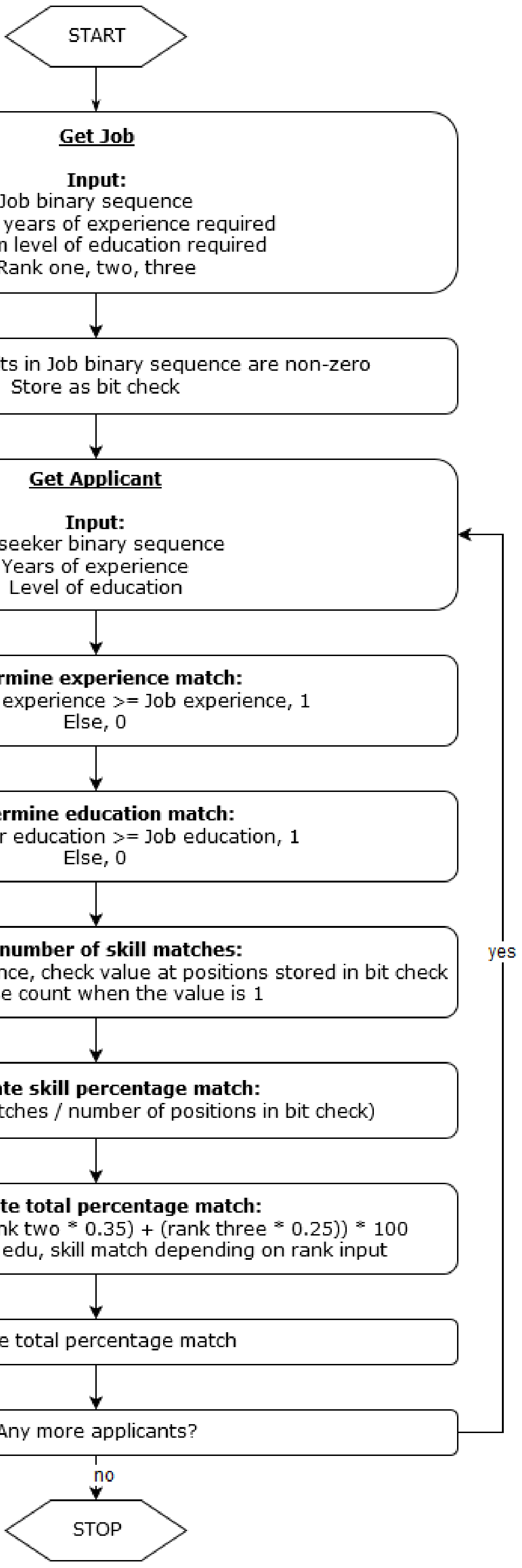
$$c = concat(u[0] \mid j[0],\ u[1] \mid j[1])$$
$$m = concat(u[0] \,\&\, j[0],\ u[1] \,\&\, j[1])$$

The limitation was resolved by dissociating the bit sequences into 32-bit (maximum) chunks, performing bitwise operations on the chunks, and concatenating the result.

To sort the jobs by order of greatest to lowest percentage match, the bubble sort algorithm was chosen. While not the most ideal of sorting algorithms in terms of run time, its simplicity allows for pseudo key-value mappings to be preserved.



4.1.2 Perform matchmaking on employers to applicants.

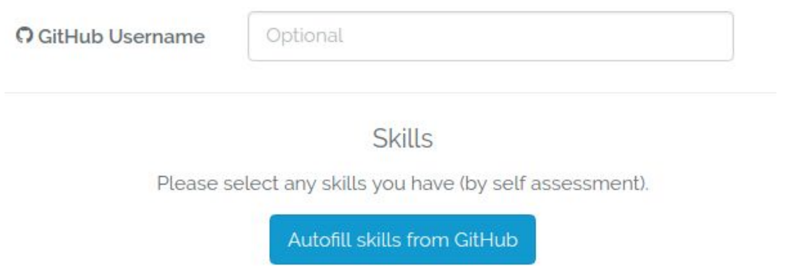

START
Get Job
Input:
Job binary sequence
Minimum years of experience required
Minimum level of education required
Rank one, two, three
Determine which bits in Job binary sequence are non-zero
Store as bit check
Get Applicant
Input:
Jobseeker binary sequence
Years of experience
Level of education
Determine experience match:
If Jobseeker experience >= Job experience, 1
Else, 0
Determine education match:
If Jobseeker education >= Job education, 1
Else, 0
Find number of skill matches:
In Jobseeker binary sequence, check value at positions stored in bit check
Increase count when the value is 1
Calculate skill percentage match:
(number of skill matches / number of positions in bit check)
Calculate total percentage match:
((rank one * 0.4) + (rank two * 0.35) + (rank three * 0.25)) * 100
Substitute with exp, edu, skill match depending on rank input
Store total percentage match
Any more applicants?
yes
no
STOP

The skill matching on the employer side differs from the jobseeker side in that it only cares about the number of skills an applicant has within those that the job requires. Instead of using bitwise logical operations to minimise execution time, this algorithm notes which bits in the job's binary sequence is 1 (corresponding to which skills it requires) and then checks only those positions in the applicant's binary sequence. The probably of worst case, in which a job would require all 50+ listed skills and thereby cause the algorithm to iterate through every skill of every applicant, is low.

The matchmaking algorithm for employers takes into account additional parameters; for each job the employer is asked to specify a minimum level of education required, minimum number of years of experience, and to rank what is most and least important to them between education, experience and skills. Depending on the determined ranking order, weights are attached to the values obtained from matching education, experience and skills, and the total percentage match is calculated.

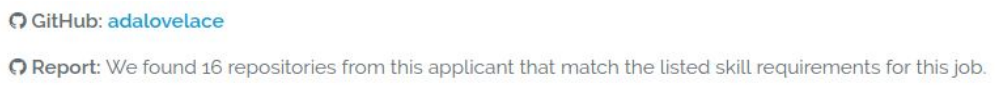
Bubble sorting is used to order the percentage matches.

4.1.3 Integrate with GitHub


GitHub Username
Optional
Skills
Please select any skills you have (by self assessment).
Autofill skills from GitHub

On job seeker registration (or editing a profile), a GitHub username is requested. From this field, and on user prompt, the public GitHub API is queried to return a list of the user's public repositories in JSON format. The list of skills on the registration form is then automatically filled based on the majority languages of the user's repositories - ignoring forks.

The GitHub field is optional.



GitHub: adalovelace

Report: We found 16 repositories from this applicant that match the listed skill requirements for this job.

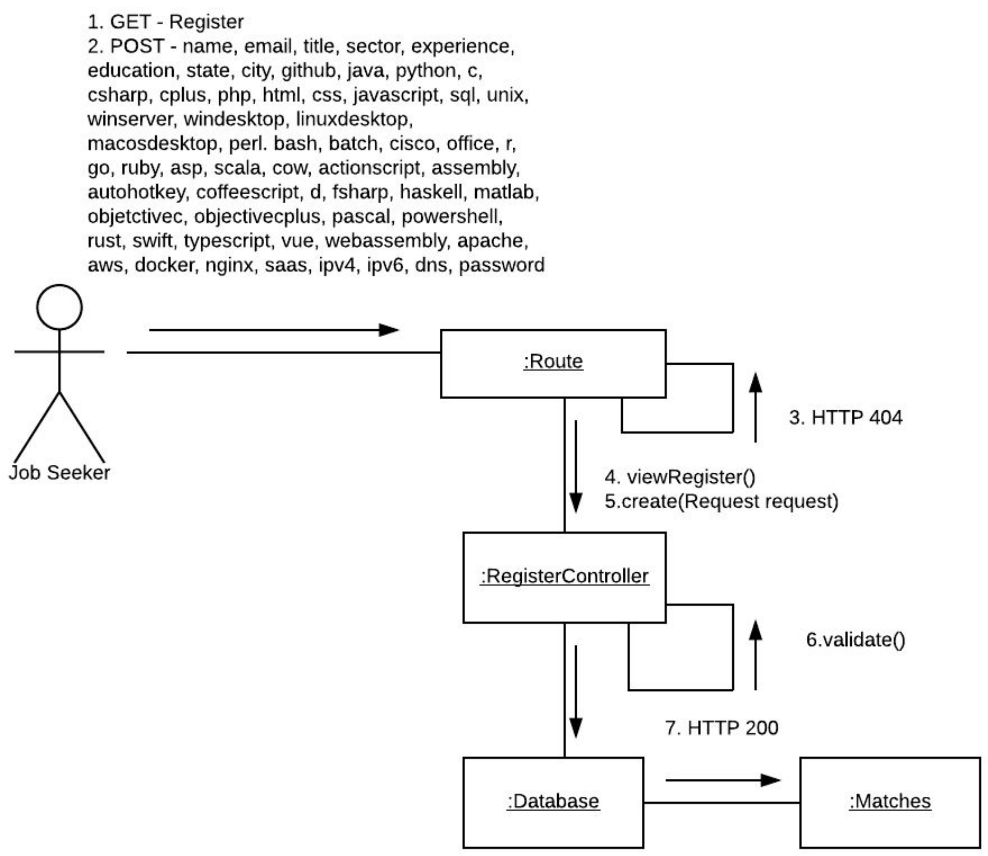
On the viewing of an application by an employer, if the applicant has stored a GitHub username, the public GitHub API is queried to return a list of the job seekers public repositories in JSON format. The number of repositories - ignoring forks - with majority languages that match required skills for the job are counted. The count is displayed on the application view, with a link to the job seekers GitHub profile.

On Github, forks are copies of repositories and do not axiomatically indicate experience or knowledge in the majority language of the repository forked.

4.1.4 Permit users to register.

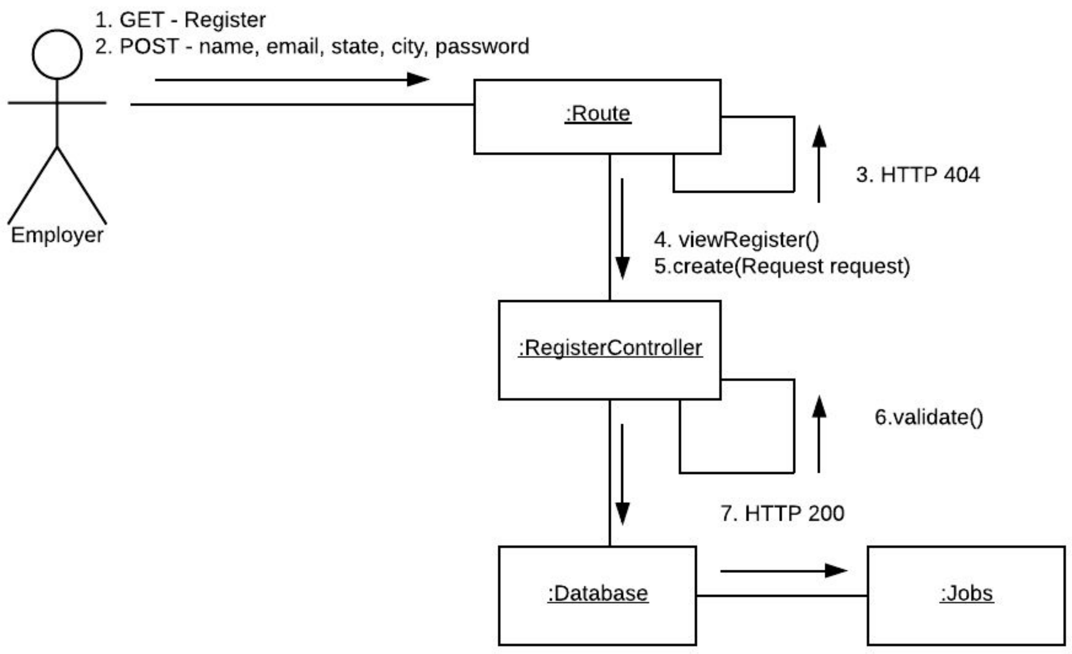
This diagram explains the process by which a job seeker registers to the sub-project job seekers website.

The job seeker registers by creating a profile with their personal information (including skills, education, and experience). The data is validated before being stored in the database. The job seeker is sent to the matching jobs view after successful registration.


1. GET - Register
2. POST - name, email, title, sector, experience, education, state, city, github, java, python, c, csharp, cplus, php, html, css, javascript, sql, unix, winserver, windesktop, linuxdesktop, macosdesktop, perl. bash, batch, cisco, office, r, go, ruby, asp, scala, cow, actionscript, assembly, autohotkey, coffeescript, d, fsharp, haskell, matlab, objetctivec, objectivecplus, pascal, powershell, rust, swift, typescript, vue, webassembly, apache, aws, docker, nginx, saas, ipv4, ipv6, dns, password
Job Seeker
:Route
3. HTTP 404
4. viewRegister()
5.create(Request request)
:RegisterController
6.validate()
7. HTTP 200
:Database
:Matches

This diagram explains the process by which an employer registers to the sub-project employers website.

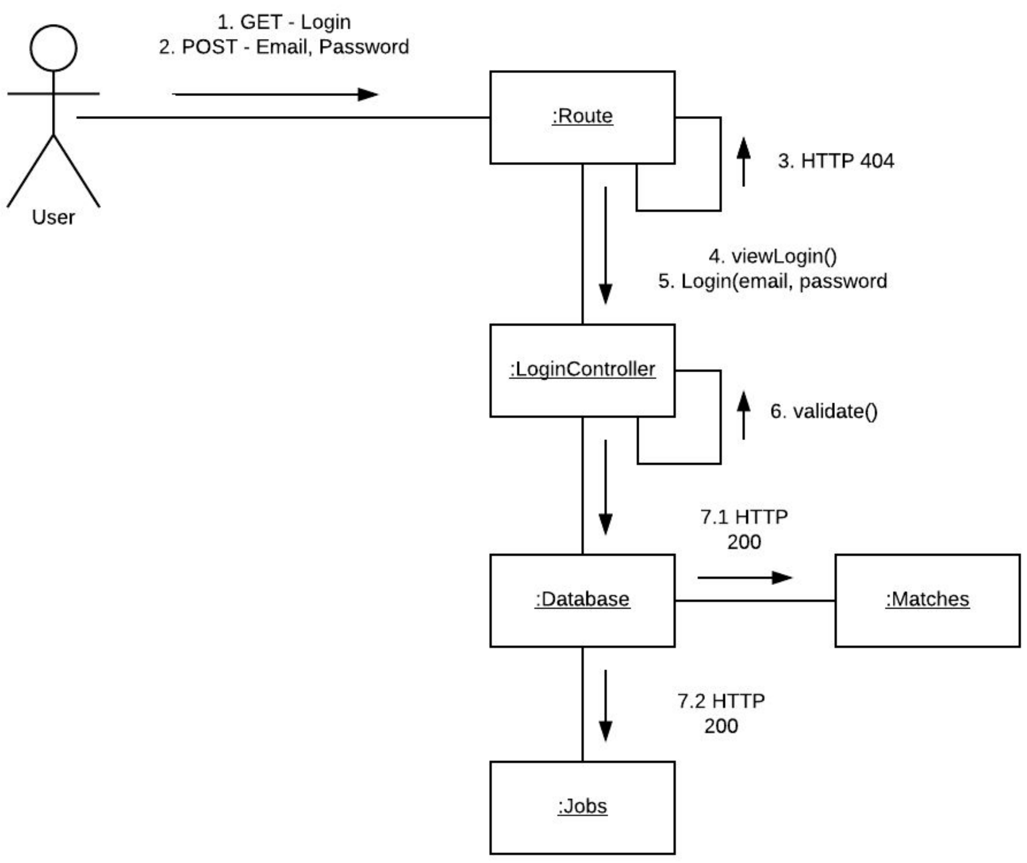
The employer registers by creating a profile with their personal information. The information is validated before being stored in the database. The employer is sent to the jobs view where they can manage their posted jobs after successful registration.



4.1.5 *Permit users to login.*

This diagram explains the process by which a user (both job seeker and employer) authenticates with each sub-project website.

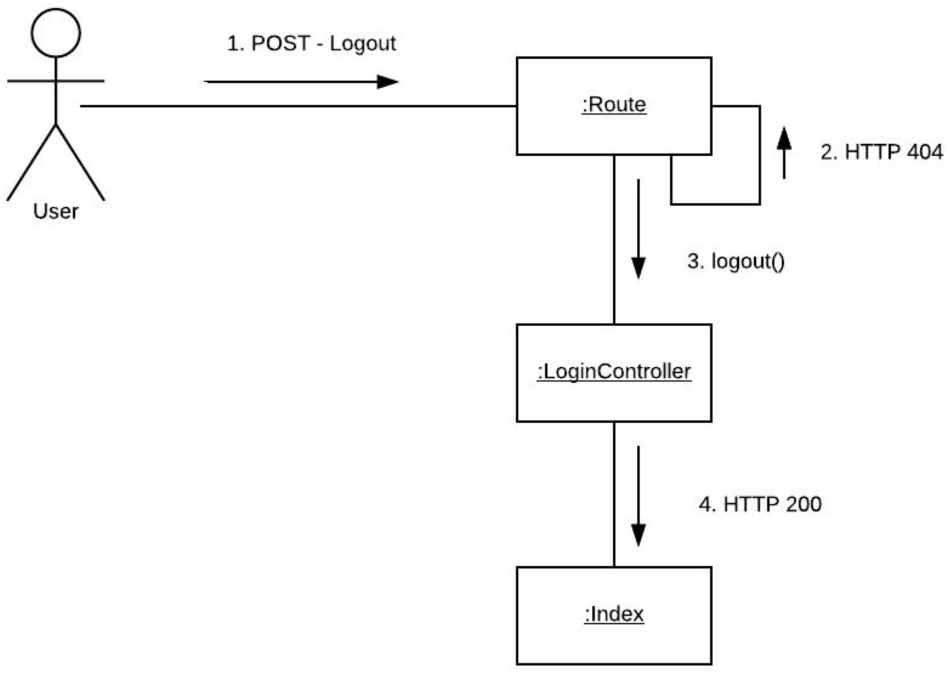
On successful login, the job seeker is sent to the matches view. The employer is sent to the jobs view.



4.1.6 *Permit users to logout.*

This diagram explains the process by which a user (both job seeker and employer) de-authenticates with each sub-project website.

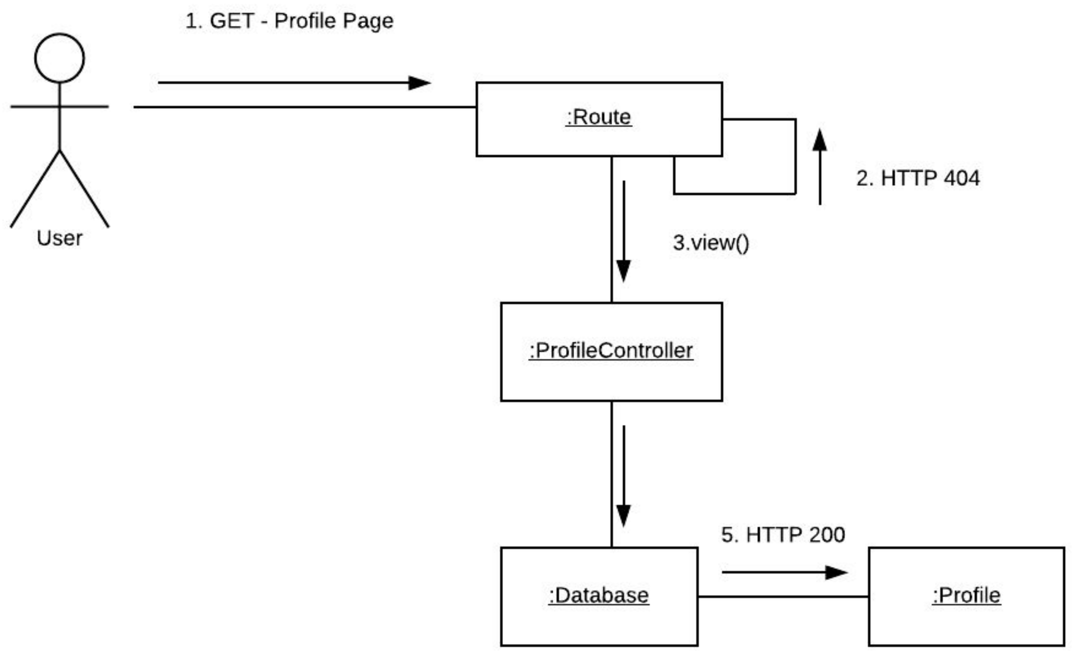
On successful logout, the user is sent to the index view (the homepage).



4.1.7 Permit users to view their own profile.

This diagram explains the process by which a user views their own profile.

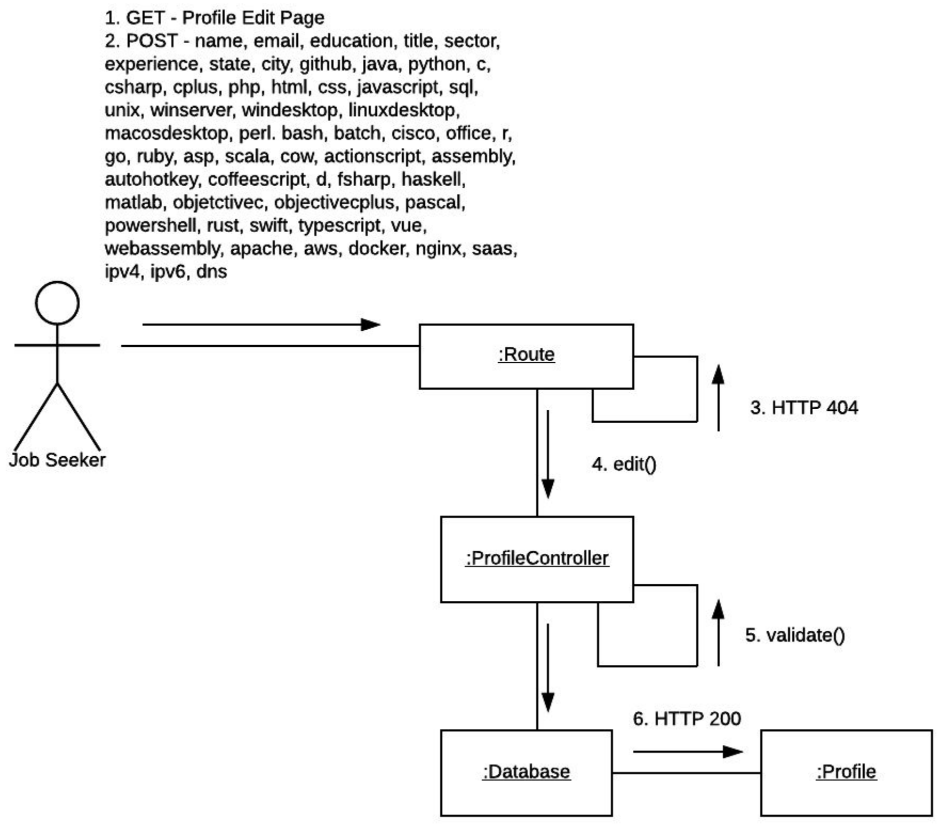
The user navigates to the profile resource. The profile controller retrieves the user's data from the database, displaying it in the profile view.



4.1.8 Permit users to edit profile.

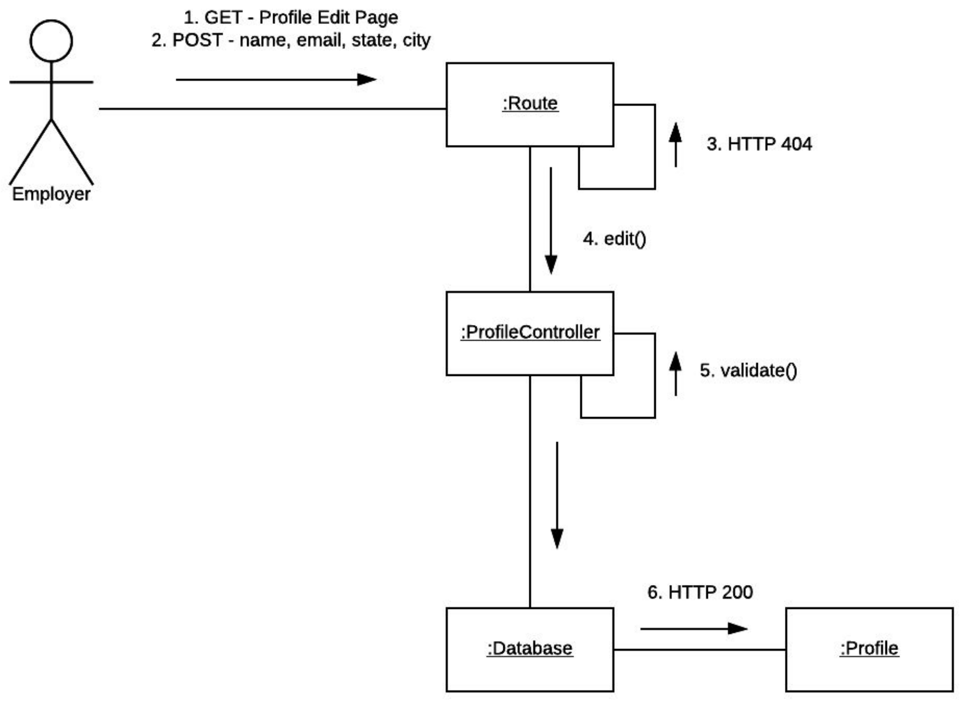
This diagram explains the process by which a job seeker edits their profile.

If the edited fields pass validation, the job seeker's details are updated in the database, and they are sent to the profile view.


1. GET - Profile Edit Page
2. POST - name, email, education, title, sector, experience, state, city, github, java, python, c, csharp, cplus, php, html, css, javascript, sql, unix, winserver, windesktop, linuxdesktop, macosdesktop, perl. bash, batch, cisco, office, r, go, ruby, asp, scala, cow, actionscript, assembly, autohotkey, coffeescript, d, fsharp, haskell, matlab, objetctivec, objectivecplus, pascal, powershell, rust, swift, typescript, vue, webassembly, apache, aws, docker, nginx, saas, ipv4, ipv6, dns
Job Seeker
:Route
3. HTTP 404
4. edit()
:ProfileController
5. validate()
6. HTTP 200
:Database
:Profile

This diagram explains the process by which an employer edits their profile.

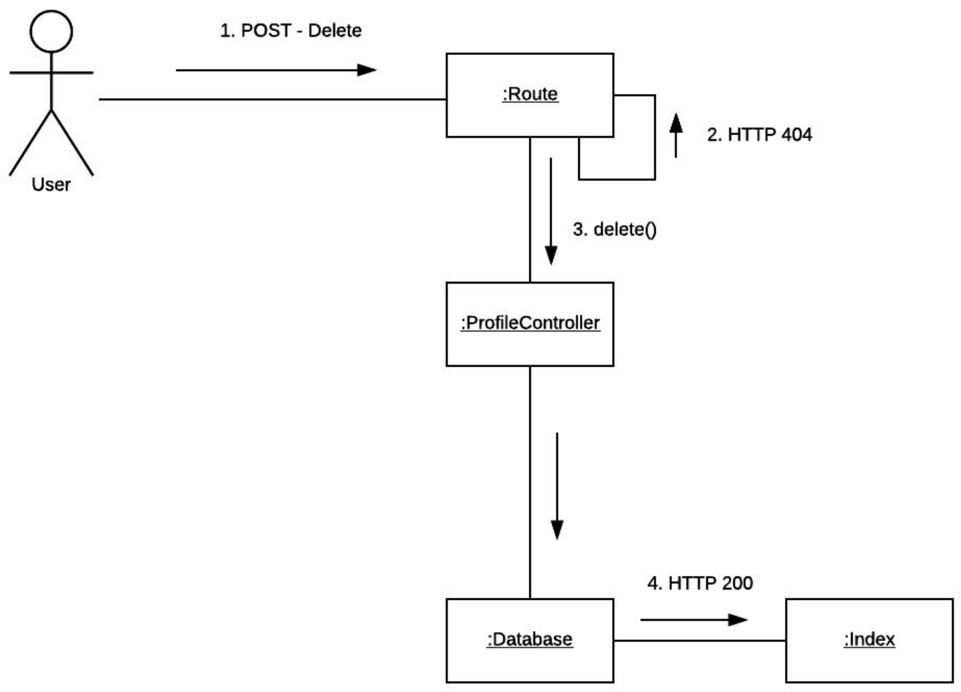
If the edited fields pass validation, the employer's details are updated in the database, and they are sent to the profile view.



4.1.9 *Permit users to delete account.*

This diagram explains the process by which a user deletes their account.

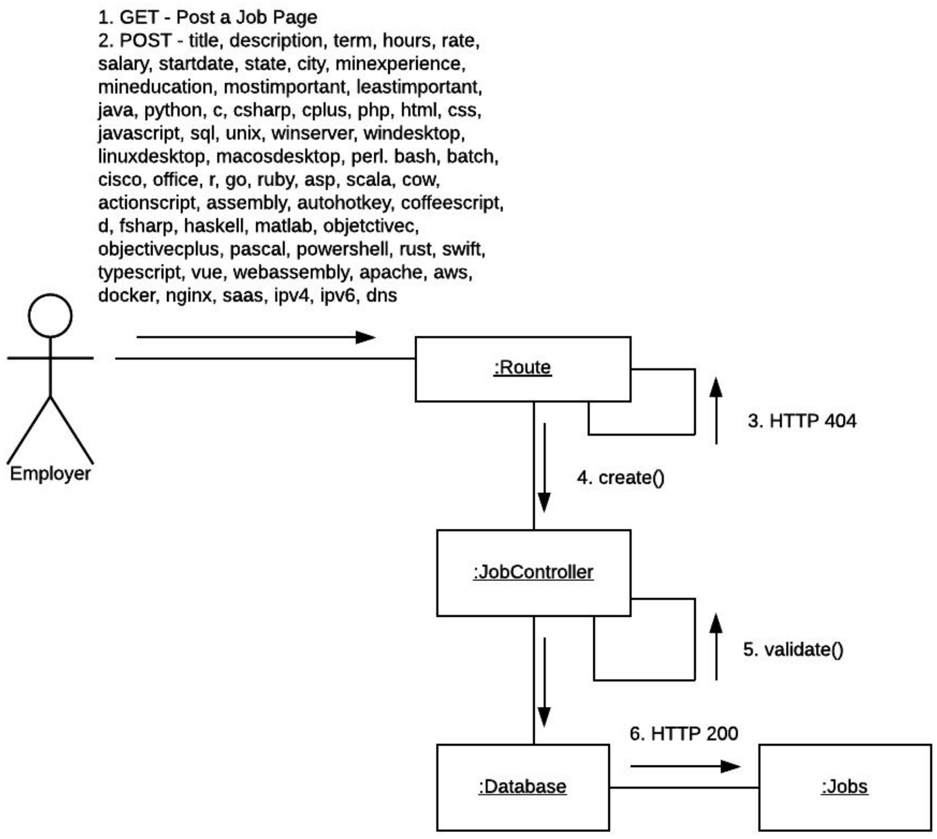
The currently authenticated user is deleted (via the destroy method in Laravel), de-authenticated, and sent to the index view (the homepage).



4.1.10 Permit employers to post a job.

This diagram explains the process by which an employer posts a job.

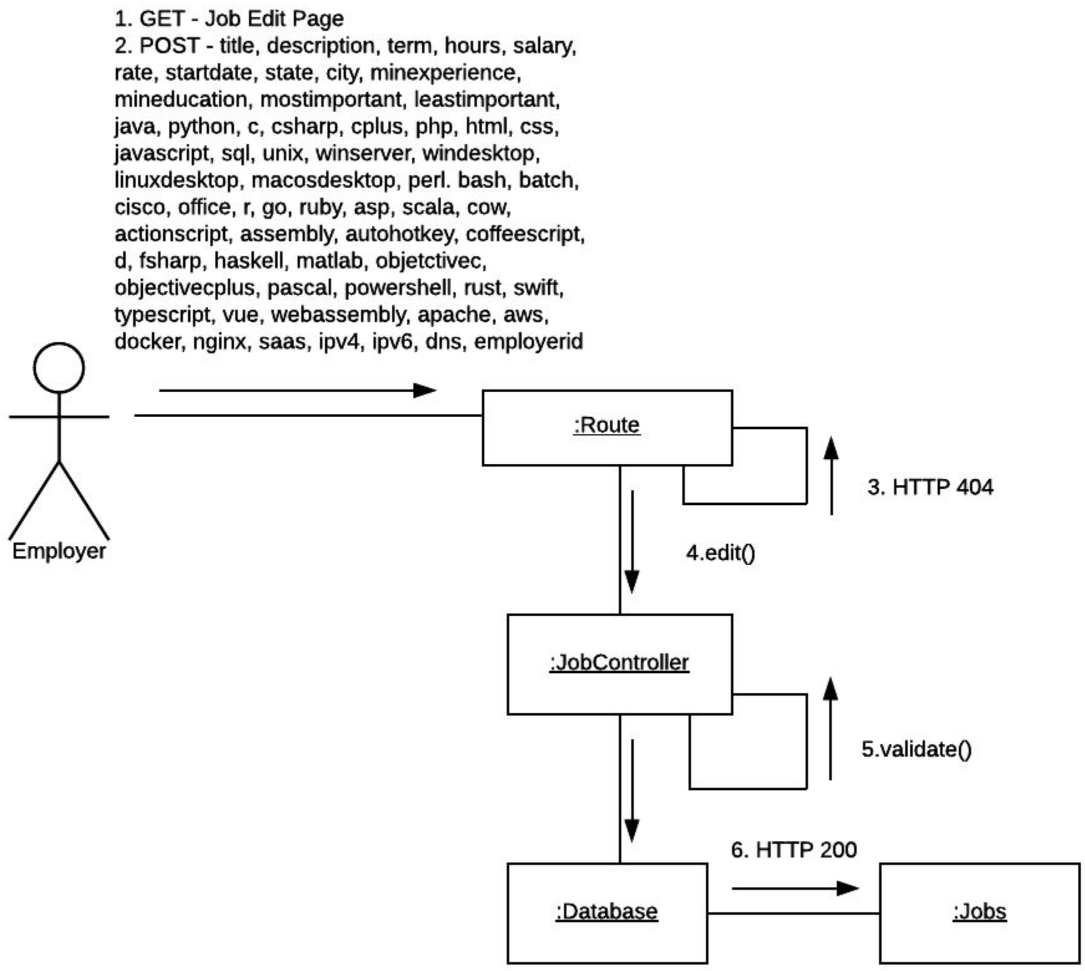
After posting a job, the employer is sent to the jobs view where they can manage their posted jobs.


1. GET - Post a Job Page
2. POST - title, description, term, hours, rate, salary, startdate, state, city, minexperience, mineducation, mostimportant, leastimportant, java, python, c, csharp, cplus, php, html, css, javascript, sql, unix, winserver, windesktop, linuxdesktop, macosdesktop, perl. bash, batch, cisco, office, r, go, ruby, asp, scala, cow, actionscript, assembly, autohotkey, coffeescript, d, fsharp, haskell, matlab, objetctivec, objectivecplus, pascal, powershell, rust, swift, typescript, vue, webassembly, apache, aws, docker, nginx, saas, ipv4, ipv6, dns
Employer
:Route
3. HTTP 404
4. create()
:JobController
5. validate()
6. HTTP 200
:Database
:Jobs

4.1.11 Permit employers to edit a job.

This diagram explains the process by which an employer edits a posted job.

If the edited fields pass validation, the job's details are updated in the database, and the employer is sent back to the job view.

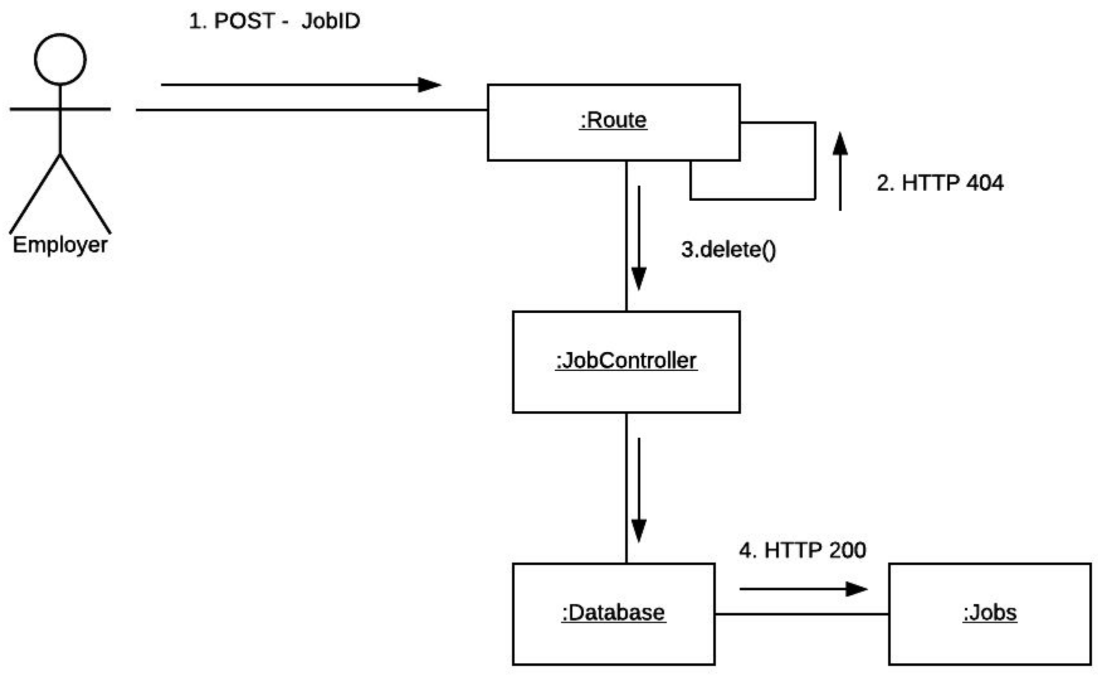

1. GET - Job Edit Page
2. POST - title, description, term, hours, salary, rate, startdate, state, city, minexperience, mineducation, mostimportant, leastimportant, java, python, c, csharp, cplus, php, html, css, javascript, sql, unix, winserver, windesktop, linuxdesktop, macosdesktop, perl. bash, batch, cisco, office, r, go, ruby, asp, scala, cow, actionscript, assembly, autohotkey, coffeescript, d, fsharp, haskell, matlab, objetctivec, objectivecplus, pascal, powershell, rust, swift, typescript, vue, webassembly, apache, aws, docker, nginx, saas, ipv4, ipv6, dns, employerid
Employer
:Route
3. HTTP 404
4.edit()
:JobController
5.validate()
6. HTTP 200
:Database
:Jobs

4.1.12 Permit employers to delete a job.

This diagram explains the process by which an employer deletes a posted job.

In the function, the job is verified as belonging to the employer who is currently authenticated.

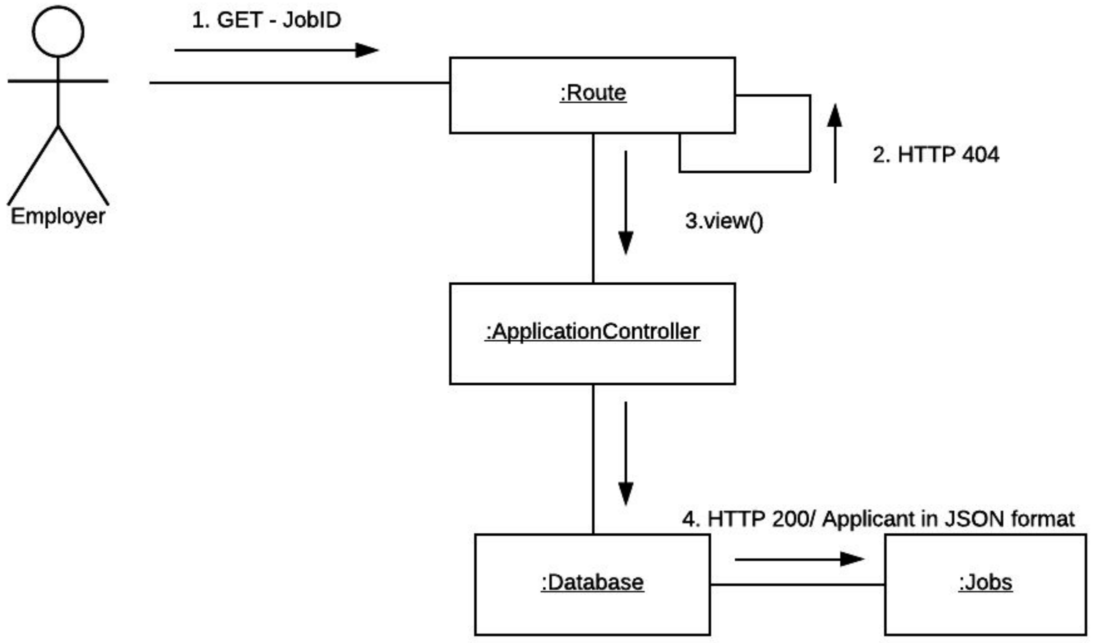
The job is deleted only if the employerID of the authenticated employer matches the employerID of the employer who posted the job.



4.1.13 Permit employers to view matching job applicants.

This diagram explains the process by which an employer views the list of matching job applicants to a specific job.

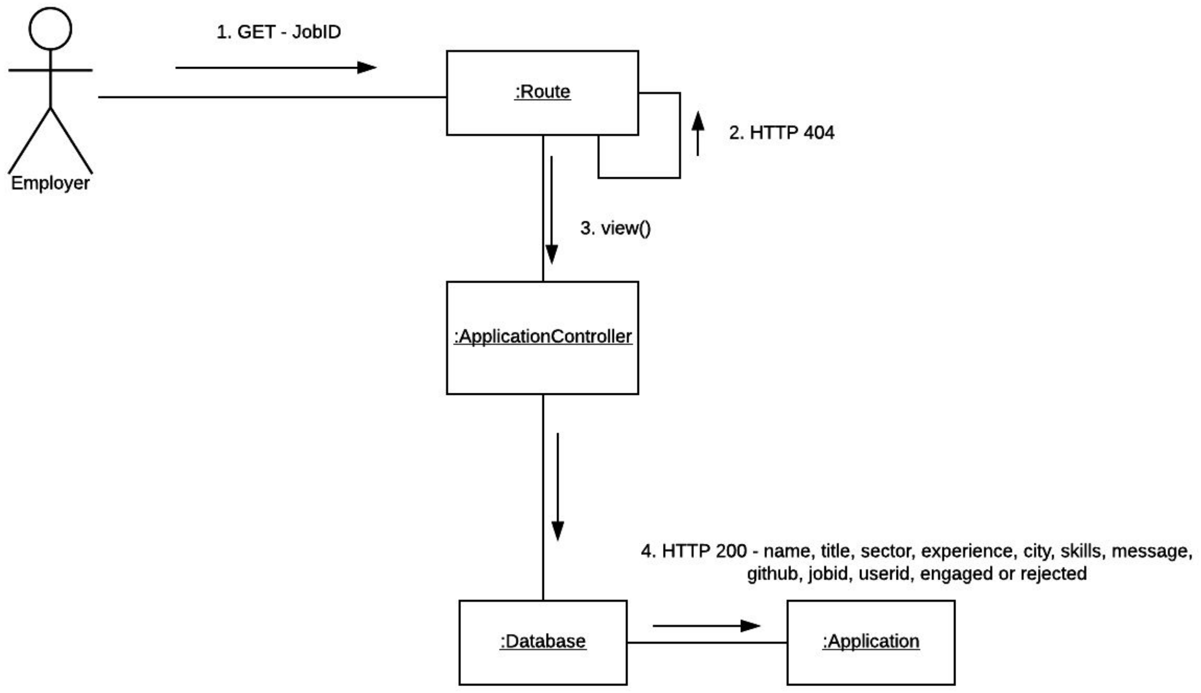
The employer sends the jobID as a GET parameter, and the database returns the applicants who applied for their posted job in JSON format for client-side matchmaking in JavaScript.



4.1.14 Permit employer to view a single job application.

This diagram explains the process by which an employer views a single job application.

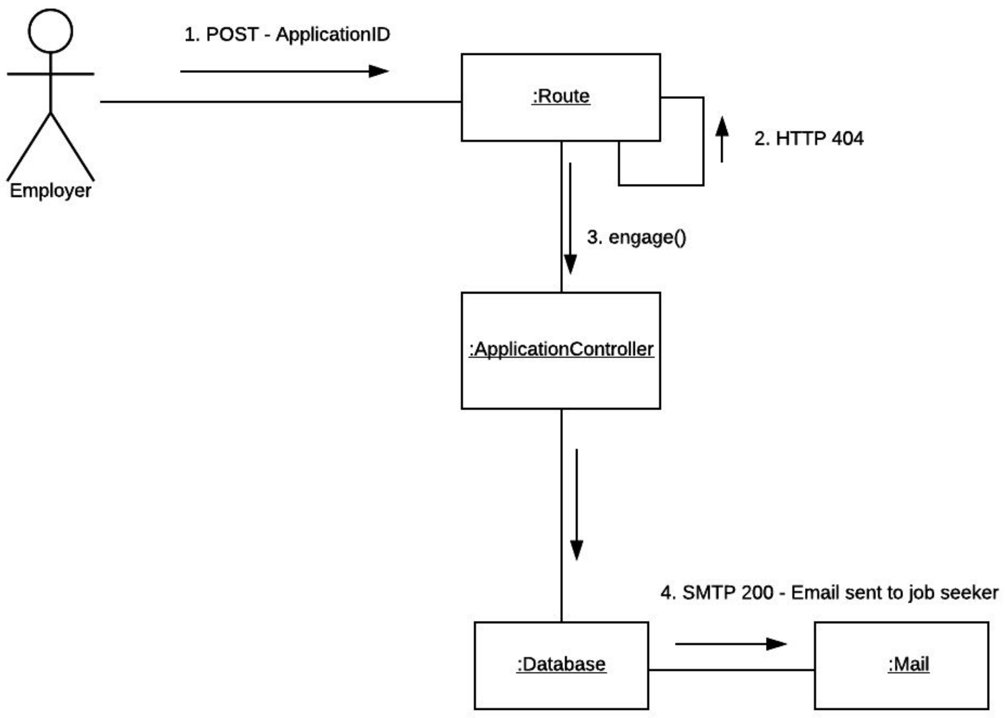
The employer sends the jobID as a GET parameter, and the database returns the data in the application view.


Employer
1. GET - JobID
:Route
2. HTTP 404
3. view()
:ApplicationController
4. HTTP 200 - name, title, sector, experience, city, skills, message, github, jobid, userid, engaged or rejected
:Database
:Application

4.1.15 Permit employers to discuss a job application with the applicant.

This diagram explains the process by which an employer initiates discussion on a job application with the applicant.

When the discuss button is pressed, an email is sent to the applicant containing the name of the job, the name of the employer, and the employers email address set as the Reply-To header.

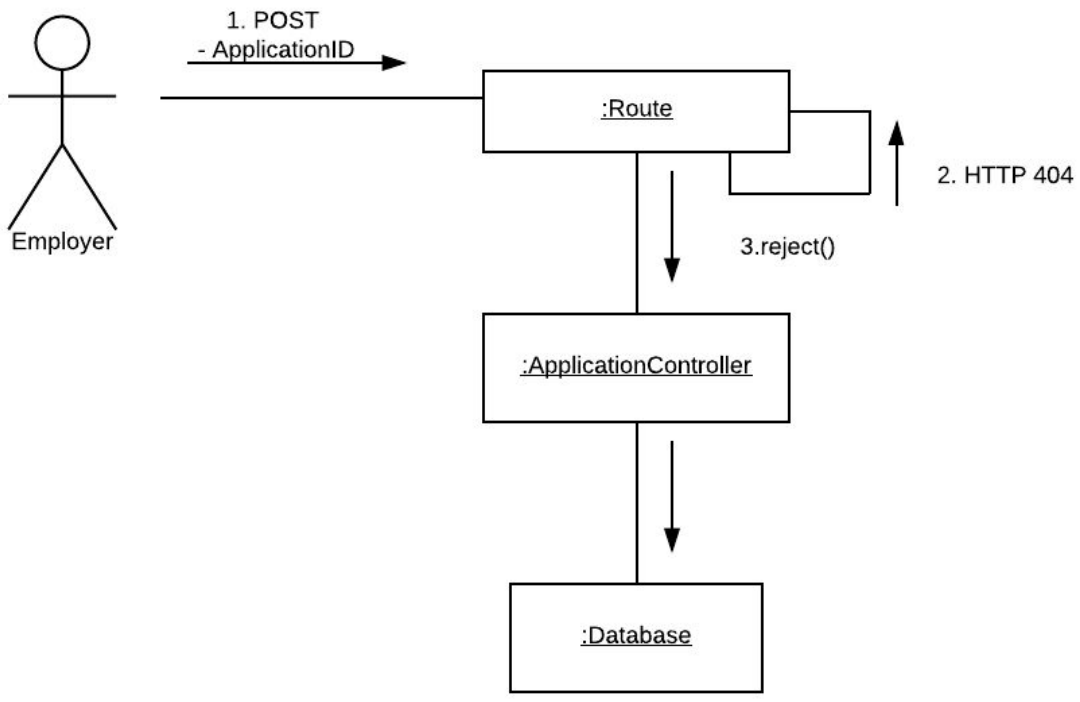


4.1.16 Permit employers to reject a job application.

This diagram explains the process by which an employer rejects a job application.

The employer POSTs the ApplicationID to the ApplicationController. The function verifies that the application is not either engage or rejected, and that the currently authenticated employer owns the job to which the application was for. Then we set the boolean rejected field to true.

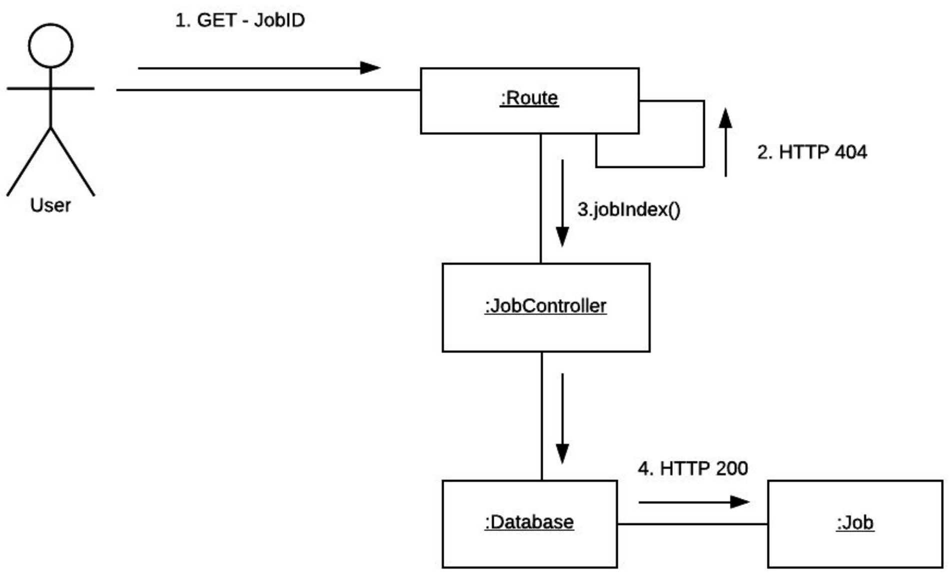
A job seeker cannot re-apply to a job once they have been rejected. Rejected applications are hidden from the employer after rejection.



4.1.17 Permit users to view a job.

This diagram explains the process by which users (both job seeker and employer) view a specific job via their respective sub-project website.

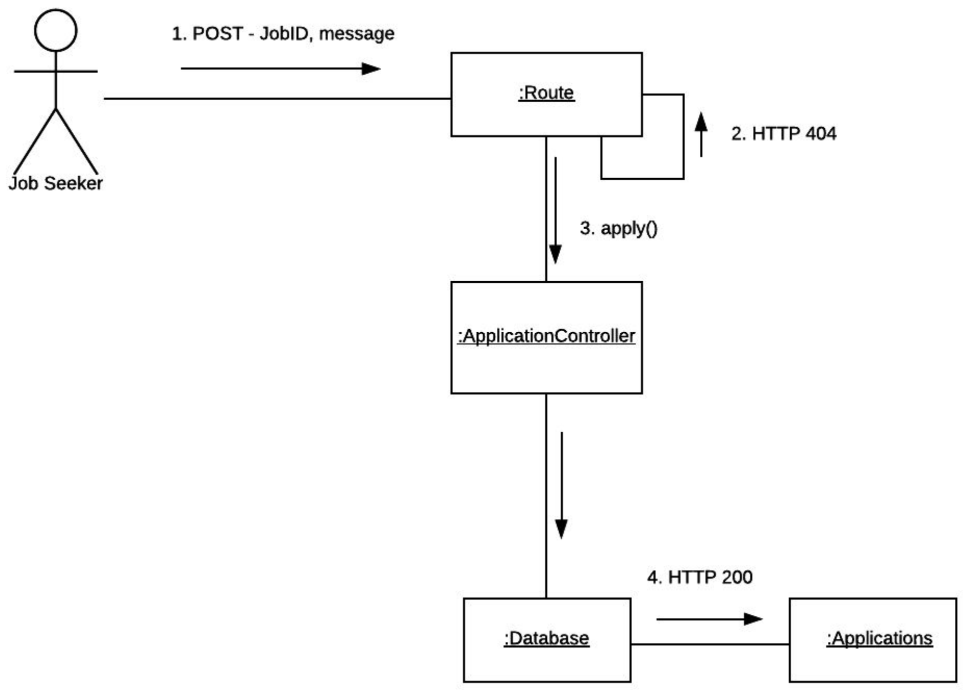
The user sends the jobID via a GET parameter to JobController. The job is then retrieved from the database, and displayed in the job view.



4.1.18 Permit job seekers to apply for a job.

This diagram explains the process by which a job seeker applies for a job.

After successful application, the job seeker is sent to the applications view - where they can view their active applications.



4.1.19 Permit job seekers to upload a resume.

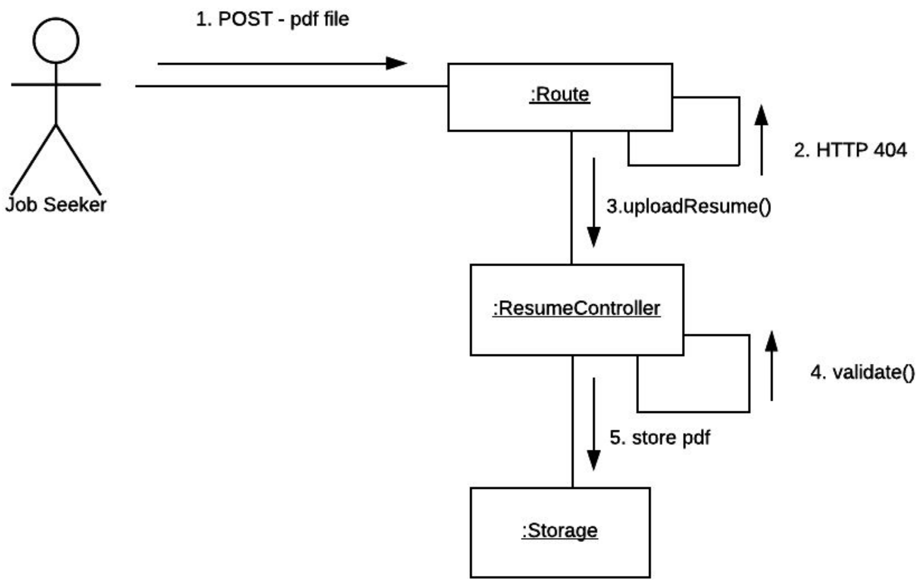
This diagram explains the process by which a job seeker uploads their resume.

The job seekers POSTs their resume (in PDF format) to ResumeController via the upload resume form in the profile view. After validating that the file is of PDF MIME type, the PDF is stored in storage.

The filename of the PDF stored is statically set as *resume-<auth-jobseeker-uuid>-.pdf*. An example being *resume-4af92e97-8f31-45e0-b39e-f8608ae0cd54.pdf*.

Any newly POSTed PDF from an authenticated user simply overwrites the old resume.

The stored resume is a PDF file that is opened in the browser.

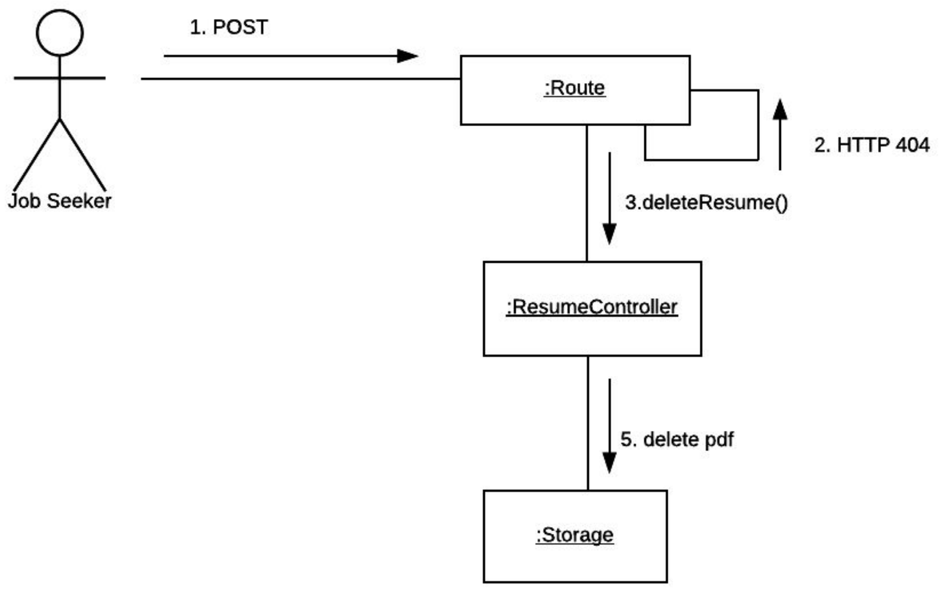


4.1.20 Permit job seekers to delete their resume.

This diagram explains the process by which a job seeker deletes their resume.

The resume deletion route is uses POST, and is a background request. Upon request to delete a resume, ResumeController deletes the currently authenticated job seeker's resume from storage - if it exists.

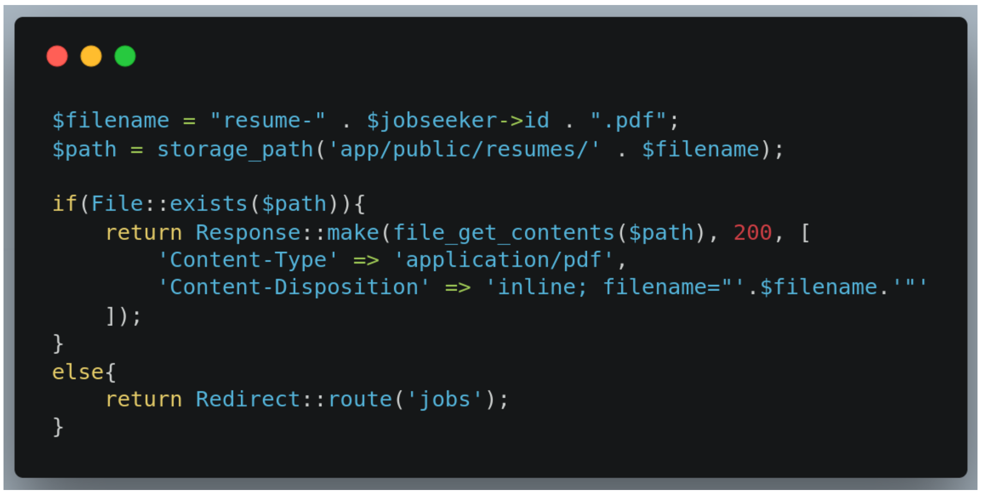
See *"4.6.17. Permit job seekers to upload a resume"* for more information on resume storage.



4.1.21 Permit employers to view an applicant's resume.

This diagram explains the process by which an employer views an applicant's resume.

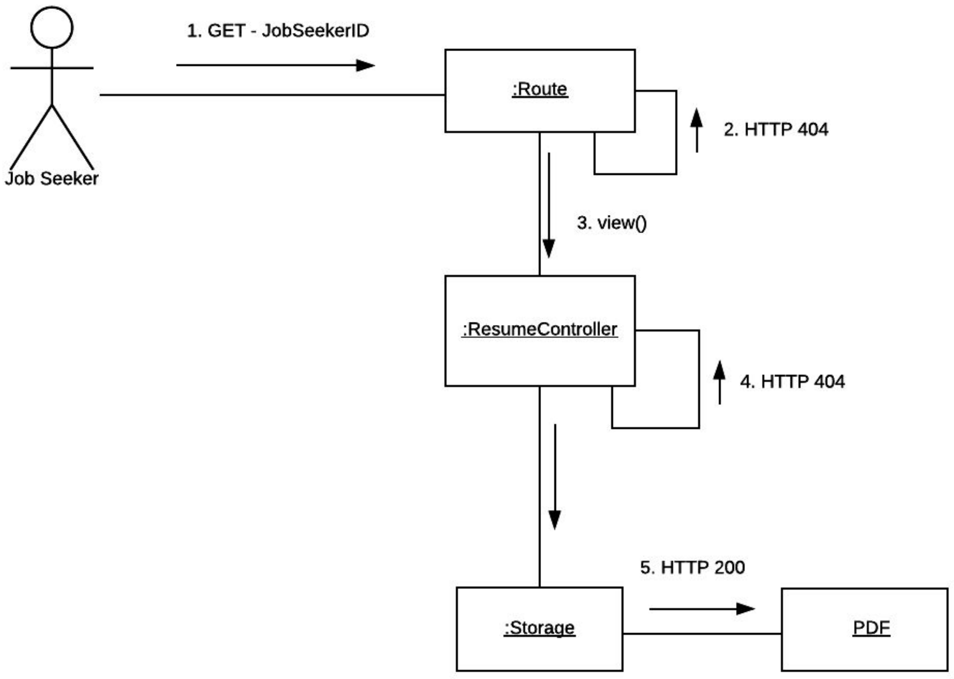
An employer is only authorised to view the resume of a job seeker who has applied to one of their jobs (an applicant).



```
$filename = "resume-" . $jobseeker->id . ".pdf";
$path = storage_path('app/public/resumes/' . $filename);

if(File::exists($path)){
    return Response::make(file_get_contents($path), 200, [
        'Content-Type' => 'application/pdf',
        'Content-Disposition' => 'inline; filename="'.$filename.'"'
    ]);
}
else{
    return Redirect::route('jobs');
}
```

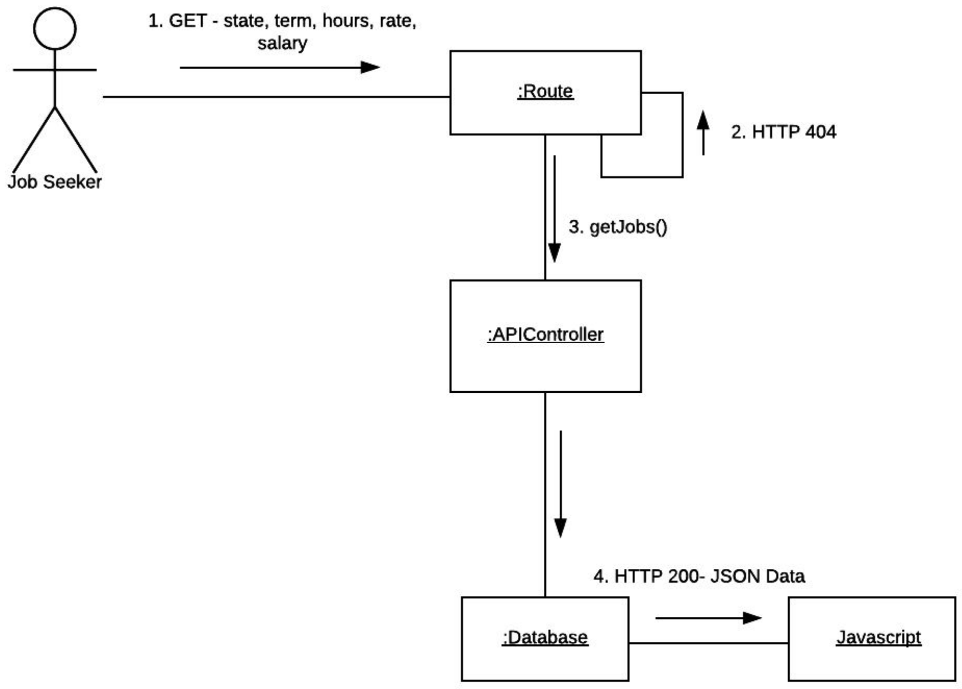
The stored resume is a PDF file that is opened in the browser.



4.1.22 Permit job seekers to view matching jobs by filter.

This diagram explains the process by which a job seeker views matching jobs by filter.

See “4.1.1 Perform matchmaking on jobs to job seekers” for detailed information on job to job seeker matchmaking.



5 Database Architecture

See the database schema diagrams for *Users (job seekers)*, *Employers*, *Jobs*, and *Applications* in “Database schema diagrams” under 11. Appendices.

Each table in the database (excluding Laravel’s *password_resets* and *migrations*) relate directly to a model.

The database is sufficiently scalable.

6 Implementation Instructions

Implementation

A linux-based server that is capable of running a traditional LAMP (Linux, Apache, MySQL, and PHP) stack is required. Further technical requirements of the server depend on client usage expectations.

For this project, the team used a server with 1GB of RAM, one virtualised CPU core, and 25GB of solid-state storage.

After installing Apache, MySQL (or MariaDB), and PHP, Composer must be installed. Composer is a dependency manager for PHP.

Following this, Git can be installed and the project can be cloned from the GitHub repositories. Composer can then be used for installation.

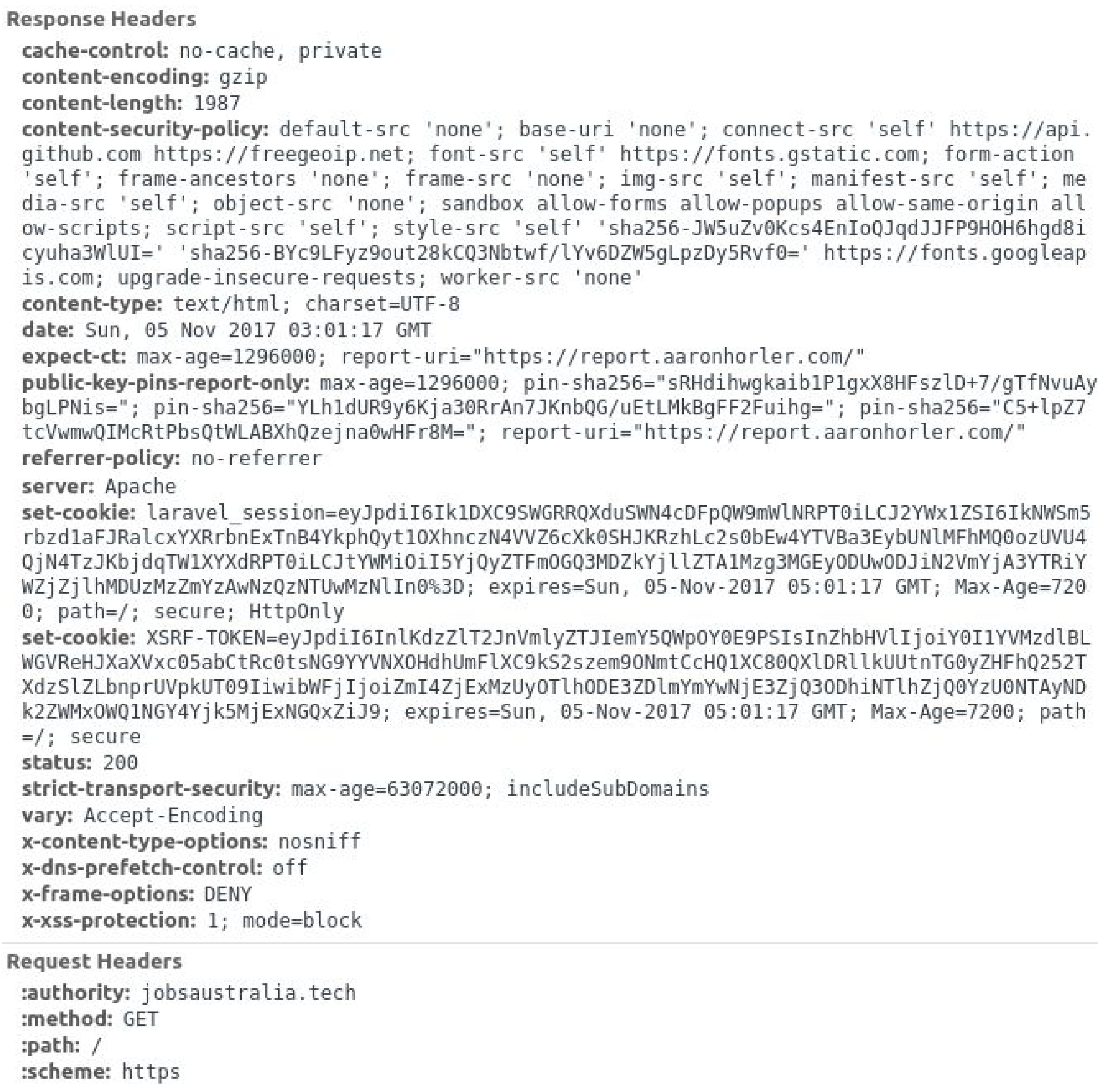
Data migration

The contents of the database can be migrated using the native export and import functionalities in MySQL (see Reference 21). This process should be executed with caution to avoid data loss.

7 Non-functional specifications

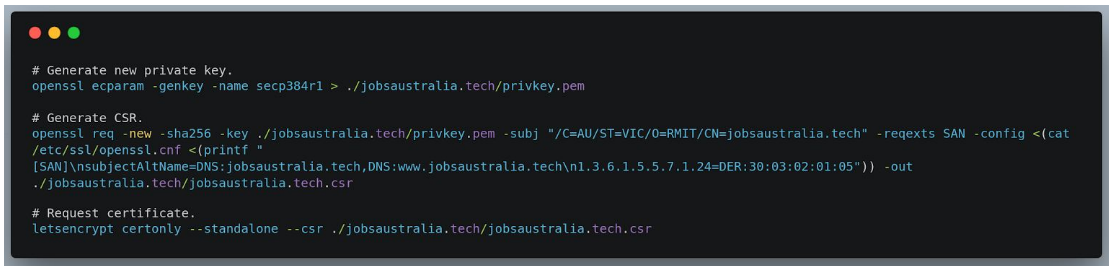
The project includes various non-functional specifications. This section details the *security*, *performance, usability, scalability*, *connectivity*, *readability*, and the *open source software model* of the project.

Security



```
Response Headers
  cache-control: no-cache, private
  content-encoding: gzip
  content-length: 1987
  content-security-policy: default-src 'none'; base-uri 'none'; connect-src 'self' https://api.
  github.com https://freegeoip.net; font-src 'self' https://fonts.gstatic.com; form-action
  'self'; frame-ancestors 'none'; frame-src 'none'; img-src 'self'; manifest-src 'self'; me
  dia-src 'self'; object-src 'none'; sandbox allow-forms allow-popups allow-same-origin all
  ow-scripts; script-src 'self'; style-src 'self' 'sha256-JW5uZv0Kcs4EnIoQJqdJJFP9HOH6hgd8i
  cyuha3WlUI=' 'sha256-BYc9LFyz9out28kCQ3Nbtwf/lYv6DZW5gLpzDy5Rvf0=' https://fonts.googleap
  is.com; upgrade-insecure-requests; worker-src 'none'
  content-type: text/html; charset=UTF-8
  date: Sun, 05 Nov 2017 03:01:17 GMT
  expect-ct: max-age=1296000; report-uri="https://report.aaronhorler.com/"
  public-key-pins-report-only: max-age=1296000; pin-sha256="sRHdihwgkaib1P1gxX8HFszlD+7/gTfNvuAy
  bgLPNis="; pin-sha256="YLh1dUR9y6Kja30RrAn7JKnbQG/uEtLMkBgFF2Fuihg="; pin-sha256="C5+lpZ7
  tcVwmwQIMcRtPbsQtWLABXhQzejna0wHFr8M="; report-uri="https://report.aaronhorler.com/"
  referrer-policy: no-referrer
  server: Apache
  set-cookie: laravel_session=eyJpdiI6Ik1DXC9SWGRRQXduSWN4cDFpQW9mWlNRPT0iLCJ2YWx1ZSI6IkNWSm5
  rbzd1aFJRalcxYXRrbnExTnB4YkphQyt1OXhnczN4VVZ6cXk0SHJKRzhLc2s0bEw4YTVBa3EybUNlMFhMQ0ozUVU4
  QjN4TzJKbjdqTW1XYXdRPT0iLCJtYWMiOiI5YjQyZTFmOGQ3MDZkYjllZTA1Mzg3MGEyODUwODJiN2VmYjA3YTRiY
  WZjZjlhMDUzMzZmYzAwNzQzNTUwMzNlIn0%3D; expires=Sun, 05-Nov-2017 05:01:17 GMT; Max-Age=720
  0; path=/; secure; HttpOnly
  set-cookie: XSRF-TOKEN=eyJpdiI6InlKdzZlT2JnVmlyZTJIemY5QWpOY0E9PSIsInZhbHVlIjoiY0I1YVMzdlBL
  WGVReHJXaXVxc05abCtRc0tsNG9YYVNXOHdhUmFlXC9kS2szem9ONmtCcHQ1XC80QXlDRllkUUtnTG0yZHFhQ252T
  XdzSlZLbnprUVpkUT09IiwibWFjIjoiZmI4ZjExMzUyOTlhODE3ZDlmYmYwNjE3ZjQ3ODhiNTlhZjQ0YzU0NTAyND
  k2ZWMxOWQ1NGY4Yjk5MjExNGQxZiJ9; expires=Sun, 05-Nov-2017 05:01:17 GMT; Max-Age=7200; path
  =/; secure
  status: 200
  strict-transport-security: max-age=63072000; includeSubDomains
  vary: Accept-Encoding
  x-content-type-options: nosniff
  x-dns-prefetch-control: off
  x-frame-options: DENY
  x-xss-protection: 1; mode=block
Request Headers
  :authority: jobsaustralia.tech
  :method: GET
  :path: /
  :scheme: https
```

Connection security



```
# Generate new private key.
openssl ecparam -genkey -name secp384r1 > ./jobsaustralia.tech/privkey.pem

# Generate CSR.
openssl req -new -sha256 -key ./jobsaustralia.tech/privkey.pem -subj "/C=AU/ST=VIC/O=RMIT/CN=jobsaustralia.tech" -reqexts SAN -config <(cat
/etc/ssl/openssl.cnf <(printf "
[SAN]\nsubjectAltName=DNS:jobsaustralia.tech,DNS:www.jobsaustralia.tech\n1.3.6.1.5.5.7.1.24=DER:30:03:02:01:05")) -out
./jobsaustralia.tech/jobsaustralia.tech.csr

# Request certificate.
letsencrypt certonly --standalone --csr ./jobsaustralia.tech/jobsaustralia.tech.csr
```

Both of the sub-project websites use HTTPS connection security. HTTPS (X.509) certificates used in the project were issued by *Let's Encrypt*, a free non-profit certificate authority (CA). An *Elliptic Curve Digital Signature Algorithm (ECDSA)* private key is used. ECDSA is newer than RSA, and is considered more secure, and faster (see Reference 22).

HTTPS connection security is enforced by forced redirection, and the use of the *HTTP Strict Transport Security (HSTS)* response header (see Reference 23 and 24). Using this, browsers are forced to utilise HTTPS connections to the project's domains (and subdomains) for two years. Due to the limited time frame of the project, HSTS is not preloaded.

To protect against person-in-the-middle attacks, the *Public-Key-Pins-Report-Only* response header (see Reference 25) is used to receive reports of any client using non-*Let's Encrypt* certificates. A *Certificate Authority Authorisation (CAA)* record has also been added to the *Vultr*-based name server to only permit certificates to be issued to the project's domains by *Let's Encrypt*.

The *Online Certificate Status Protocol (OCSP)* is used to handle certificate revocations (see Reference 26).

Cross-site scripting (XSS) protection

Various protections against XSS attacks have been implemented. This includes the *implementation of a Content Security Policy* (see Reference 27 and 28) via its associated response header, the *X-Content-Type-Options* response header, and the *X-XSS-Protection* response header.

The project's *Content Security Policy (CSP)* does not permit inline CSS, inline JavaScript, and JavaScript evaluation. These vectors are commonly used to perform XSS attacks. The CSP has also been implemented in a strict mode - only allowing access to content sources we explicitly list.

Content Security Policy

Directive	Functionality
default-src 'none'	Restrict all resources by default.
base-uri 'none'	Do not permit any URL in a *base* element.
connect-src 'self' https://api.github.com https://freegeoip.net	Allow scripts to connect to the same host, GitHub's API, and FreeGeoIP.
font-src 'self' https://fonts.gstatic.com	Allow fonts from the same host, and Google Fonts.
form-action 'self'	Allow the same host for form submissions.
frame-ancestors 'none'	Do not permit the embedding of the site in *frame*, *iframe*, *object*, *embed*, or *applet* elements.
frame-src 'none'	Do not permit *frame* or *iframe* elements.
img-src 'self'	Permit images from the same host.
manifest-src 'self'	Permit manifests from the same host.
media-src 'self'	Permit media (*audio* and *video* elements) from the same host.
object-src 'none'	Do not permit *object*, *embed*, or *applet* elements.
sandbox allow-forms allow-popups allow-same-origin allow-scripts	Enable a browser sandbox, except allow form submissions, popups, and scripts.
script-src 'self'	Permit scripts from the same host.

style-src 'self' 'sha256-...' https://fonts.googleapis.com	Permit styles from the same host, Google Fonts, and allow specific inline Laravel error page styles.
upgrade-insecure-requests	Upgrade any HTTP connection (on any resource) to HTTPS.
worker-src 'none'	Do not permit *Worker*, *SharedWorker*, or *ServiceWorker* scripts.

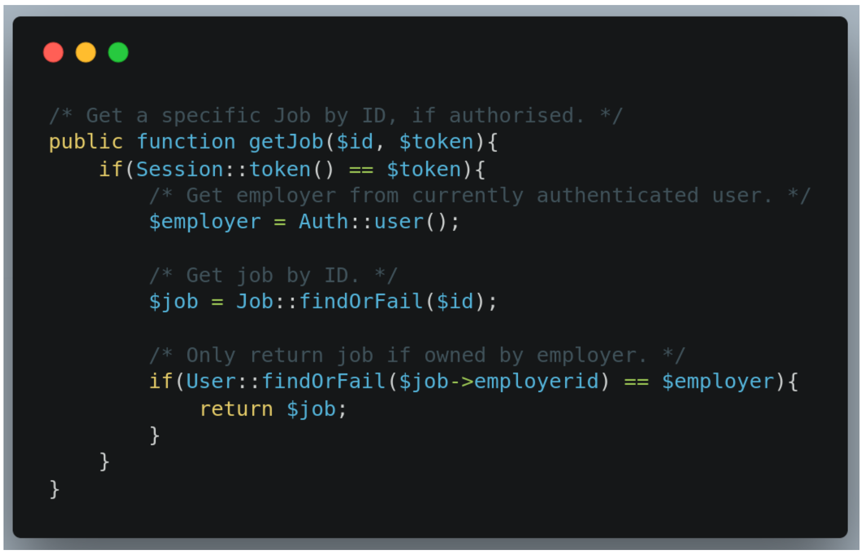
Laravel also provides protection against XSS attacks (see Reference 29).

SQL injection protection

Laravel provides protection against SQL injection. This is achieved via Eloquent in Laravel building prepared statements, and other precautions (see Reference 29).

Cross-Site Request Forgery (CSRF) protection

All forms are protected by Laravel using a CSRF token (see Reference 30).



```php
/* Get a specific Job by ID, if authorised. */
public function getJob($id, $token){
    if(Session::token() == $token){
        /* Get employer from currently authenticated user. */
        $employer = Auth::user();

        /* Get job by ID. */
        $job = Job::findOrFail($id);

        /* Only return job if owned by employer. */
        if(User::findOrFail($job->employerid) == $employer){
            return $job;
        }
    }
}
```

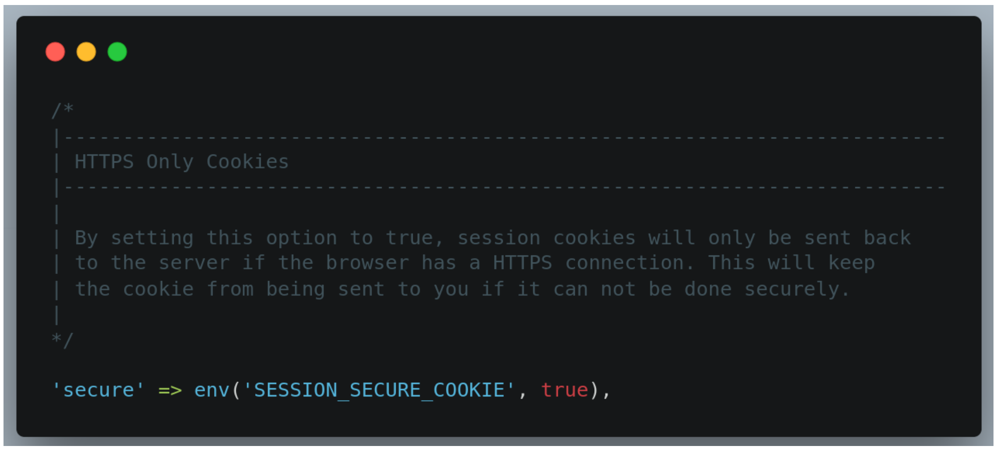
Manual CSRF token verification has also been implemented in API functions.

Project websites are not accessible via *cross-origin resource sharing (CORS)*, this is beneficial to security (see Reference 31).

Clickjacking protection

Project websites are protected from clickjacking attacks via the *X-Frame-Options* response header, and the *frame-ancestors* directive in the CSP (see Reference 32 and 33).

Cookies



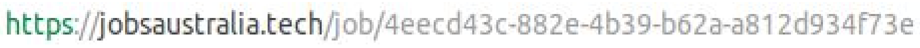
```
/*
|--------------------------------------------------------------------------
| HTTPS Only Cookies
|--------------------------------------------------------------------------
|
| By setting this option to true, session cookies will only be sent back
| to the server if the browser has a HTTPS connection. This will keep
| the cookie from being sent to you if it can not be done securely.
|
*/

'secure' => env('SESSION_SECURE_COOKIE', true),
```

HTTPS connection security is further enforced by setting the *Secure* flag on cookies. This forces browsers to only accept the cookie over a HTTPS connection (see Reference 34).

Session cookies are also set with the *HttpOnly* flag. This prevents non-http access (by JavaScript, for example) (see Reference 34).

Universally unique identifiers (UUIDs)



https://jobsaustralia.tech/job/4eecd43c-882e-4b39-b62a-a812d934f73e

By default, Laravel uses auto-incrementing integers as IDs for models. In review, the team found this to be a security vulnerability when it was allowed, for example, users and jobs to be accessed via their IDs.

The issue is that it is possible to deduce other model IDs from the ID of a current model. An example of this is assuming that, if a job just created has the ID of 4, there are also probably jobs with IDs 1 to 3, 5, and so on.

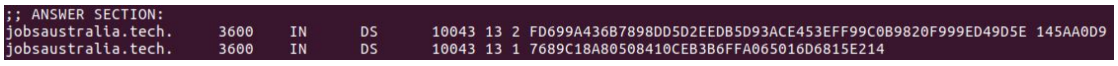
To resolve this, and improve security, model identifiers were switched to UUIDS.

UUIDs are 128-bit random strings. Being random, they are not possible to deduce (see Reference 35).

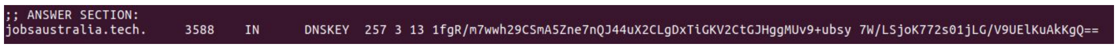
Software versions

The latest versions of Apache, PHP, MySQL, OpenSSH, and Laravel are used on the production server. The team considers this beneficial to security.

DNSSEC



```
;; ANSWER SECTION:
jobsaustralia.tech.     3600    IN      DS      10043 13 2 FD699A436B7898DD5D2EEDB5D93ACE453EFF99C0B9820F999ED49D5E 145AA0D9
jobsaustralia.tech.     3600    IN      DS      10043 13 1 7689C18A80508410CEB3B6FFA065016D6815E214
```



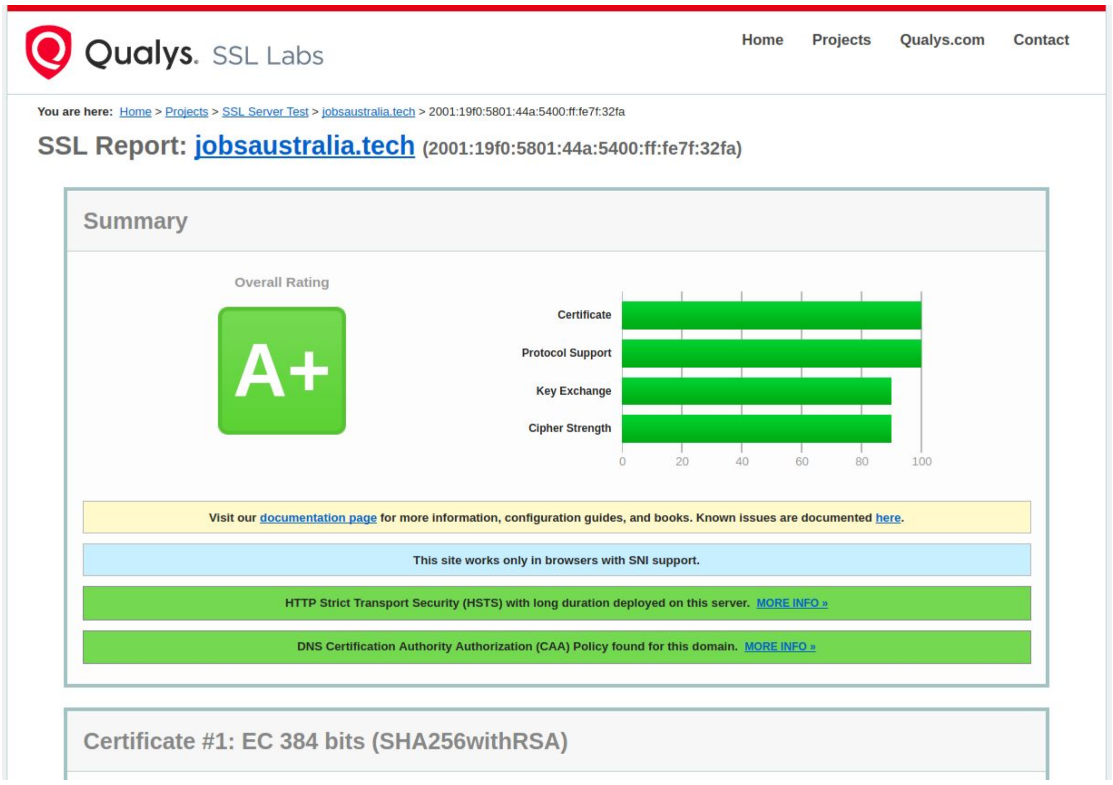
```
;; ANSWER SECTION:
jobsaustralia.tech.     3588    IN      DNSKEY  257 3 13 1fgR/m7wwh29CSmA5Zne7nQJ44uX2CLgDxTiGKV2CtGJHggMUv9+ubsy 7W/LSjoK772s01jLG/V9UElKuAkKgQ==
```

The domain name *jobsaustralia.tech* is DNSSEC-signed. This protects against DNS-based attacks.

This was achieved by adding DS and DNSKEY records to the DNS zone.

Tests

HTTPS connection security was verified using Qualys SSL Labs, where the project scored an *A+* (see Reference 36).



You are here: Home > Projects > SSL Server Test > jobsaustralia.tech > 2001:19f0:5801:44a:5400:ff:fe7f:32fa

SSL Report: jobsaustralia.tech (2001:19f0:5801:44a:5400:ff:fe7f:32fa)

Summary

Visit our documentation page for more information, configuration guides, and books. Known issues are documented here.

This site works only in browsers with SNI support.

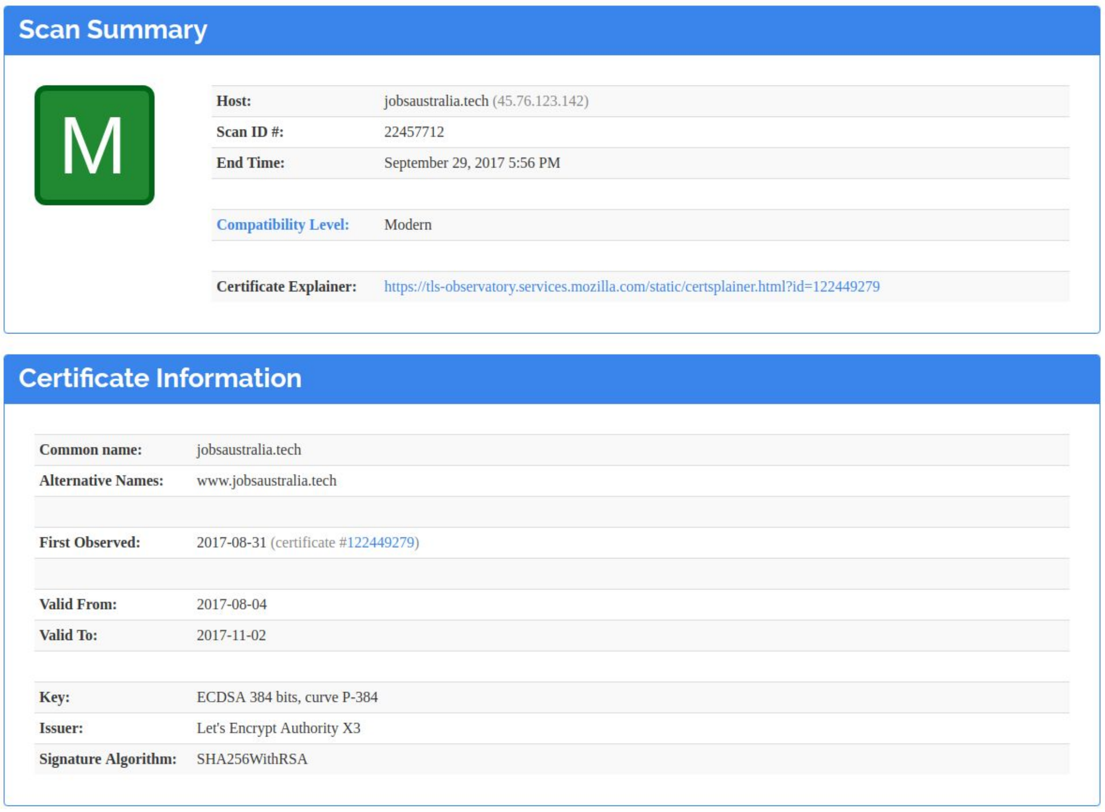
HTTP Strict Transport Security (HSTS) with long duration deployed on this server. MORE INFO »

DNS Certification Authority Authorization (CAA) Policy found for this domain. MORE INFO »

Certificate #1: EC 384 bits (SHA256withRSA)

To achieve this, the team enabled HSTS, created a DNS CAA record permitting only *letsencrypt.org* to issue certificates, disabled all TLS/SSL protocol versions expect TLS 1.2, and used a list of ciphersuites recommended by Mozilla (see Reference 37).

The project's TLS implementation was verified using Mozilla's *TLS Observatory*, where the project scored *Modern* (see Reference 38).



Scan Summary

M

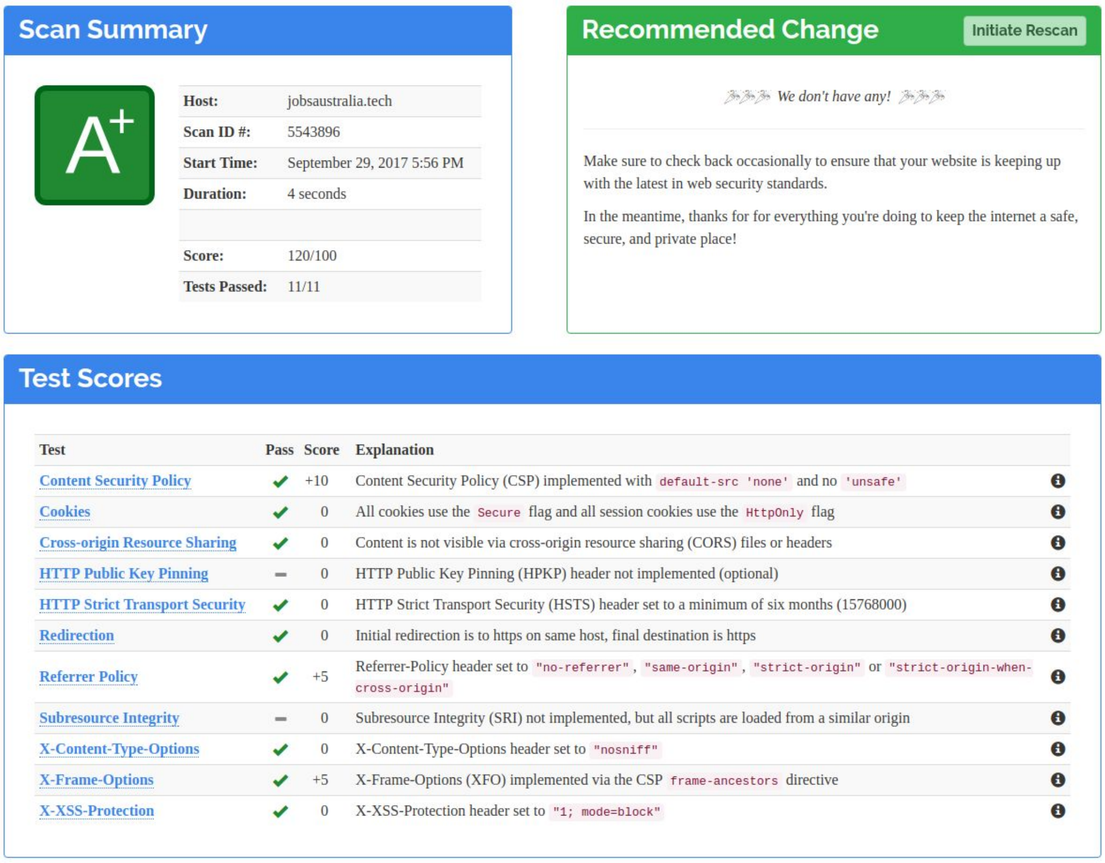
Host:	jobsaustralia.tech (45.76.123.142)
Scan ID #:	22457712
End Time:	September 29, 2017 5:56 PM
Compatibility Level:	Modern
Certificate Explainer:	https://tls-observatory.services.mozilla.com/static/certsplainer.html?id=122449279

Certificate Information

Common name:	jobsaustralia.tech
Alternative Names:	www.jobsaustralia.tech
First Observed:	2017-08-31 (certificate #122449279)
Valid From:	2017-08-04
Valid To:	2017-11-02
Key:	ECDSA 384 bits, curve P-384
Issuer:	Let's Encrypt Authority X3
Signature Algorithm:	SHA256WithRSA

To achieve this, the team disabled all TLS/SSL protocol versions expect TLS 1.2, used a list of ciphersuites recommended by Mozilla (see Reference 37), and used an ECDSA private key.

Our general security implementation was verified using Mozilla's *HTTP Observatory*, where the project scored an *A+* (120/100), and passed all tests (see Reference 38).



Scan Summary

A+

Host:	jobsaustralia.tech
Scan ID #:	5543896
Start Time:	September 29, 2017 5:56 PM
Duration:	4 seconds
Score:	120/100
Tests Passed:	11/11

Recommended Change

We don't have any!

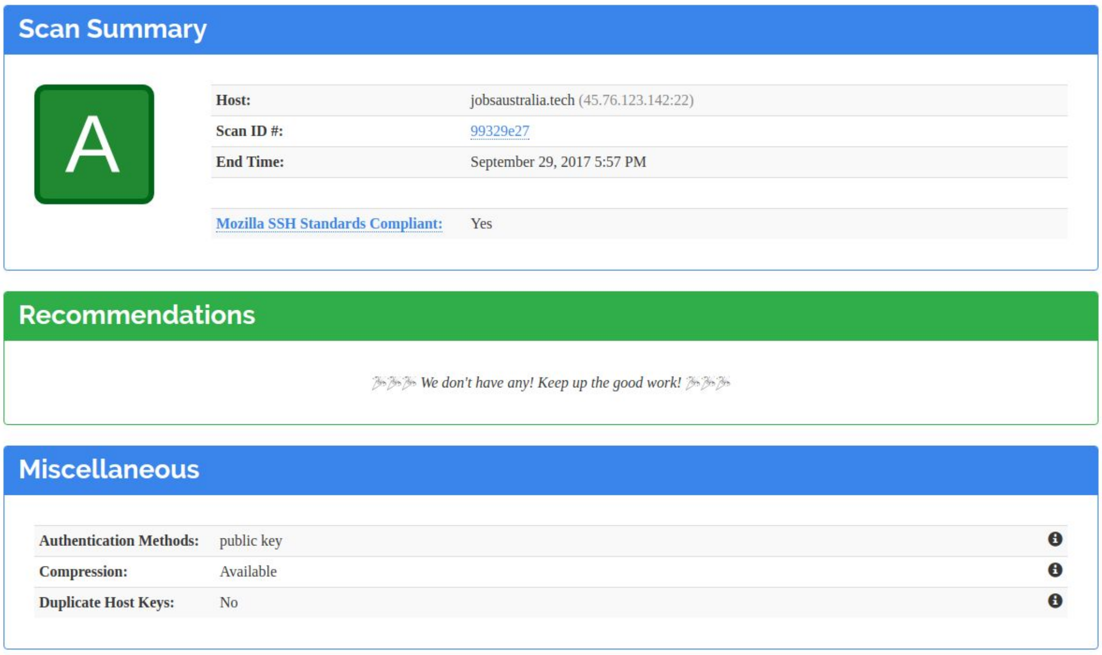
Make sure to check back occasionally to ensure that your website is keeping up with the latest in web security standards.

In the meantime, thanks for for everything you're doing to keep the internet a safe, secure, and private place!

Test Scores

Test	Pass	Score	Explanation
Content Security Policy	✓	+10	Content Security Policy (CSP) implemented with `default-src 'none'` and no `'unsafe'`
Cookies	✓	0	All cookies use the `Secure` flag and all session cookies use the `HttpOnly` flag
Cross-origin Resource Sharing	✓	0	Content is not visible via cross-origin resource sharing (CORS) files or headers
HTTP Public Key Pinning	–	0	HTTP Public Key Pinning (HPKP) header not implemented (optional)
HTTP Strict Transport Security	✓	0	HTTP Strict Transport Security (HSTS) header set to a minimum of six months (15768000)
Redirection	✓	0	Initial redirection is to https on same host, final destination is https
Referrer Policy	✓	+5	Referrer-Policy header set to `"no-referrer"`, `"same-origin"`, `"strict-origin"` or `"strict-origin-when-cross-origin"`
Subresource Integrity	–	0	Subresource Integrity (SRI) not implemented, but all scripts are loaded from a similar origin
X-Content-Type-Options	✓	0	X-Content-Type-Options header set to `"nosniff"`
X-Frame-Options	✓	+5	X-Frame-Options (XFO) implemented via the CSP `frame-ancestors` directive
X-XSS-Protection	✓	0	X-XSS-Protection header set to `"1; mode=block"`

Our OpenSSH implementation was verified using Mozilla's *SSH Observatory*, where the project scored an *A* (see Reference 38). This was achieved by making the OpenSSH server *Mozilla SSH Standards Compliant* (see Reference 39).



Scan Summary

A

Host:	jobsaustralia.tech (45.76.123.142:22)
Scan ID #:	99329e27
End Time:	September 29, 2017 5:57 PM
Mozilla SSH Standards Compliant:	Yes

Recommendations

We don't have any! Keep up the good work!

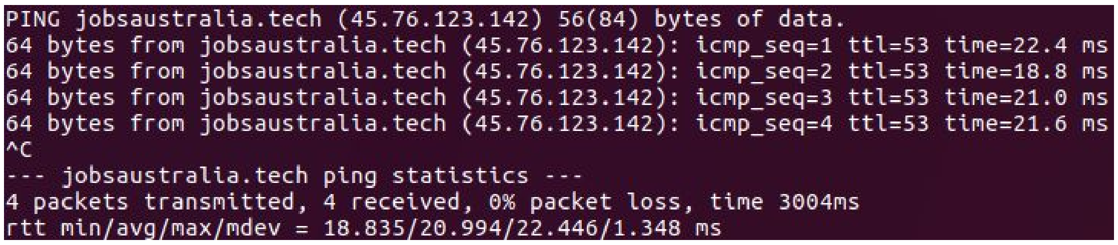
Miscellaneous

Authentication Methods:	public key
Compression:	Available
Duplicate Host Keys:	No

Performance

Server performance

Project websites are hosted on an SSD-based VPS (*Vultr*) in Sydney, Australia (See more about Vultr in “Vultr” under 2. Technical Environment). This location was chosen as it is geographically closest to southern Victoria (where the team is located), and also geographically close to most of Australia’s population.

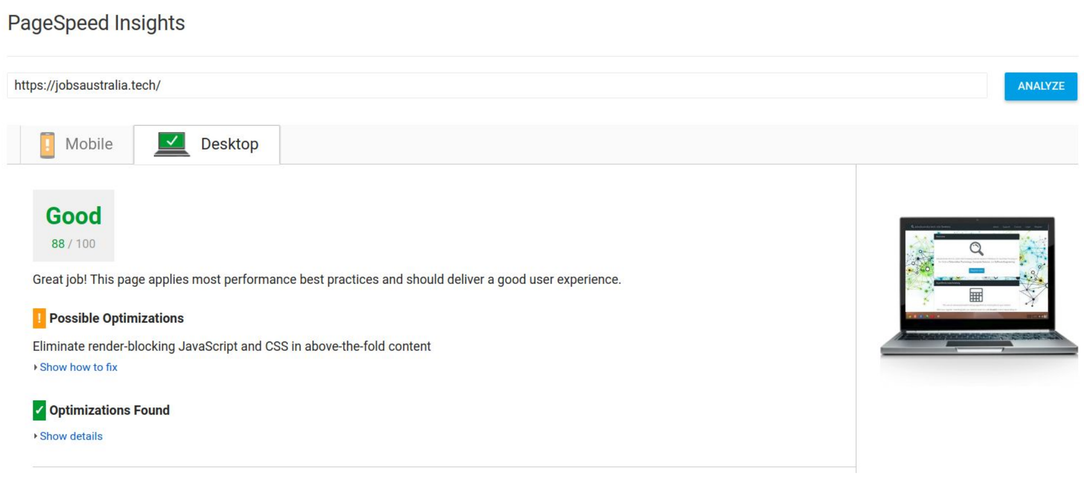


```
PING jobsaustralia.tech (45.76.123.142) 56(84) bytes of data.
64 bytes from jobsaustralia.tech (45.76.123.142): icmp_seq=1 ttl=53 time=22.4 ms
64 bytes from jobsaustralia.tech (45.76.123.142): icmp_seq=2 ttl=53 time=18.8 ms
64 bytes from jobsaustralia.tech (45.76.123.142): icmp_seq=3 ttl=53 time=21.0 ms
64 bytes from jobsaustralia.tech (45.76.123.142): icmp_seq=4 ttl=53 time=21.6 ms
^C
--- jobsaustralia.tech ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 18.835/20.994/22.446/1.348 ms
```

In latency testing from a standard south-east suburban Melbourne residential ISP connection, the latency to the production server was between 19ms and 22ms. Tests were performed between 4pm and 5pm on a Saturday afternoon.

HTTP/2 has been enabled in *Apache*, using the *apache2* package published by *Ondřej Surý* with *mpm_event* and *php7.0-fpm*. This should marginally improve load performance (see Reference 40).

Page performance



PageSpeed Insights

https://jobsaustralia.tech/ ANALYZE

Mobile Desktop

Good
88 / 100

Great job! This page applies most performance best practices and should deliver a good user experience.

Possible Optimizations

Eliminate render-blocking JavaScript and CSS in above-the-fold content

Show how to fix

Optimizations Found

Show details

Google PageSpeed Insights was used to assess the loading performance of project web pages.

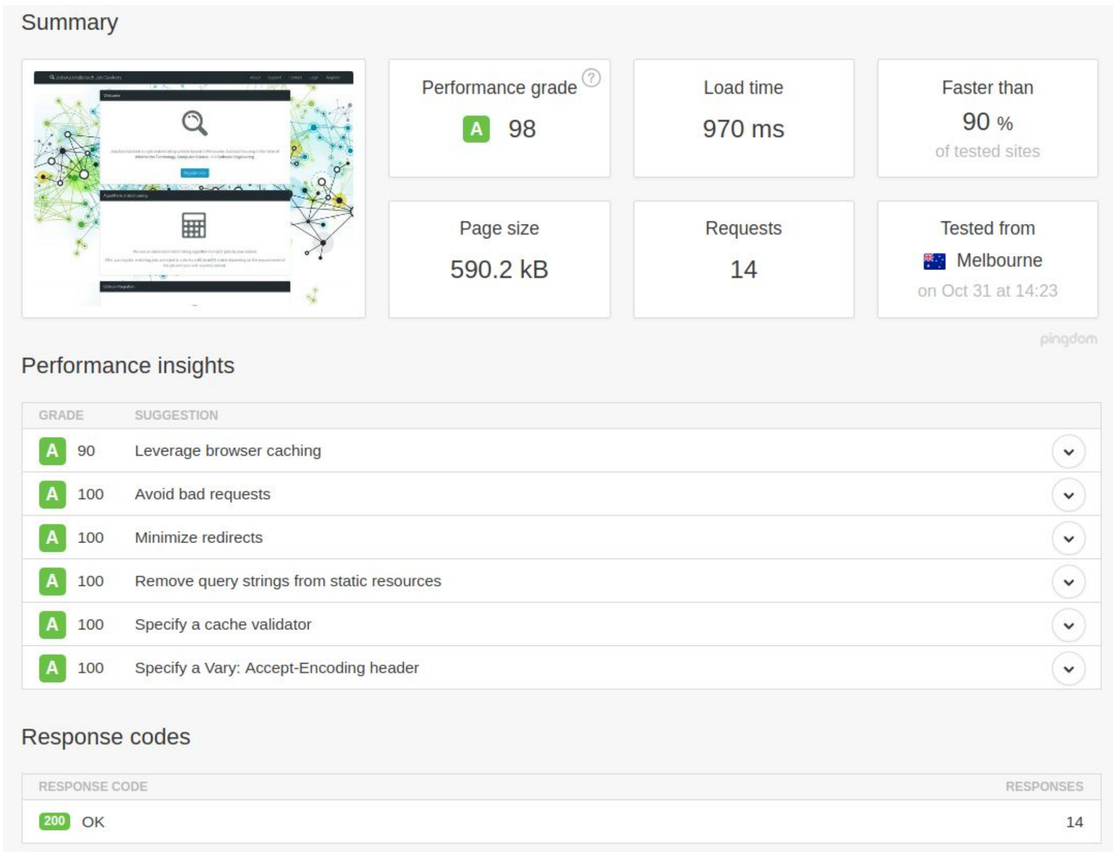
Per instruction by *PageSpeed* (see Reference 41), images were optimised using the *PageSpeed* algorithm, CSS and JavaScript were minified, and browsers were instructed to cache suitable resources (via the Cache-Control response header).

The team were unable to remove render-blocking CSS and JavaScript due to workflow restrictions, and the *Content Security Policy (CSP)* (pagespeed recommends inlining these resources).

In experimentation, the steam tried the *mod_pagespeed Apache* module. This was found to be unsuitable because it violated the project’s CSP.

Through these efforts, the Desktop *PageSpeed* score was increased from **Poor** to **Good (88/100)**, and the Mobile score was increased from **Poor** to **Needs Work (68/100)**.

Through testing, the team believes system performance is very acceptable.



Summary

Performance grade A 98	Load time 970 ms	Faster than 90 % of tested sites
Page size 590.2 kB	Requests 14	Tested from Melbourne on Oct 31 at 14:23

Performance insights

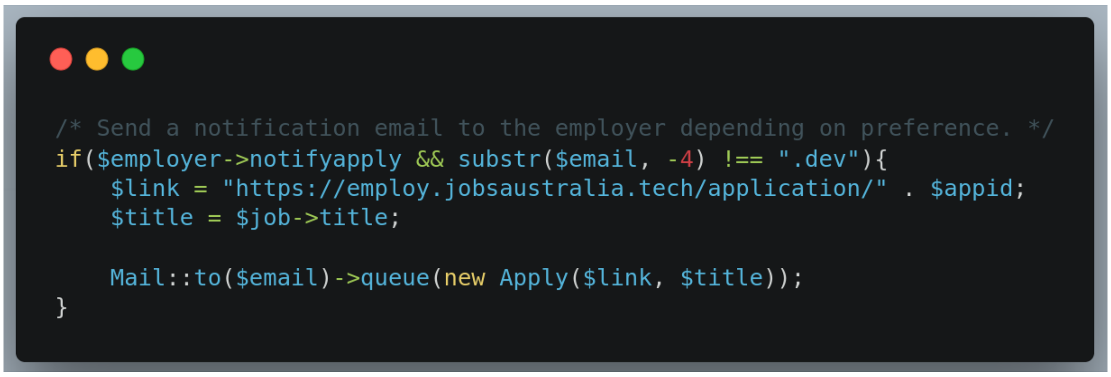
GRADE	SUGGESTION
A 90	Leverage browser caching
A 100	Avoid bad requests
A 100	Minimize redirects
A 100	Remove query strings from static resources
A 100	Specify a cache validator
A 100	Specify a Vary: Accept-Encoding header

Response codes

RESPONSE CODE	RESPONSES
200 OK	14

The project’s Pingdom score was 98.

Mail queuing



```
/* Send a notification email to the employer depending on preference. */
if($employer->notifyapply && substr($email, -4) !== ".dev"){
    $link = "https://employ.jobsaustralia.tech/application/" . $appid;
    $title = $job->title;

    Mail::to($email)->queue(new Apply($link, $title));
}
```

In testing, it was found that PHP response times were reduced significantly by the sending of emails for notifications.

This issue was resolved, improving performance, by implementing mail queuing. With this, Laravel does not wait until an email (or set of emails) is successfully sent before executing the return statement in a function (see Reference 42).

Client-side matchmaking

The decision to implement matchmaking on the client-side (using JavaScript) for both jobs to job seekers and applicants to employers is beneficial to performance. The reason for this is that the server does not need to perform matchmaking. The work is instead offset to each user's browser.

Considering security, and data integrity, client-side matchmaking was deemed not to be a security vulnerability has the calculated percentage match is not stored on the server.

Usability

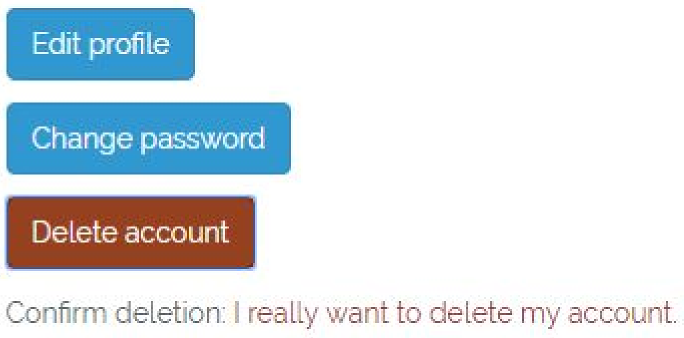
Simple and concise

The project website is simple, and easy to navigate. Users are not bombarded or overwhelmed with information, and can quickly see what is available rather than having to think it through. The navigation bar contains the most important links.

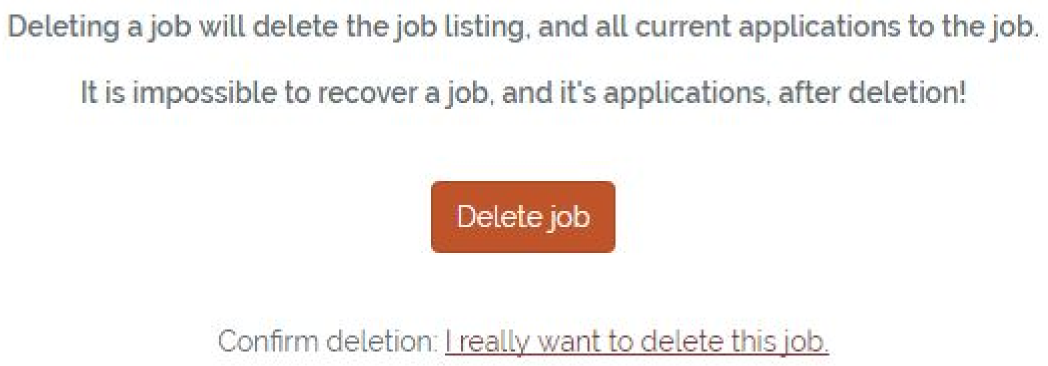
Consistency

The project theme, and colour scheme particularly, is consistent throughout the website. The same navigation model is used in all project web pages. Users can easily browse through project websites without becoming lost.

Error prevention



Confirm deletion: I really want to delete my account.



Deleting a job will delete the job listing, and all current applications to the job.

It is impossible to recover a job, and it's applications, after deletion!

Confirm deletion: I really want to delete this job.

The project websites provide users with a confirmation option before they commit to an important, or potentially damaging, action. This is to prevent errors as the user may change their mind about deleting their account, or deleting a posted job. They may have even pressed the button accidentally therefore it helps them to recover from errors.

Help and documentation
The project websites contain a support pages for users who need help using the system.

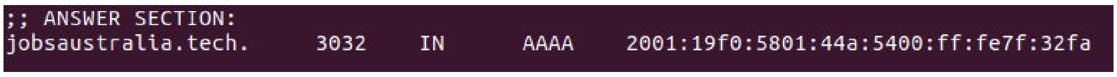
Through frequently asked questions, users can find answers to questions in various categories of issues. If further support is required, users have the option to send the team their enquires through the contact form.

Scalability

The project system is suitable for basic and advanced load balancing. This would require the database be migrated to a standalone server, which would only require minor configuration changes.

The current implementation does not require load balancing, due to limited use.

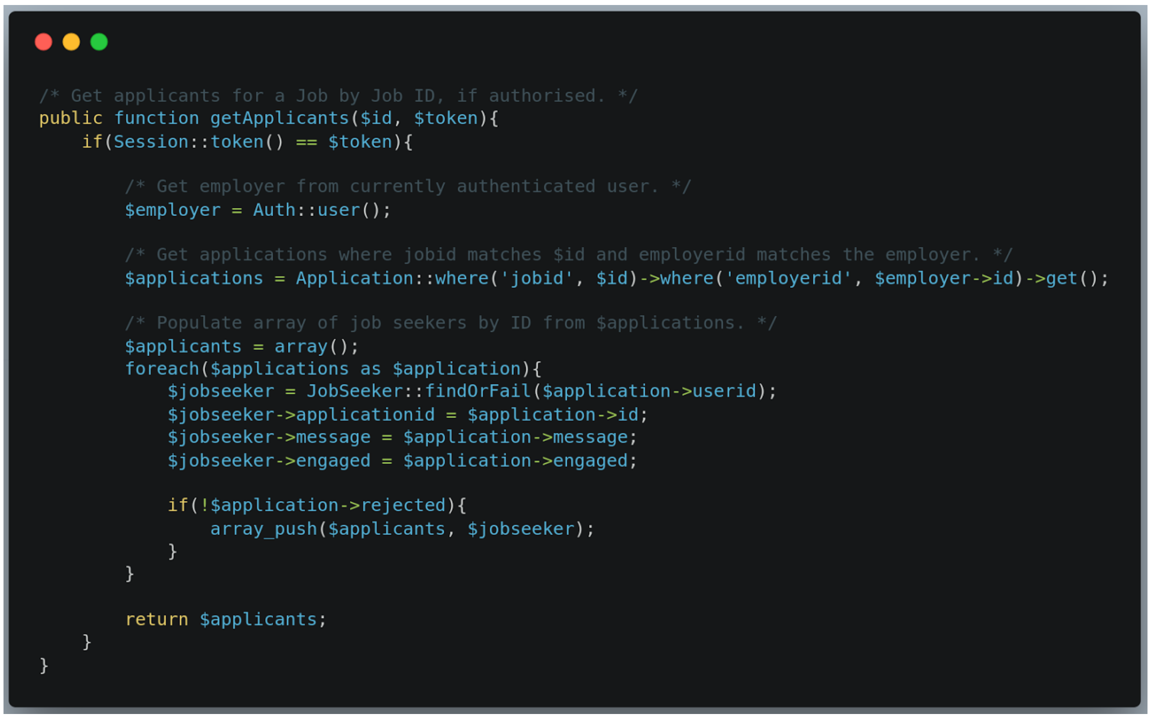
IPv6 connectivity



```
;; ANSWER SECTION:
jobsaustralia.tech.     3032    IN      AAAA    2001:19f0:5801:44a:5400:ff:fe7f:32fa
```

The production server fully supports, and is accessible via, IPv6.

Readability



```
/* Get applicants for a Job by Job ID, if authorised. */
public function getApplicants($id, $token){
    if(Session::token() == $token){

        /* Get employer from currently authenticated user. */
        $employer = Auth::user();

        /* Get applications where jobid matches $id and employerid matches the employer. */
        $applications = Application::where('jobid', $id)->where('employerid', $employer->id)->get();

        /* Populate array of job seekers by ID from $applications. */
        $applicants = array();
        foreach($applications as $application){
            $jobseeker = JobSeeker::findOrFail($application->userid);
            $jobseeker->applicationid = $application->id;
            $jobseeker->message = $application->message;
            $jobseeker->engaged = $application->engaged;

            if(!$application->rejected){
                array_push($applicants, $jobseeker);
            }
        }

        return $applicants;
    }
}
```

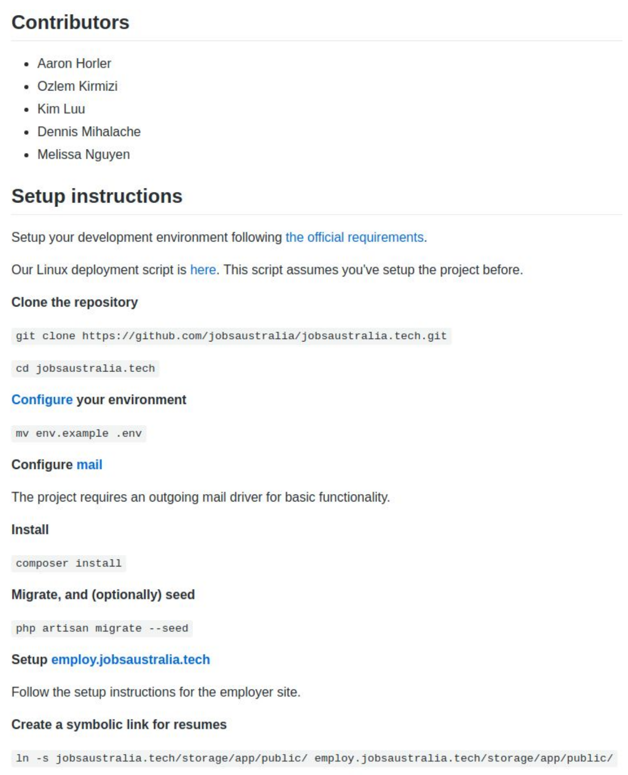
Project code follows a predefined standard.

- Functions and variables are named appropriately.
- Indentation is four spaces per nesting.
- Comments are written above every function, and above any segment of code requiring explanation.
- English spelling and grammar rules are followed.
- Unused code is removed.

The readability of the project's codebase is improved by these standards.

Open source development

The project is entirely open source, and licensed under the MIT license.



Contributors

- Aaron Horler
- Ozlem Kirmizi
- Kim Luu
- Dennis Mihalache
- Melissa Nguyen

Setup instructions

Setup your development environment following the official requirements.

Our Linux deployment script is here. This script assumes you've setup the project before.

Clone the repository

```
git clone https://github.com/jobsaustralia/jobsaustralia.tech.git
```

```
cd jobsaustralia.tech
```

Configure your environment

```
mv env.example .env
```

Configure mail

The project requires an outgoing mail driver for basic functionality.

Install

```
composer install
```

Migrate, and (optionally) seed

```
php artisan migrate --seed
```

Setup employ.jobsaustralia.tech

Follow the setup instructions for the employer site.

Create a symbolic link for resumes

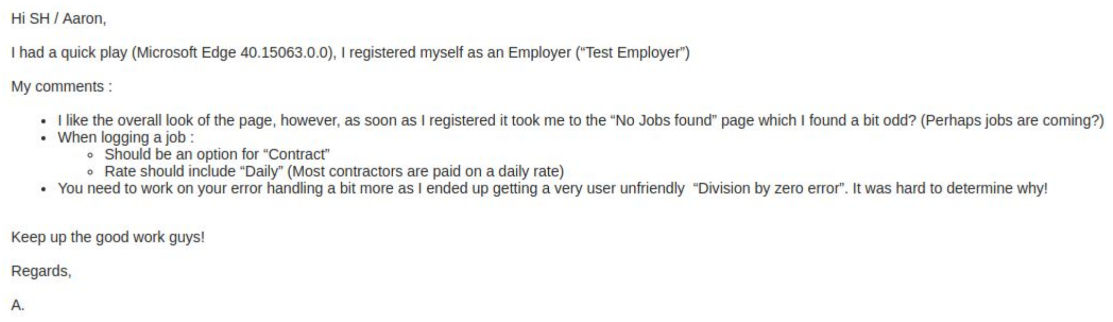
```
ln -s jobsaustralia.tech/storage/app/public/ employ.jobsaustralia.tech/storage/app/public/
```

The team is further facilitating open source contribution via the readme files on GitHub, which detail configuration.

8 Summary of test results

Refer to the Test Plan (external document) document for information on testing practices, and results.

Further, an IT professional was contact, asking for feedback on the employer's website.



Hi SH / Aaron,

I had a quick play (Microsoft Edge 40.15063.0.0), I registered myself as an Employer ("Test Employer")

My comments :

- I like the overall look of the page, however, as soon as I registered it took me to the "No Jobs found" page which I found a bit odd? (Perhaps jobs are coming?)
- When logging a job :
  - Should be an option for "Contract"
  - Rate should include "Daily" (Most contractors are paid on a daily rate)
- You need to work on your error handling a bit more as I ended up getting a very user unfriendly "Division by zero error". It was hard to determine why!

Keep up the good work guys!

Regards,

A.

Based on the suggestions from this individual, the "No Jobs Found" message was altered to "You've Posted No Jobs" - as employers are only able to see jobs they have posted.

A JQuery issue in Edge and Internet Explorer was also fixed, plus the division by zero error, and a "Contract" option was added to term and a "Daily" option was added to rate.

See *"Tests under Security in 7. Non-functional specifications"* for information of security testings. Also see *"Performance in 7. Non-functional specification"* for information on performance testing.

9 Known Issues & Risks

The project currently has no known issues or risks. Past issues, and risks, are recorded in the *Issues Register* and the *Risk Register*, respectively.

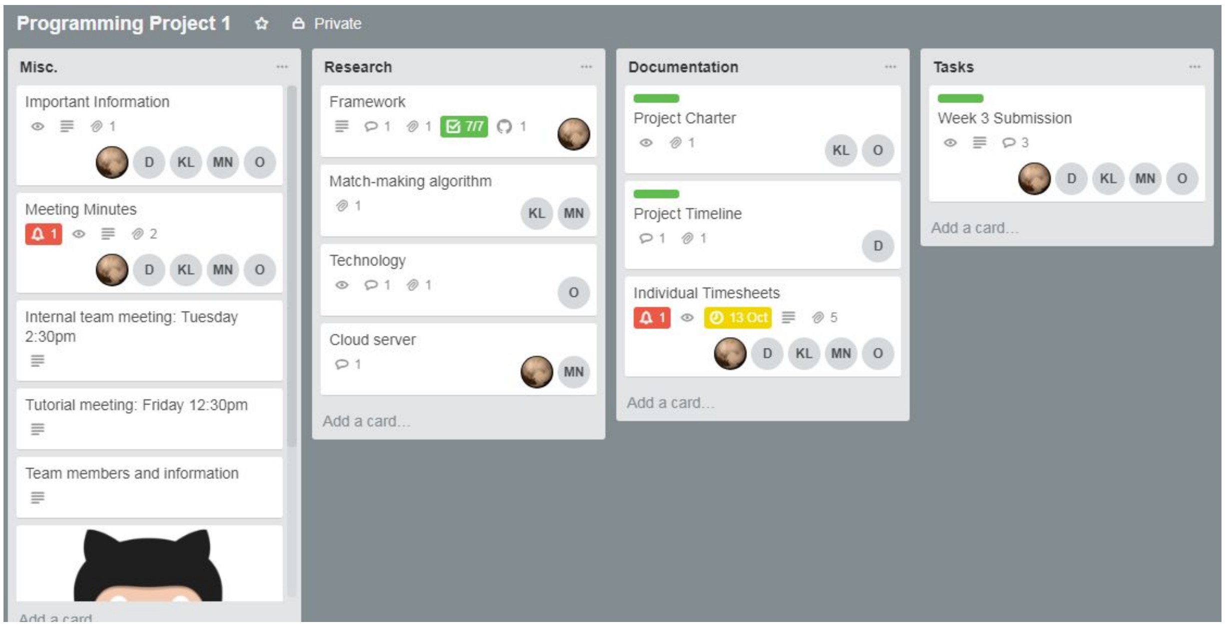
10 References

1. PHP: Hypertext Processor. PHP. Accessed October 15th, 2017. https://secure.php.net/
2. Laravel - The PHP Framework For Web Artisans. Taylor Otwell. Accessed October 15th, 2017. https://laravel.com
3. MySQL. MySQL. Accessed October 15th, 2017. https://www.mysql.com/
4. Welcome! - The Apache HTTP Server Project. Apache Software Foundation. Accessed October 15th, 2017. https://httpd.apache.org
5. The leading operating system for PCs, IoT devices, servers and the cloud | Ubuntu. Canonical. Accessed October 15th, 2017. https://www.ubuntu.com
6. XAMPP Installers and Downloads for Apache Friends. Apache Friends. Accessed October 15th, 2017. https://www.apachefriends.org/index.html
7. SSD VPS Servers, Cloud Servers and Cloud Hosting by Vultr - Vultr.com. Vultr. Accessed October 15th, 2017. https://www.vultr.com
8. GitHub. GitHub. Accessed October 15th, 2017. https://github.com
9. Git. Git. Accessed October 15th, 2017. https://git-scm.com/
10. JobsAustralia.tech. GitHub. Accessed October 15th, 2017. https://github.com/jobsaustralia
11. Composer. Composer. Accessed October 15th, 2017. https://getcomposer.org
12. Font Awesome, the iconic font and CSS toolkit. Dave Gandy. Accessed October 15th, 2017. http://fontawesome.io/
13. GitHub API v3 | GitHub Developer Guide. GitHub. Accessed October 15th, 2017. https://developer.github.com/v3/
14. jQuery. jQuery Foundation. Accessed October 15th, 2017. https://jquery.com
15. Notepad++ Home. Don Ho. Accessed October 15th, 2017. https://notepad-plus-plus.org
16. PHPUnit - The PHP Testing Framework. Sebastian Bergmann. Accessed October 15th, 2017. https://phpunit.de/
17. Sublime Text - A sophisticated text editor for code, markup and prose. Sublime HQ Pty Ltd. Accessed October 15th, 2017. https://www.sublimetext.com/
18. Icon Font & SVG Icon Sets ❍ IcoMoon. Keyamoon. Accessed October 15th, 2017. https://icomoon.io
19. Let's Encrypt - Free SSL/TLS Certificates. Let's Encrypt. Accessed October 15th, 2017. https://letsencrypt.org/
20. Transactional Email API Service For Developers | Mailgun. Mailgun Technologies, Inc. Accessed October 15th, 2017. https://www.mailgun.com/
21. How To Import and Export Databases in MySQL or MariaDB | DigitalOcean. DigitalOcean. Accessed October 15th, 2017. https://www.digitalocean.com/community/tutorials/how-to-import-and-export-databases-in-mysql-or-mariadb
22. Testing out ECDSA certificates. Scott Helme. Accessed October 15th, 2017. https://scotthelme.co.uk/ecdsa-certificates/
23. HTTP Strict Transport Security Cheat Sheet - OWASP. Open Web Application Security Project. Accessed October 15th, 2017. https://www.owasp.org/index.php/HTTP_Strict_Transport_Security_Cheat_Sheet
24. Strict-Transport-Security - HTTP | MDN. Mozilla and individual contributors. Accessed October 15th, 2017. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security
25. Public-Key-Pins-Report-Only - HTTP | MDN. Mozilla and individual contributors. Accessed October 15th, 2017. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Public-Key-Pins-Report-Only
26. OCSP Must-Staple. Scott Helme. Accessed October 15th, 2017. https://scotthelme.co.uk/ocsp-must-staple/

27. Content Security Policy (CSP) - HTTP | MDN. Mozilla and individual contributors. Accessed October 15th, 2017. https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP
28. Content Security Policy CSP Reference & Examples. Foundeo Inc. Accessed October 15th, 2017. https://content-security-policy.com
29. How Laravel 5 Prevents SQL Injection, Cross-Site Request Forgery, and Cross-Site Scripting - Easy Laravel. W. Jason Gilmore. Accessed October 15th, 2017. http://www.easylaravelbook.com/blog/2015/07/22/how-laravel-5-prevents-sql-injection-cross-site-request-forgery-and-cross-site-scripting/
30. CSRF Protection - Laravel - The PHP Framework For Web Artisans. Taylor Otwell. Accessed October 15th, 2017. https://laravel.com/docs/5.5/csrf
31. Cross-Origin Resource Sharing (CORS) - HTTP | MDN. Mozilla and individual contributors. Accessed October 15th, 2017. https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS
32. X-Frame-Options - HTTP | MDN | MDN. Mozilla and individual contributors. Accessed October 15th, 2017. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
33. Clickjacking Defense Cheat Sheet - OWASP. Open Web Application Security Project. Accessed October 15th, 2017. https://www.owasp.org/index.php/Clickjacking_Defense_Cheat_Sheet
34. HTTP cookies - HTTP | MDN. Mozilla and individual contributors. Accessed October 15th, 2017. https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
35. RFC 4122 - A Universally Unique IDentifier (UUID) URN Namespace. Leach, Paul J. Mealling, Michael. Salz, Rich. Accessed October 15th, 2017. https://tools.ietf.org/html/rfc4122
36. SSL Server Test: jobsaustralia.tech (Powered by Qualys SSL Labs). Qualys SSL Labs. Accessed October 15th, 2017. https://www.ssllabs.com/ssltest/analyze.html?d=jobsaustralia.tech
37. Security/Server Side TLS - MozillaWiki. MozillaWiki. Accessed October 15th, 2017. https://wiki.mozilla.org/Security/Server_Side_TLS#Modern_compatibility
38. Observatory by Mozilla. April King. Accessed October 15th, 2017. https://observatory.mozilla.org/analyze.html?host=jobsaustralia.tech
39. Security/Guidelines/OpenSSH - MozillaWiki. MozillaWiki. Accessed October 15th, 2017. https://wiki.mozilla.org/Security/Guidelines/OpenSSH
40. A Simple Performance Comparison of HTTPS, SPDY and HTTP/2. HttpWatch BlogHttpWatch Blog. Accessed October 15th, 2017. https://blog.httpwatch.com/2015/01/16/a-simple-performance-comparison-of-https-spdy-and-http2/
41. PageSpeed Tools. Google Developers. Accessed October 15th, 2017. https://developers.google.com/speed/pagespeed/
42. Mail - Laravel - The PHP Framework For Web Artisans. Taylor Otwell. Accessed October 15th, 2017. https://laravel.com/docs/5.1/mail#queueing-mail
43. A Framework and QoS Matchmaking Algorithm for Dynamic Web Services Selection. L Taher, H El Khatib & R Basha. Accessed October 15th, 2017. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.84.5780&rep=rep1&type=pdf
44. The Hospitals/Residents Problem (1962; Gale, Shapley). David F Manlove. Accessed October 15th, 2017. http://www.dcs.gla.ac.uk/publications/PAPERS/8632/hr.pdf
45. Website speed test. Pingdom Tools. Accessed October 31, 2017. https://tools.pingdom.com

11 Appendix

Trello



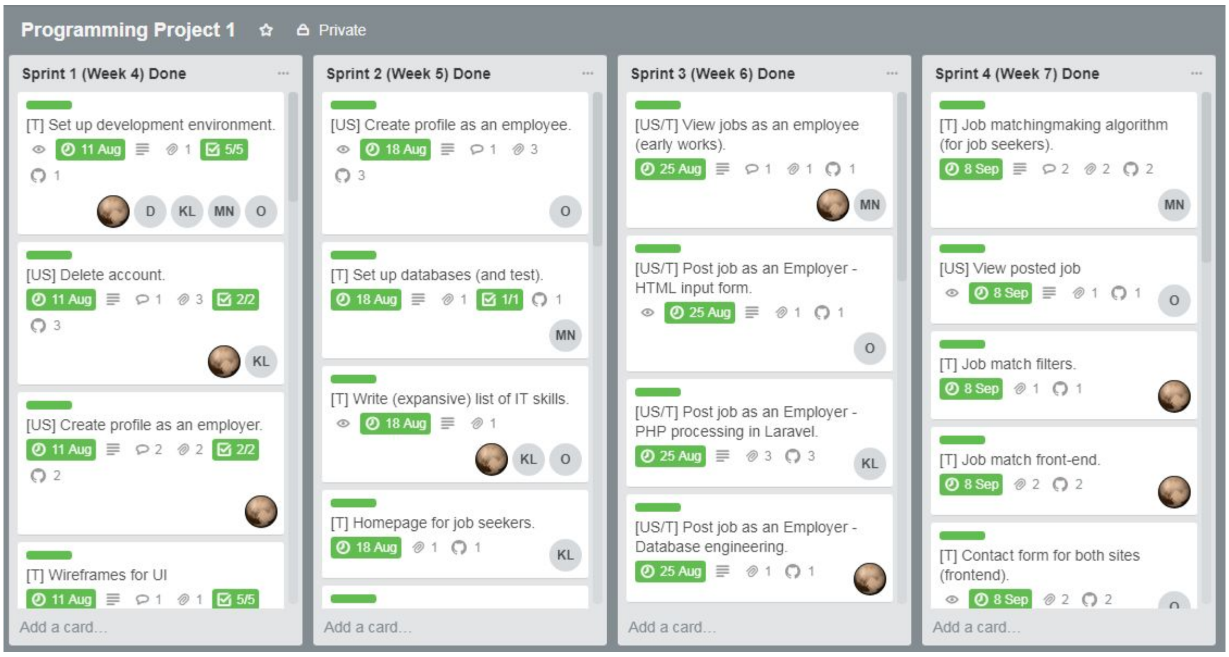
Programming Project 1 ☆ Private

Misc.
- Important Information — 1 — D KL MN O
- Meeting Minutes — 1 — 2 — D KL MN O
- Internal team meeting: Tuesday 2:30pm
- Tutorial meeting: Friday 12:30pm
- Team members and information
- Add a card...

Research
- Framework — 1 — 1 — 7/7 — 1
- Match-making algorithm — 1 — KL MN
- Technology — 1 — 1 — O
- Cloud server — 1 — MN
- Add a card...

Documentation
- Project Charter — 1 — KL O
- Project Timeline — 1 — 1 — D
- Individual Timesheets — 1 — 13 Oct — 5 — D KL MN O
- Add a card...

Tasks
- Week 3 Submission — 3 — D KL MN O
- Add a card...



Programming Project 1 ☆ Private

Sprint 1 (Week 4) Done
- [T] Set up development environment. — 11 Aug — 1 — 5/5 — 1 — D KL MN O
- [US] Delete account. — 11 Aug — 1 — 3 — 2/2 — 3 — KL
- [US] Create profile as an employer. — 11 Aug — 2 — 2 — 2/2 — 2
- [T] Wireframes for UI — 11 Aug — 1 — 1 — 5/5
- Add a card...

Sprint 2 (Week 5) Done
- [US] Create profile as an employee. — 18 Aug — 1 — 3 — 3 — O
- [T] Set up databases (and test). — 18 Aug — 1 — 1/1 — 1 — MN
- [T] Write (expansive) list of IT skills. — 18 Aug — 1 — KL O
- [T] Homepage for job seekers. — 18 Aug — 1 — 1 — KL
- Add a card...

Sprint 3 (Week 6) Done
- [US/T] View jobs as an employee (early works). — 25 Aug — 1 — 1 — 1 — MN
- [US/T] Post job as an Employer - HTML input form. — 25 Aug — 1 — 1 — O
- [US/T] Post job as an Employer - PHP processing in Laravel. — 25 Aug — 3 — 3 — KL
- [US/T] Post job as an Employer - Database engineering. — 25 Aug — 1 — 1
- Add a card...

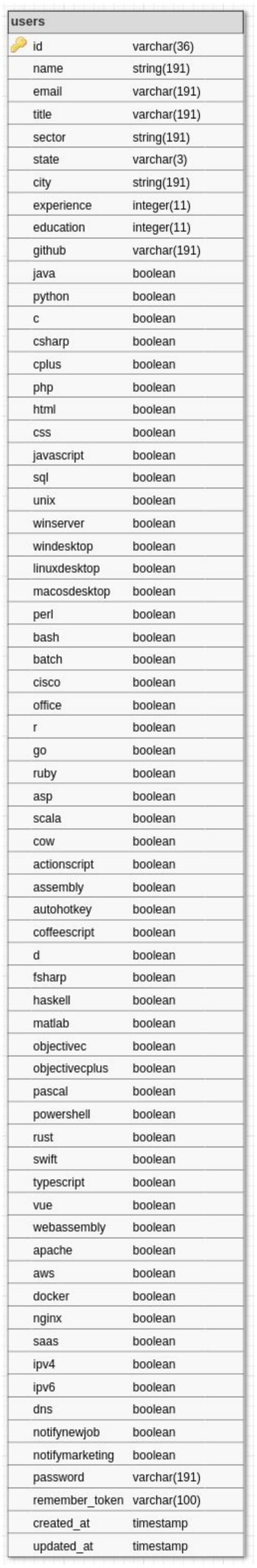
Sprint 4 (Week 7) Done
- [T] Job matchingmaking algorithm (for job seekers). — 8 Sep — 2 — 2 — 2 — MN
- [US] View posted job — 8 Sep — 1 — 1 — O
- [T] Job match filters. — 8 Sep — 1 — 1
- [T] Job match front-end. — 8 Sep — 2 — 2
- [T] Contact form for both sites (frontend). — 8 Sep — 2 — 2
- Add a card...

Trello is an important tool that was used to manage the project. By following Scrum processes, the project manager created stages for our tasks.

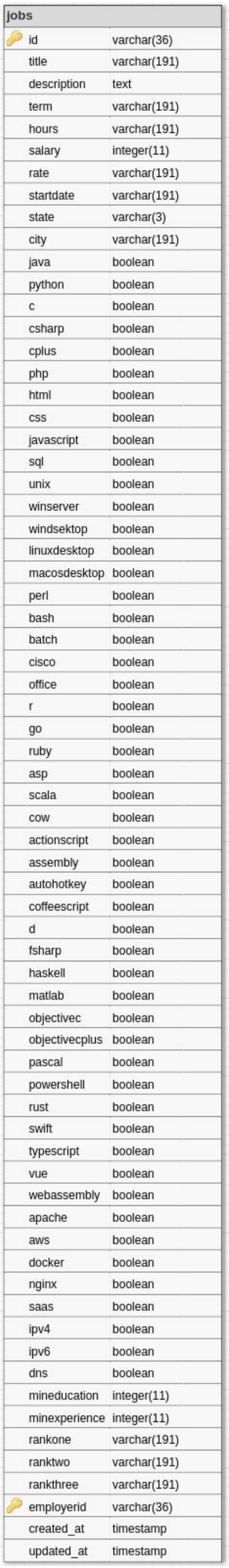
These stages included IN PROGRESS, and DONE. The list of tasks were placed in Sprints every week. These tasks were assigned to different team members. Tasks were also given due dates so that the functions could be completed on time.

As shown in the above image, once the tasks for the week were completed, the Sprints were relabeled - adding the word 'Done'.

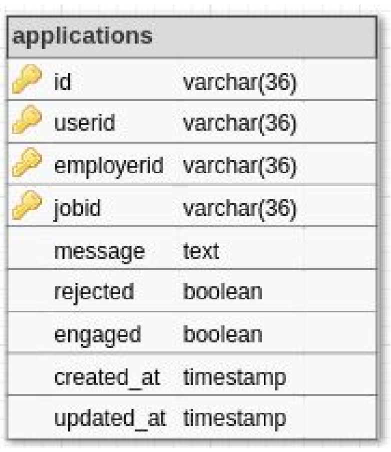
Database table schemas



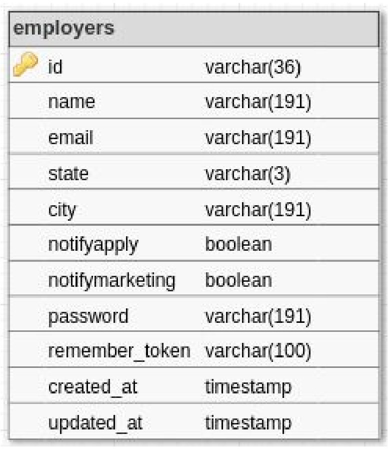
users	
id	varchar(36)
name	string(191)
email	varchar(191)
title	varchar(191)
sector	string(191)
state	varchar(3)
city	string(191)
experience	integer(11)
education	integer(11)
github	varchar(191)
java	boolean
python	boolean
c	boolean
csharp	boolean
cplus	boolean
php	boolean
html	boolean
css	boolean
javascript	boolean
sql	boolean
unix	boolean
winserver	boolean
windesktop	boolean
linuxdesktop	boolean
macosdesktop	boolean
perl	boolean
bash	boolean
batch	boolean
cisco	boolean
office	boolean
r	boolean
go	boolean
ruby	boolean
asp	boolean
scala	boolean
cow	boolean
actionscript	boolean
assembly	boolean
autohotkey	boolean
coffeescript	boolean
d	boolean
fsharp	boolean
haskell	boolean
matlab	boolean
objectivec	boolean
objectivecplus	boolean
pascal	boolean
powershell	boolean
rust	boolean
swift	boolean
typescript	boolean
vue	boolean
webassembly	boolean
apache	boolean
aws	boolean
docker	boolean
nginx	boolean
saas	boolean
ipv4	boolean
ipv6	boolean
dns	boolean
notifynewjob	boolean
notifymarketing	boolean
password	varchar(191)
remember_token	varchar(100)
created_at	timestamp
updated_at	timestamp



jobs		
🔑	id	varchar(36)
	title	varchar(191)
	description	text
	term	varchar(191)
	hours	varchar(191)
	salary	integer(11)
	rate	varchar(191)
	startdate	varchar(191)
	state	varchar(3)
	city	varchar(191)
	java	boolean
	python	boolean
	c	boolean
	csharp	boolean
	cplus	boolean
	php	boolean
	html	boolean
	css	boolean
	javascript	boolean
	sql	boolean
	unix	boolean
	winserver	boolean
	windsektop	boolean
	linuxdesktop	boolean
	macosdesktop	boolean
	perl	boolean
	bash	boolean
	batch	boolean
	cisco	boolean
	office	boolean
	r	boolean
	go	boolean
	ruby	boolean
	asp	boolean
	scala	boolean
	cow	boolean
	actionscript	boolean
	assembly	boolean
	autohotkey	boolean
	coffeescript	boolean
	d	boolean
	fsharp	boolean
	haskell	boolean
	matlab	boolean
	objectivec	boolean
	objectivecplus	boolean
	pascal	boolean
	powershell	boolean
	rust	boolean
	swift	boolean
	typescript	boolean
	vue	boolean
	webassembly	boolean
	apache	boolean
	aws	boolean
	docker	boolean
	nginx	boolean
	saas	boolean
	ipv4	boolean
	ipv6	boolean
	dns	boolean
	mineducation	integer(11)
	minexperience	integer(11)
	rankone	varchar(191)
	ranktwo	varchar(191)
	rankthree	varchar(191)
🔑	employerid	varchar(36)
	created_at	timestamp
	updated_at	timestamp



applications		
🔑	id	varchar(36)
🔑	userid	varchar(36)
🔑	employerid	varchar(36)
🔑	jobid	varchar(36)
	message	text
	rejected	boolean
	engaged	boolean
	created_at	timestamp
	updated_at	timestamp



employers		
🔑	id	varchar(36)
	name	varchar(191)
	email	varchar(191)
	state	varchar(3)
	city	varchar(191)
	notifyapply	boolean
	notifymarketing	boolean
	password	varchar(191)
	remember_token	varchar(100)
	created_at	timestamp
	updated_at	timestamp