

# COMP3900

## Capstone Project

Project: Recruit Assistant

Team: Kai-Will-Make-Tony-Rich

Class: Wednesday 5-7pm W17A (Ali)

Submission Date: 16<sup>th</sup> September 2020

Name	Email	zID	Role
Kaiqi Liang	kaiqi.liang@unsw.edu.au	z5210932	Scrum Master
William Huang	z5205986@unsw.edu.au	z5205986	Backend Developer
Tony Lu	z5204814@unsw.edu.au	z5204814	Backend Developer

<b>Richard Wang</b>	z5166088@unsw.edu.au	z5166088	Frontend Developer
---------------------	----------------------	----------	--------------------

### **3. Table of contents and page numbers.**

#### **Overview - Architecture / design of the overall system & functionalities - Page 3-6**

PL: Presentation Layer - Page 5

AL: Application Layer - Page 5

DL: Data Layer - Page 5-6

API Layer - Page 6

#### **Descriptions of the functionalities developed by the team and how they map/address all project objectives - Page 6-13**

Novelty Feature: Matching Employers and Job Seekers based on Skills - Page 8-10

Functionalities - Page 10-13

#### **Proper references and brief descriptions of ALL third-party functionalities (clouds/services/APIs/libraries/code) used by the team, with justification for their use and discussion how their licensing terms impact results of this project - Page 13-16**

Front End - Page 13-15

React.js - Page 13

Material UI - Page 13-14

Material-ui-search-bar - Page 14

Testing-library/jest-dom - Page 14

Testing-library/react - Page 14

Testing-library/user-event - Page 14

React-datepicker - Page 15

React-dom - Page 15

React-scripts - Page 15

Styled-components - Page 15

Backend - Page 16-18

CORS - Page 16

Express.js - Page 16

JSON Web Token - Page 16

SQLite3 - Page 16

#### **Implementation challenges: descriptions of any tricks, non-trivial algorithms, special architecture/design, etc. - Page 16-18**

#### **User documentation/manual: how to build, setup, configure, and use your system and functionalities - Page 18-21**

Installing GitHub Desktop/Git Bash - Page 18-19

Cloning the Project - Page 19

Install Node.js - Page 19

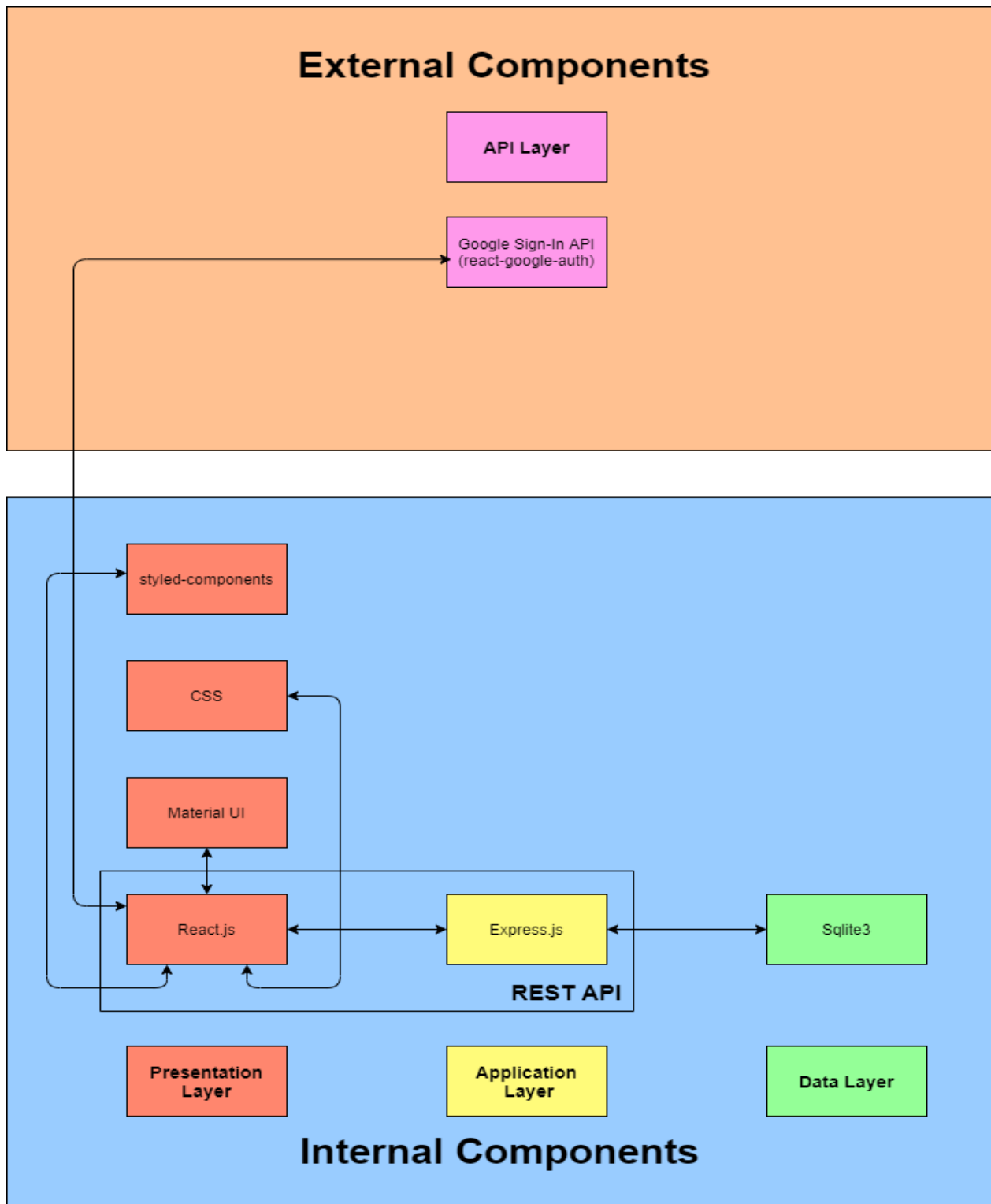
Install System Dependencies/Setup the Server - Page 19-21

#### **References - Page 21-23**

#### **4. Overview - Architecture / design of the overall system & functionalities.**

To design our overall system, we first contemplated on each component of the system and the relationships between these components and how they could be implemented code-wise. This was achieved through our ER-diagram, which can be seen below.





*Above: Diagram of the Overall Architecture*

A 3-tier architecture is a type of client-server architecture consisting of three tiers of logical computing, being the presentation, application and data layers. The modular design has the benefit of improving work efficiency as project teams have the ability to continually upgrade

individual tiers as new needs arise, all without impacting the rest of the system. It also means different developers can work independently and concurrently on tiers in different languages. Our project uses a 3-tier architecture to effectively utilise all members in our group. The diagram below shows the components of our software architecture.

We decided to change our website architecture slightly compared to our original proposal. This was in part due to the fact that we changed our novelty feature from job seekers applying to a general pool of jobs to a system which was based on matching job skills which were desired by the employer. Our reasoning behind this implementation change was that some jobs may actually be very similar, but due to different job listing names, they may not be identified as the same thing. For example, a job seeker could be applying for a software engineer role, but for example, a job listing with name 'C++ developer', whilst not being called 'software engineer', is a subset of that. On the contrary, the aforementioned problem would be fixed with skill matching as non-identical yet similar roles would also share similar skills.

### **PL: Presentation Layer**

The presentation layer is the top-most layer and displays all the information from the website in the form of a graphical user interface (GUI). It translates tasks and results to a user-readable format. Our project team built this layer for the final implementation with web development frameworks and technologies, in line with initial proposals. These include the JavaScript web framework React.js, style sheet language CSS and markup language HTML. We also used the React component libraries Material UI and Styled Components to complement our GUI and keep the design consistent.

### **AL: Application Layer**

The application layer, also called the business (logic) layer, takes user-inputted commands from the presentation layer and controls how the application and core functionalities are executed. It makes logical decisions, processes commands and performs calculations. It also acts as a gateway for data processing and movement between the presentation layer and data layer. Our team decided to change our initially proposed idea of programming this layer in Python to Javascript. This is because Javascript allows for greater convenience and versatility due to being able to be used for both front-end and back-end code, facilitating the smooth transition for programming between the two sides.

### **DL: Data Layer**

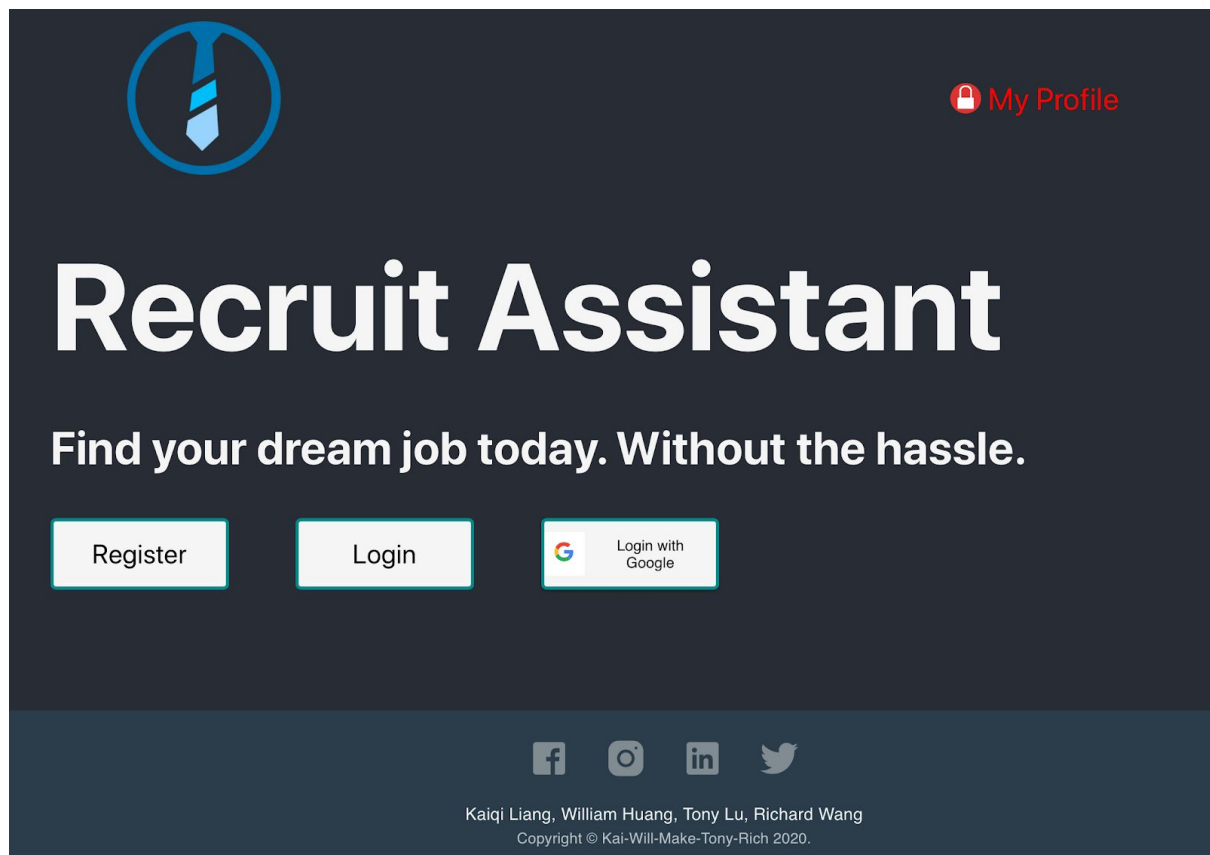
The data layer contains the Relational Database Management System (RDBMS) which stores data in a structured format. The RDBMS executes queries, which include creating tables, inserting, deleting and searching for values, allowing for easy data manipulation separate from the presentation and application layers. Our team used the lightweight, serverless SQLite3 as the RDBMS. A RDBMS such as MySQL and PostgreSQL requires client/server

architecture, therefore being way less portable and space-consuming than SQLite3. Our team also wanted the advantage of being able to store the database locally, which the other RDBMSs don't provide. We decided on using SQLite3 in the initial stages and kept it for the final implementation.

### API Layer

Our project team also uses APIs to complement our 3-tier architecture. APIs allow applications to access data from external services to add additional functionalities. Our project team decided that users sometimes do not want to have to remember multiple account passwords. The Google Sign-in API, an inbuilt component of React, was used so that in addition to being able to login via a user-created account on our website, a user also has the additional option of signing in with a Google account.

### **5. Descriptions of the functionalities developed by the team and how they map/address all project objectives**



*Above: Homepage of the Website*

**Sign Up**

Full Name  
Full Name

Email Address  
Email Address

Password  
Password

Confirm Password  
Confirm Password

☒ I'm an Employer   ☐ I'm looking for a job

[Already have an account? No worries, come login here](#)

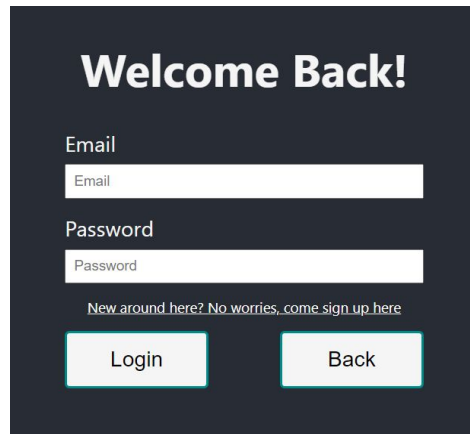
Register   Back

*Above: Sign Up Page*

Although not explicitly stated as a project objective, it is assumed that users will be able to register for an account as well as sign in and sign out of it after they have done so. Once a user enters the website, they will be met with its homepage. From here, they can either register for or login to an employer or job seeker's account. This can be done by pressing the 'register' or 'login' button, and filling out the form which subsequently appears. In the top right hand side of the website, the 'My Profile' button is also red and unable to be pressed as you must first sign in to your account before being able to view your profile.

When signing up, the form must be completely filled correctly and with no empty fields, which is implemented through constraint checking. The 'Name' field must be a full name consisting of first and last name, with a space between the two words to denote this. Next, the email address must also be a valid email address, with the constraint that the email address must contain '@%.com', where % is any text. In addition, the password must also be at least 3 characters, and the password must be successfully confirmed in the 'Confirm Password' field underneath. When all of these details have been verified, the type of account can be chosen by either pressing the 'I'm an Employer' or 'I'm looking for a job' radio button. If you have already signed up for an account, you can also press the 'Already have an account? No worries, come login here' link which will direct you to the sign in page. After this, pressing the 'Register' button will create an account which will be saved into the system's database so that you will be able to login.



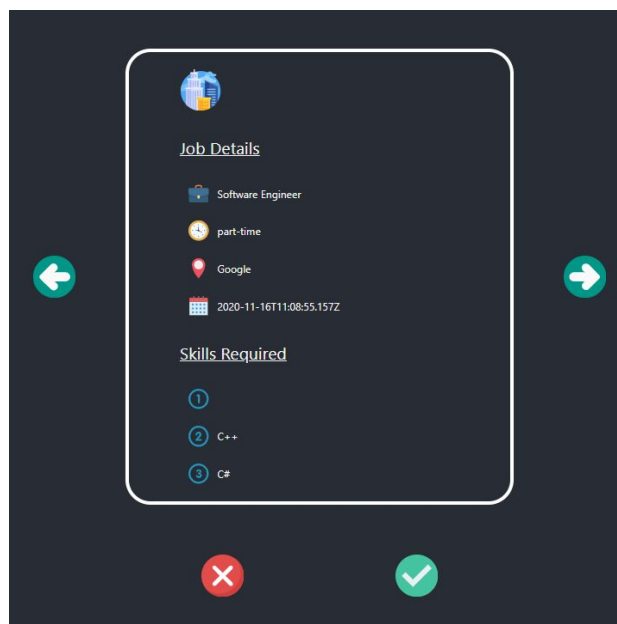


A dark-themed login page with the heading "Welcome Back!". It features two input fields: "Email" and "Password". Below the password field is a link that says "New around here? No worries, come sign up here". At the bottom are two buttons: "Login" and "Back".

*Above: Login Page*

Similar to signing up, logging into the website also requires both the 'Email' and 'Password' fields to be filled. After they have been filled, a user can press the 'Login' button, and if these details are correct, the user will be able to access all the website's functionality. From the home page, login can also be achieved with using the Google Account.

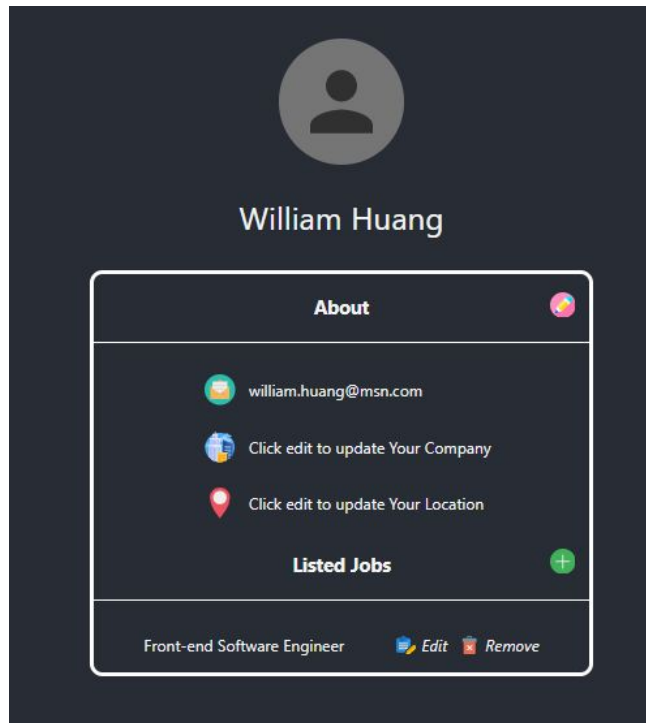
#### Novelty Feature: Matching Employers and Job Seekers based on Skills



A dark-themed job details page. At the top is a globe icon. Below it is the section "Job Details" with the following information: "Software Engineer", "part-time", "Google", and a date "2020-11-16T11:08:55.157Z". Below this is the section "Skills Required" with a list: 1, 2 C++, and 3 C#. At the bottom are two circular buttons: a red one with a white 'X' and a green one with a white checkmark. There are also green left and right arrow buttons on the sides of the central content area.

*Above: Job Details*

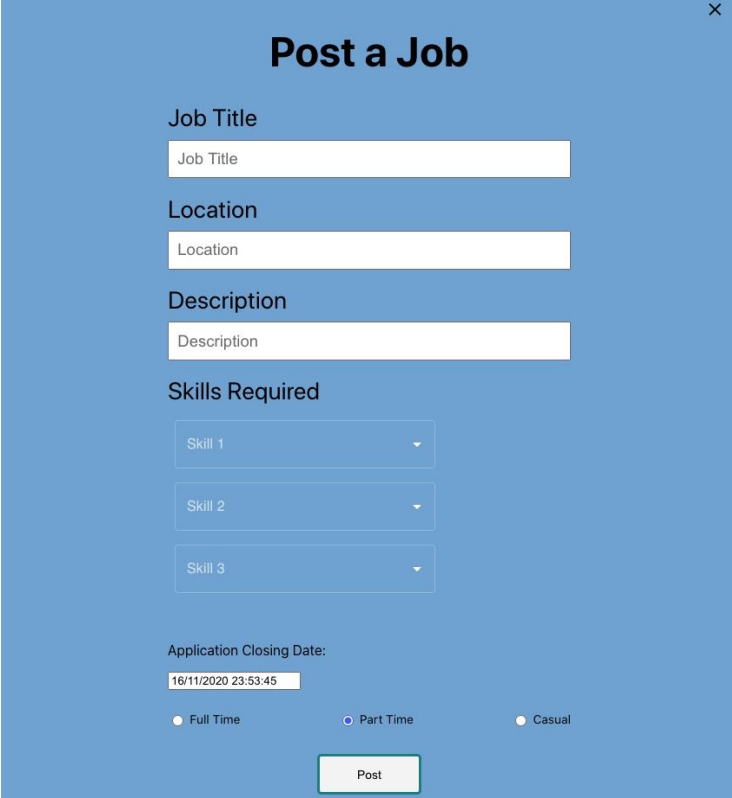
Although there are many websites which can assist job seekers for finding a job and employers for recruiting new employees, often this process can be quite cumbersome and



*Above: Employer Profile Page*

tedious as there are too many options, often unoptimised for the job seeker, which appear. A lot of the time job seekers see repeat jobs, and employers see repeat applications from the same job seekers, or applicants which are woefully unprepared for the job but submit anyway, which can lead to a large amount of wasted time. Our novelty feature removes all these complications. A job seeker will choose three skills to feature on their profile, and the application automatically finds the best job matches based on the number of skill matches between the user profile and the skills required for the job. Once a job seeker updates their jobs, the job finding simplification has begun. A job seeker's responsibility is to now go to 'Apply Now' and begin the swiping. Instead of tediously browsing jobs traditionally, the jobs essentially come to you and a job seeker just has to filter those. If a job seeker presses the 'right arrow' button or the 'tick button', they have the opportunity to be matched based on the condition that the employer does the same action for their respective application. If a job seeker does not express any interest in the job, they simply swipe left and do not have to worry about it again. From an employer's point of view, this process is largely similar. Instead of updating skills as a job seeker, an employer posts jobs that they wish to recruit people for. Within these jobs are several details such as the job title, a description, closing date but most importantly 3 skills that the job requires. These skills are the basis of our algorithm and determines if a match between a job seeker and employer is created. If the skills of a job seeker and a particular job match, then both the job seeker and employer will see the job posted and job seeker's profile respectively when either begins swiping. If both job seeker and employer swipe right (signalling interest) on each other's application then this constitutes a match. After a match, employer's can pursue a job seeker by requesting further

documentation such as resumes, cover letters and requesting interviews and ultimately a job offer if successful.

A screenshot of a web form titled "Post a Job" with a blue background and a white close button in the top right corner. The form contains several input fields: "Job Title" (text box), "Location" (text box), "Description" (text box), and "Skills Required" (three dropdown menus labeled "Skill 1", "Skill 2", and "Skill 3"). Below these is an "Application Closing Date:" field showing "16/11/2020 23:53:45". At the bottom, there are three radio buttons for "Full Time", "Part Time", and "Casual", with "Part Time" selected. A green "Post" button is at the very bottom.

*Above: Posting a Job*

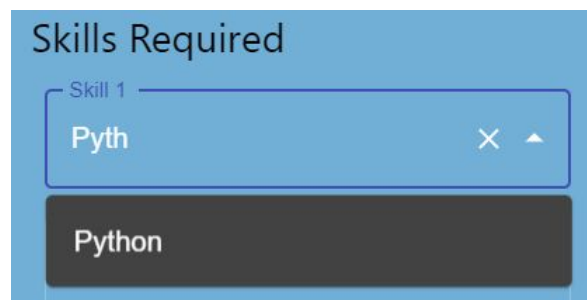
### Functionalities

While logged into the website, an employer can hover their cursor over the “My Profile” button to reveal a drop down menu with two options: “My Account” and “Log out”. After clicking “My Account”, they can see a section called “Listed Jobs” where they are able to click on the green circle with a plus sign, popping up a form which enables them to post a job. This form gives text boxes to fill in for job title, location and description. Within the description, employers can give an overview of the job, responsibilities, qualifications, experience, and remuneration details. In order to fit this information, the description text box has an adjustment on the bottom right corner which allows the employer to change the text box margins if necessary. The form allows three skills required to be selected from dropdown menus, further capturing the necessary qualifications for the job in a concise manner. For the application closing date, a pop up table revealing a calendar and a list of times separated by 30 minute intervals is shown, allowing the employer to not only show the day the application closes, but clearly show the time too as it can be confusing for job seekers who apply for a job on the closing date but miss out because the due time was undisclosed. At the bottom, there are three options for the employment type, where an employer can choose between full-time, part-time and casual. Finally, after clicking the large “Post” button, the website redirects the employer back to the “My Profile” page where they are able to see their new job listing.



*Above: List of all the Posted Jobs*

If both a job seeker and an employer press the ‘tick’ or press ‘right’ button on available jobs indicating their interest in each other, this will result in a match for both of them. At this stage, a job seeker has the option to upload additional documents such as a resume, cover letter, written references and other documents. This can be done from the dashboard.



*Above: Searching for a Skill using the search bar*

Our project team decided on an unconventional method to search for roles. Instead of having a traditional search bar to search for job roles, a user is able to find the most optimised jobs not based on job title, but based on skills when they update their profile. A job seeker can hover over ‘My Profile’ and click on ‘My Account’ to edit such skills. There are a total of 100 technical, business and design skills in which the job seeker can choose from, all searchable with a drop-down menu in combination with a search bar. This all ensures that the job being displayed is the most suitable possible job for the job seeker’s skillset, regardless of the job title, and if they do not want to apply they can ‘swipe’ left, showing the next suitable job. Our search functionality removes many of the drawbacks of major job application websites which require search terms to find the best jobs, as often employers can have multiple possible names for a job eg. searching for the term ‘software engineer’ can omit many entries for specific language-oriented development jobs such as ‘C++ developer’ as neither the term ‘software’ nor the term ‘engineer’ appears in the latter. Instead, if the job seeker chooses ‘C++’ as a skill, they will see both job openings ‘software engineer’ which has C++ as a required skill and ‘C++ developer’ as they all have the necessary skill match.

Job seekers submit an application by choosing the top 3 skills they feel strongest in. This can be done on the profile page for a job seeker. On this same page, job seekers can add in additional detail such as ‘education’ and ‘location’ which will be seen by the employers they swipe. Upon choosing their top 3 skills they are automatically added for consideration for open advertised positions (posted by employers) if at least 1 of their 3 skills match. This means when the employer visits the ‘Recruit Now’ page, they will eventually be able to see the job seeker’s application depending on how many other job seekers have mutual skills. The bare input requirement for employers is that there is at least 1 mutual skill however, all necessary qualifications and experiences are included for the employer to analyse when they are on the swiping interface at the ‘Recruit Now’ page.

Employers are able to re-order associated job applications by unmatching certain applicants and hence this changes the order whereby applicants appear on the screen. An employer can view all their posted jobs in the Matches section. From here, job seekers will be sorted based on how many skills they matched for the required skills for that particular job. In other words, job seekers who matched all three of their skills with the required ones for a particular job will be displayed first to the employer, followed by 2 matching skills and 1 matching skill. If there are no matching skills, a job seeker will not be displayed to an employer because there was no match to begin with. In addition, matches will also be sorted based on other factors such as qualifications and experience.

Applicant	Application	Stage	Action		Status
Kaiqi Liang	Software Engineer Google	Documentation	Proceed	Unmatch	Received (attachment)
Richard Wang	Atlassian Front End Engineer	Job Offer	Unmatch		Sent
William Huang	Deloitte Vacationer Program	Interview	Unmatch		Sent
Tony Lu	Quantum Summer Internship Program	Documentation	Request	Unmatch	Action Required

*Above: Status of all Applicants*

An employer can click on the “My Matches” option from the homepage to see all applicants who applied to their job advertisement. From there, they can click on any particular applicant, view their experience, resume, cover letter and skills. At the bottom are two options: “Offer Interview” and “Decline Applicant”, colour-coded in green and red respectively to reinforce the functionality.

Recruiter	Application	Stage	Action		Status
googlehr@gmail.com	Software Engineer at Google	Interview	Accept	Unmatch	Requested
atlassianhr@atlassian.com	Front End Engineer at Atlassian	Job Offer	Accept	Unmatch	Requested

*Above: Status of Applications*

Once an employer has sent a job offer to a job seeker, a job seeker can navigate to their dashboard. From here, they will have an option to view the status of all their job applications. Within the ‘stage’ column, a job seeker can see what part of the job application process they are up to. In the above diagram, the applicant can see that they have made it to the interview stage for the Google Job and have received an offer for the Atlassian job. Underneath ‘Action’, they can either accept or unmatched/decline the outcome, and the employer will be able to see this result after the applicant sends their action.

## **6. Proper references and brief descriptions of ALL third-party functionalities (clouds/services/APIs/libraries/code) used by the team, with justification for their use and discussion how their licensing terms impact results of this project**

Our system utilised many third-party resources to achieve its functionality. Below is a list of these resources and a brief description on them. Most of these third-party dependencies are non-restrictive for usage and thus, did not affect our capability to use them.

### **Front End**

#### **React.js**

React.js is a Javascript library which served as the main foundation for our front-end development, assisting in the creation of the website’s UI . It is mainly used for single-page applications and is simple and scalable. In addition, using React, web pages do not need to reload entirely whenever users interact with it but rather only the portions of the interface which actually need updating will update. Besides improving speed, this will also reduce resource usage. Many other functionalities which we use are also a part of React.js itself. React.js has a MIT license, allowing us to freely use it even for commercial purposes.

#### **Material UI**

Material UI is an open source user interface library in React that features components from Material, a Google-created design system. These Material Components are interactive building blocks for creating a UI. For the web, these include app bars, buttons, cards, data tables, dialogs, image lists, menus and text fields, among others. Material-UI’s components are based on these ideas, categorising them into layout, inputs, navigation, surfaces, feedback, data display, data grid, utils and lab. This enables a large variety of user interfaces to be built

depending on specifications and requirements. Our project uses Material-UI to create an accessible, clean user interface for our website. Material UI has a MIT license, allowing us to freely use it even for commercial purposes.

#### Material-ui-search-bar

The SearchBar is a UI component created by TeamWertarbyte openly accessible on github. It is based on Material UI's Text Field component, which allows users to enter text. It offers standard, filled and outlined stylisations, in which the SearchBar uses the outlined styling. The Material UI search bar has a MIT license, allowing us to freely use it even for commercial purposes.

#### Testing-library/jest-dom

jest-dom streamlines the process of asserting things about the state of a Document Object Model (DOM), whether that is checking for an element's attributes or CSS classes. By providing custom element matchers for Jest, it removes the need for unnecessary repetition, making our project team's development process much easier. The React Testing Library (jest-dom) has a MIT license, allowing us to freely use it even for commercial purposes.

#### Testing-library/react

The React Testing Library tests React components in a light-weight manner, realising the importance of tests resembling actual software usage. A major problem for writing tests is when a developer refactors code, they can break and have to be rewritten for each different implementation. By working with actual DOM nodes rather than instances of React components, the React Testing Library removes the need for tests to include implementation details and work as if a user were using the application. Our project team decided this was the best way to write software tests for React and hence used this library. The React Testing Library has a MIT license, allowing us to freely use it even for commercial purposes.

#### Testing-library/user-event

The React Testing Library (user-event) attempts to simulate the events which occur when users interact with the website. These include events such as clicking on a field, tabbing to a button as well as clicking it. By using this library, it allows for testing of a combination of related events rather than testing each event one at a time, which allows for more realistic and accurate testing of a system. The React Testing Library (user-event) has a MIT license, allowing us to freely use it even for commercial purposes.

### React-datepicker

The React-datepicker is a package within React which allows us to choose a date into a website. It appears as a toolbar, which if pressed, displays a calendar which allows you to choose both the day and time on that day. For our website, this was implemented so that employers can choose a closing time for a job which they are posting. The React-datepicker has a MIT license, allowing us to freely use it even for commercial purposes.

### React-dom

React-dom is a package within React which as its name suggests, serves as the Document Object Model (DOM). The DOM forms the foundation for a website's structure including its content and style. By representing the DOM as a logical tree of nodes and objects, programming languages are able to manipulate it and thus, change the website. Although there are many methods, the main one used within React-dom is the render method which allows content to be displayed onto the webpage. The React-dom has a MIT license, allowing us to freely use it even for commercial purposes.

### React-scripts

React-scripts is a set of scripts, or a list of instructions that is executed by another program, from the create-react-app starter pack which assists with initial configuration of the project so that manual configuration for initialising a project is not necessary. The main scripts include start, which start the server, test, which allows for testing of the system, build, which combines individual files into one and eject, which is required when making a complex system which does not use the default configuration so that other files can see normally hidden files. The React-scripts has a MIT license, allowing us to freely use it even for commercial purposes.

### Styled-components

Styled-components is an inbuilt library within React which grants flexibility by allowing users to manually write CSS for styling components. However, in combination with JavaScript, it also solves some traditional CSS problems too such as class name collisions. It solves this by targeting CSS to only one component rather than a global scope whilst also removing the necessity of HTTP requests to load resources. Styled-components has a MIT license, allowing us to freely use it even for commercial purposes.



## **Backend**

### CORS

CORS, or cross-origin resource sharing, is a HTTP mechanism which allows browsers to interact with and determine what ‘origins’ their resources are coming from. An origin is simply the remote repository where you initially obtained your files from. By using CORS, our system is able to utilise resources from other domains, schemes or ports, rather than just its own. CORS has a MIT license, allowing us to freely use it even for commercial purposes.

### Express.js

Express.js is a website application framework based on Node.js and was used for the backend of our system. It was used as it provides many features which greatly eased the web development process as many modules are compatible with Express. In addition, Express, which is based on Node.js and thus utilises Chrome’s V8 Javascript engine, is extremely responsive, which was another one of the reasons why we selected it. Express.js has a MIT license, allowing us to freely use it even for commercial purposes.

### JSON Web Token

JSON Web Token is used for securely transmitting and authorising information between two parties as a JSON object. Its specialty is that it is digitally signed through signed tokens, which means that a request can only be made if the token permits it, such as when logging in. In addition, these tokens can also be used with public and private keys, ensuring security when transmitting information. JSON Web Token has a MIT license, allowing us to freely use it even for commercial purposes.

### SQLite3

SQLite3 was used for managing, storing and collecting data for our system. While we also considered a close alternative RDBMS (PostgreSQL), we ultimately chose Sqlite3 due to being a file-based system, removing the necessity for a server which would complicate the design process. Sqlite3 also has a public domain license which means that it is available for us by everyone as it belongs to the public and not protected under any laws.

## **7. Implementation challenges: descriptions of any tricks, non-trivial algorithms, special architecture/design, etc.**

One of the problems that we faced when implementing the system was that we did not use any stage management tools or libraries. This was due to the high learning curve which it brought which our team unfortunately did not have the time for. The state of an application is simply what the application knows about itself and the world around it which ensures the

program runs properly. Being able to persist state is critical as unlike applications such as Windows applications, websites are stateless and thus after each HTTP request, no information on the website which can be utilised for the next HTTP request is saved. For example, the buttons on the front-end user interface should be enabled and the state of these buttons should be maintained even after refreshing the web page. To solve this issue, we utilised the React Hook `useContext`, which alleviates the problem of state management by storing all the necessary state data into an internal global state, similar to global variables. Although this is not the most ideal solution due to the nature of a global state having the possibility of randomly and suddenly changing unexpectedly due to interaction with other system components, using a Hook also has the advantage that they only use functions and not classes to retrieve the necessary state information.

In addition, as we continued to progress through our implementation development, we realised that our database design was insufficient to satisfy all the necessary requirements which adapted as the sprints progressed. This was most likely due to an in-depth but not fully complete understanding of the problem domain, resulting in us to change and add more tables into our relational schema. For example, our initial proposal did not have a 'Skills' table as we initially planned to add all the skills into one text attribute within the 'JobSeekers' table. However, we realised that obtaining the value of each skill within this text attribute would prove challenging due to variations of how a value was inputted by a user, thus leading us to create this new 'Skills' table.

Another problem which we faced was an inconsistent data representation during the implementation process. Although sometimes coding was done together as a team, most of the time each team member was assigned and worked on their own task, and thus, this lack of communication during these time periods would sometimes cause 'spaghetti code'. For example, although the front-end was attempting to pass in 'null' into the database, the backend registered this as a string of value 'null' rather than the 'null' constant. These errors also became apparent as individual changes were merged together during the testing phase. During the testing phases, we fixed these inconsistencies together as a team.

In addition, we could not find an API that met our requirements for the 'Skills' table. This was due to the fact that these online APIs contained too many skills (hundreds and thousands) which would thus make skill matching quite impractical and infeasible during the implementation stage for the 'Matches' table. In addition, using some of these APIs, such as the LinkedIn one, also required being registered under a company before being able to be used. Resultantly, we decided to single out the most important skills, which focused on the technical field, such as knowledge of programming languages, as well as design skills, such as knowledge of Photoshop, from various sources online and manually add this into our 'Skills' table. Although this is not the most effective implementation, it captures the majority of the scenarios of what users would most likely input when using our system and highlights our target audience of startups, which are mainly in the tech, business and design sector.

Finally, our team needed to build a matching algorithm which updated the PotentialJobs table every time a job seeker updated their featured skills. This proved quite a difficult task as initially it was allocated to one team member who could not complete it himself. Only after working as a group on devising an algorithm, then a solution could be found. Our team built an algorithm in SQL where when a job seeker updates a particular skill, all rows associated with the previous skill in the PotentialJobs table and no longer match with any job seeker skill are deleted. Then, a loop goes through all entries in the Jobs table, finds the jobs which are associated with the newly updated skill and those are added to the PotentialJobs table. A record of the number of skill matches for each job in the PotentialJobs table is also kept.

## **8. User documentation/manual: how to build, setup, configure, and use your system and functionalities**

Before our system can be run, it is essential that all the necessary software and modules are installed and configured correctly beforehand. Our system is comprised of three main components:

1. Front-end (React.js) - This provides users with an interface which they can interact with for our website.
2. Back-end (Node.js/Express.js) - This interacts with the frontend for processing data and providing database access for our website.
3. Database (SQLite3) - This interacts with the backend for storage and retrieval of data.

Below are instructions on how to set-up these components.

*Note: The attached screenshots use Command Prompt in Windows, but other alternatives are also usable such as using the inbuilt terminal within an IDE like Visual Studio Code.*

### **Installing GitHub Desktop/Git Bash**

Before our system can be run, its files must first be obtained and stored locally on your device. This can be done by cloning the Git project from GitHub where our repository is stored. There are two programs which can be used to perform this depending on personal preference: GitHub Desktop or Git Bash, which both effectively have the same functionality, but GitHub Desktop provides a user interface whereas Git Bash is a terminal.

To install GitHub Desktop, navigate to:

<https://desktop.github.com/>.

To install Git Bash, navigate to:

<https://git-scm.com/downloads>

Choose the correct installer depending on your system and open the installer, following the instructions to install the software.

To ensure that it has installed successfully, enter the command 'git --version' within your terminal. If successful, a message similar to that below should appear.

```
C:\Users\William\Documents\GitHub>git --version
git version 2.28.0.windows.1
```

### Cloning the Project

Before the repository files can be stored locally onto your device, you must first setup the SSH key. The instructions to do this can be accessed via:

<https://docs.github.com/en/github/authenticating-to-github/adding-a-new-ssh-key-to-your-github-account>.

Finally, within the terminal, perform a 'git clone' operation by entering the command shown below:

```
C:\Users\William\Documents\GitHub>git clone git@github.com:unsw-cse-capstone-project/capstone-project-comp3900-w17a-kai-will-make-tony-rich.git
```

### Install Node.js

The next step is to install Node.js. This is essential as by doing so, the NodeJS Package Manager (or npm for short) becomes available for installing the other necessary packages. To do this, open your browser and navigate to:

<https://nodejs.org/en/download/>.

Download and open the appropriate installer depending on your system and follow the instructions to install Node.js. Our system was built using Node.js v14 so we highly recommend you to download this version to ensure everything will be compatible with our system. To ensure that Node.js installed successfully, enter the command 'node' within the terminal. A message similar to the one below will be displayed.

```
C:\Users\William\Documents\GitHub\capstone-project-comp3900-w17a-kai-will-make-tony-rich\backend>node
Welcome to Node.js v14.13.1.
Type ".help" for more information.
>
```

### Install System Dependencies/Setup the Server

To run the server, navigate to the 'backend' directory where the 'server.js' file is located. However, before we can do this, we must first install all the backend dependencies for the system which are all already predefined in the package.json file within the directory. This is a one-time action necessary when setting up the system for the first time, with these

dependencies continuing to exist even after the terminal has been closed for the next time of usage. By entering the command ‘npm install’, all these defined dependencies are resultantly installed.

```
C:\Users\William\Documents\GitHub\capstone-project-comp3900-w17a-kai-will-make-tony-rich\backend>npm install
```

Once all of this is done, enter the command ‘node server.js’. If this is successful, a message “Server running at [http://localhost:\\$port](http://localhost:$port)” along with “Connected to sqlite database” will be printed to the console, where port can take any available value defaulting at 8000, or otherwise, an error message will be printed if a connection cannot be established. Usually, an error occurs if the port is already in use, in which case ensuring the port is free such as by restarting all terminals should be the next step.

```
C:\Users\William\Documents\GitHub\capstone-project-comp3900-w17a-kai-will-make-tony-rich\backend>node server.js
Server running at http://localhost:8000
Connected to sqlite database
```

After this, navigate to the ‘frontend’ directory, and enter the command ‘npm install’, which similar to above, will install all the necessary frontend dependencies for the system.

```
C:\Users\William\Documents\GitHub\capstone-project-comp3900-w17a-kai-will-make-tony-rich\frontend>npm install
```

Finally, enter the command ‘npm start’, which will start the website automatically within the browser. Otherwise, the website can also be accessed by manually entering the URL <http://localhost:port> in the browser. After the first time, the two ‘npm install’ commands are unnecessary for setting up the server. Instead, all that is necessary is:

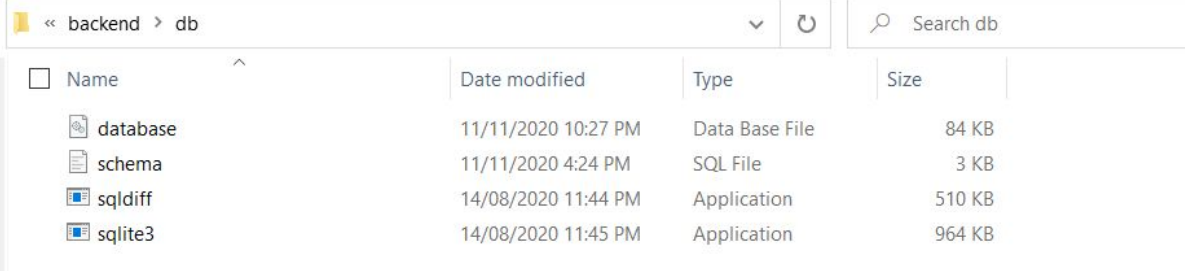
1. Navigate to the ‘backend’ directory and run ‘node server.js’.
2. Navigate to the ‘frontend’ directory and run ‘npm start’.





```
C:\Users\William\Documents\GitHub\capstone-project-comp3900-w17a-kai-will-make-tony-rich\frontend>npm start
```

In addition, installation of these modules within the terminal may sometimes be unsuccessful. In these situations, you should install these modules locally, following a procedure similar to that of installing Node.js above but afterwards, copying the module files into the appropriate directory depending on what type of module it is.

For example, to install Sqlite3 locally, open a browser and navigate to:  
<https://www.sqlite.org/download.html>.

Choose the appropriate installer and download and install as necessary. As Sqlite3 is a database module, the module files should then be copied manually into the 'backend' directory, and for Sqlite3 specifically, into the 'backend/db'. This will look as follows:



« backend » db					Search db	
<input type="checkbox"/> Name	Date modified	Type	Size			
 database	11/11/2020 10:27 PM	Data Base File	84 KB			
 schema	11/11/2020 4:24 PM	SQL File	3 KB			
 sqldiff	14/08/2020 11:44 PM	Application	510 KB			
 sqlite3	14/08/2020 11:45 PM	Application	964 KB			

It should also be noted, however, that necessary dependencies, no matter how they are installed, are not necessary to be pushed with GitHub.

Now, you should be able to access and use the website. If for some reason there are still any problems, common troubleshooting methods such as restarting your browser and terminals again should be attempted. Running the command 'npm run reset' will also reset the system which may remove any existing, hidden problems.

## **References**

jinfonet.com. (n.d.). 3-tier Architecture: A Complete Overview. [online] Available at: <https://www.jinfonet.com/resources/bi-defined/3-tier-architecture-complete-overview/> [Accessed 16 November 2020].

Rouse M. (n.d.). What Is a 3-tier Application Architecture?. - Definition from Whatis.com. [online] Available at: <https://searchsoftwarequality.techtarget.com/definition/3-tier-application> [Accessed 16 November 2020].

Morris S. (n.d.) React Js-what Is It? What Is Used For? Why Should You Learn It?. [online] Available at: <https://skillcrush.com/blog/what-is-react-js/> [Accessed 16 November 2020].

freecodecamp.org. 2018. Meet Material-ui - Your New Favorite User Interface Library. [online] Available at: <https://www.freecodecamp.org/news/meet-your-material-ui-your-new-favorite-user-interface-library-6349a1c88a8c/> [Accessed 16 November 2020].

material-ui.com. 2020. Material-Ui: A Popular React Ui Framework. [online] Available at: <https://material-ui.com/> [Accessed 16 November 2020].

TeamWertarbyte. 2020. Teamwertarbyte/material-ui-search-bar. [online] Available at: <https://github.com/TeamWertarbyte/material-ui-search-bar> [Accessed 16 November 2020].

material-ui.com. 2020. Text Field React Component - Material-ui. [online] Available at: <https://material-ui.com/components/text-fields/> [Accessed 16 November 2020].

GitHub, Inc. 2020. Testing-library/jest-dom. [online] Available at: <https://github.com/testing-library/jest-dom> [Accessed 16 November 2020].

Borenkraout M. 2020. React Testing Library | Testing Library. [online] Available at: <https://testing-library.com/docs/react-testing-library/intro/> [Accessed 16 November 2020].

Adobe Inc. 2020. Testing - React Spectrum. [online] Available at: <https://react-spectrum.adobe.com/react-spectrum/testing.html> [Accessed 16 November 2020].

npmjs.com. 2020. react-date-picker-npm. [online] Available at: <https://www.npmjs.com/package/react-date-picker> [Accessed 16 November 2020].

reactjs.org. 2020. ReactDOM - React. [online] Available at: <https://reactjs.org/docs/react-dom.html> [Accessed 16 November 2020].

Mozilla. 2020. Document Object Model (DOM)- Web APIs | MDN. [online] Available at: [https://developer.mozilla.org/en-US/docs/Web/API/Document\\_Object\\_Model](https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model) [Accessed 16 November 2020].

Ndaw I. 2020. Everything you need to know about react-scripts- LogRocket Blog. [online] Available at: <https://blog.logrocket.com/everything-you-need-to-know-about-react-scripts/> [Accessed 16 November 2020].

Lukman A. 2020. How To Use Styled-components In React - Smashing Magazine. [online] Available at: <https://www.smashingmagazine.com/2020/07/styled-components-react/> [Accessed 16 November 2020].

Hobbs S. 2019. Cors Tutorial: A Guide To Cross-origin Resource Sharing. [online] Available at: <https://auth0.com/blog/cors-tutorial-a-guide-to-cross-origin-resource-sharing/> [Accessed 16 November 2020].

Agarwal R. 2014. Why Use Expressjs Over Nodejs For Server-side Development?. [online] Available at: <https://www.algoworks.com/blog/why-use-expressjs-over-nodejs-for-server-side-coding/> [Accessed 16 November 2020].

jwt.io. 2020. JSON Web Token Introduction. [online] Available at:  
<https://jwt.io/introduction/> [Accessed 16 November 2020].

sqlite.org. 2020. Appropriate Uses for SQLite. [online] Available at:  
<https://www.sqlite.org/whentouse.html#:~:text=They%20emphasize%20scalability%2C%20concurrency%2C%20centralization,compete%20with%20client%2Fserver%20databases.>  
[Accessed 16 November 2020].

GitHub, Inc. 2020. GitHub Desktop | Simple collaboration from your desktop. [online]  
Available at:  
<https://desktop.github.com/>. [Accessed 16 November 2020].

git-scm.com. 2020. Git - Downloads. [online] Available at:  
<https://git-scm.com/downloads> [Accessed 16 November 2020].

GitHub, Inc. 2020. Adding a New Ssh Key To Your Github Account- GitHub Docs. [online]  
Available at:  
<https://docs.github.com/en/github/authenticating-to-github/adding-a-new-ssh-key-to-your-github-account> [Accessed 16 November 2020].

sqlite.org. 2020. SQLite Download Page. [online] Available at:  
<https://www.sqlite.org/download.html>. [Accessed 16 November 2020].

Duffy J. (n.d.) State Management. [online] Available at:  
<https://www.codemag.com/Article/0409061/State-Management>. [Accessed 16 November 2020].