COMP9900: Information Technology Project

UNSW
SYDNEY

Final Report of Project

The 'skills backpack': e-portfolios for learning

Group Name:
Amazing team

Group members:
Programming Master**:**
Weijie Xiang z5143263 z5143263@ad.unsw.edu.au

Scrum Master / Developer:
Yuyao Guo z5040672 z5040672@ad.unsw.edu.au
Yu Niu z5124321 z5124321@ad.unsw.edu.au
Yinglun Qian z5081383 z5081383@ad.unsw.edu.au

Submission date:
21/10/2018

Content

1. Overview

1.1 Introduction

With the significant development of economy and high-technology, more and more jobs with specific technical requirements appear in the society. At the same time, more and more students who just graduated from university are looking for the jobs which matching their major they studied. As a result, both employer and employee have the demand in the personal market. However, there is a problem that how to help them find content that meets their requirements (Granovetter, 1995). Nowadays, there are already many job search websites on the internet, and some of them are very popular such as Seek and Indeed. These websites provide the following elements: keyword, classification, work type and location.

Different from the traditional job search website, our group project aims to set up an integrated system with more efficiency and accuracy. We establish a specific technical skill proficiency ranking system from 1 to 5. The employee could demonstrate each particular skill and proficiency which they have in their personal profile. The employer could show their specific requirement according to the position which they posted.

1.2 Main Requirements

Our project is requested to design for three different types of users which are candidates, prospective employer and admin.

1. Candidate:
   The candidate could build and create an e-portfolio representing their employability skills. Also, the candidate could find the job which they are looking for.

2. Prospective employer:
   The employer users could search the database of existing data from a candidate and get the result list according to the relevancy of skills.

3. Admin:
   Admin could respond a report to employers when finding a same or relevant requirement between candidate and employer.

1.3 Key Components

The whole system consists of 4 major parts including front-end, back-end, database and searching algorithm.

- **Front End**
  Front end structure is divided by separate by different role in the system. Since the angular cli is supporting for the component designing, which means different roles could have different component for supporting their functions.

  The main roles in this project is employer, employee, admin and visitor. Users could update their individual technical skills, find the related position, apply for job applications, find information by recommend system and get the feedback from admin. Employer users could post job position, check the information that already posted and reply the application by email.

- **Back End**
  Candidate skill proficiency store as an array in the database. It is represented by 1 to 5 to indicate the different proficiency.

  It is same as the front-end that divide into two parts. One is for the different users, and another is for job information. Also, this part could store the data for the whole system.

- **Database**
  The database divides into two parts. One is for storing the information of candidate; the other one is for storing the information of the employer.

- **Recommendation Algorithm**
  For matching the relevant specific skill between employee and employer, we use the K-Nearest Neighbour (KNN) algorithm with variations. In order to correspond with our demand, we add weight to the KNN algorithm.

1.4 Architecture

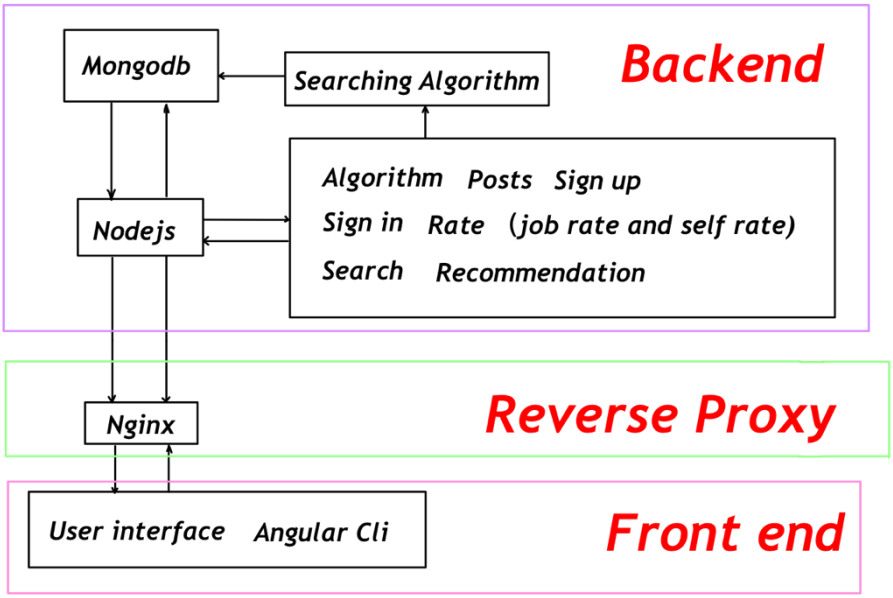

Figure 1 Architecture of the system

Figure 1 shows the architecture of the integrated system.

- **Front End**
  The front end has several webpages for the candidate, employer and admin. Candidate users could update their individual technical skills, find the related position, apply for job applications, find information by recommend system and get the feedback from admin. Employer users could post job position, check the information that already posted and reply the application by email.

- **Back-end/Database**
  Candidate skill proficiency store as an array in the database. It is represented by 1 to 5 to indicate the different proficiency.

  It is same as the front-end that divide into two parts. One is for the different users, and another is for job information. Also, this part could store the data for the whole system.

  The various job information stores by different data types. The name and description are stored as a string. The keywords for jobs are stored as a list. The requirements are stored as a dictionary.

- **Recommendation Algorithm**
  After collecting plenty of recruitment information which already exists in current

job searching website, we design a particular scoring system and weight system for different position requirements.

2. Functionalities

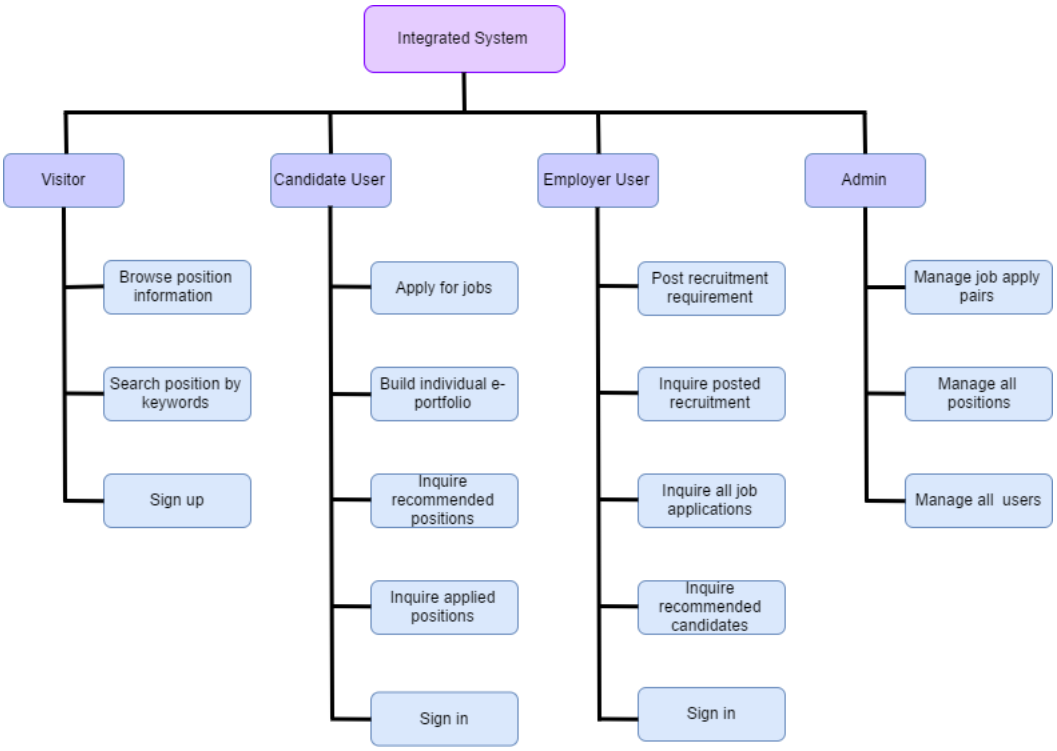

Figure 2 Functions of Project

Figure 2 shows the functions of our project.

2.1 Visitor

Users can utilize our website as visitors and browse all recruitment information at the 'all position' page.

Furthermore, as a visitor, users can search for interested recruitment information by keywords at the home page of the website. Then, the search engine will return all related position information at a new page.

The limitation of the visitor is that they could not apply for jobs directly and need to sign up.

2.2 Sign in/up

- **Sign up**
  Users need to sign up by their available email address, create a password and select a user identity from employee and employer.

- **Sign in**
  Sign in by registered the email address and password.

2.3 Candidate (Employee)

- **Search by keywords**
  The type of candidate (employee) users could search for favoured jobs by keywords at the home page of the website. The system will return the list of all related jobs in a new page and user can apply for jobs directly on this page.

- **Build and curate an e-portfolio**
  Users can build and curate an e-portfolio at candidate (employee) user page by inputting and updating mastered skills and corresponding skill level.

- **Job recommendation**
  The recommendation system will match and recommend suitable positions according to the skills and skill level mentioned in the user e-portfolio. Furthermore, users can view detail information of the recommended jobs and directly apply.

- **Job application inquiry**
  Candidate (employee) users can enquire the jobs they have applied.

2.4 Prospective employer

- **Post recruitment requirement**
  Prospective employer user can post recruitment requirements which contain required skills, corresponding skill level and other detail description.

- **Posted recruitment inquiry**
  Prospective employer user can check, update and delete posted recruitment information.

- **All application inquiry**

The system will display all applications for each posted job.

- **Candidate recommendation**
  The system will return the top three candidates who are matched by the recommendation algorithm.

2.5 Admin

- **Posted position management**
  Admin user can manage all recruitment information, such as the detail information of the position and the recommended candidates.

- **Prospective employer user management**
  Admin user can manage all information of Prospective employer users, such as the registration information and posted position information.

- **Candidate (employee) user management**
  Admin user can manage all candidate (employee) users' information, such as the registration information and the recommended positions information.

- **Notification management**
  If the Prospective employer users find candidates who may be suitable for the position, they can send a notification to the administrator, after receiving the notification, admin users operate according to the management requirements, such as noticing candidate (employee) users to attend the interview or sending admission to candidate (employee) users.

3. Third-party Functionalities

Throughout the project, third-party features used by our team include Angular CLI, Node.js, MongoDB, Beautiful Soup. This part of the report will briefly introduce and explain these third-party functions used by our group in the project, as well as how we use these third-party functionalities correctly.

3.4 Angular 4

Angular 4

- **Introduction**

  Angular is a front-end development platform. It can help developers to build web applications more easily. Angular combines declarative templates, dependency injection, end-to-end tools and the practical ability to solve development-level challenges (Angular.io). It can improve the ability to build web, mobile or desktop applications for developers.

  One of Angular's advantages is that Angular splits a web application into one component, and each component can be a smaller component (Angular.io). This method is very common in modern front-end development frameworks. The advantage is that a function is a component, independent of each other, it is convenient to modify and replace, and it is convenient to maintain and expand in the future.

- **Angular CLI**

  At the development level, we chose to use Angular CLI 6, which is an integrated development environment that comes with Angular. Using this tool can not only make our development more convenient but also provide us with many useful modules in web pack.

- **Http Client Module**

```
class HttpClientModule {
}
```

Http Client Module

We use the Http Client Module to Perform HTTP requests. *HttpClient* is available as an injectable class, with methods to perform HTTP requests. Each request method has multiple signatures, and the return type varies according to which signature is called (mainly are the values of *observe* and *responseType*) (Angular's API). We use this module to implement the functions that students and employers can access and publish on the website. Specifically, students and employers can interact with information on the site, including acquisition, publishing and deletion through this module.

3.5 Node.js

Node.js

- **Introduction**

  Node.js is a platform built on the Chrome JavaScript runtime (Node.js.org). Node.js is an event-driven I/O server-side JavaScript environment. Based on Google's V8 engine, the V8 engine executes JavaScript fast and performs well. One of the features of Node.js is asynchronous I/O and event-driven. For high-concurrency solutions, Node.js using a single-threaded model instead of the traditional multi-threaded model for all I/O requests.

- **Node.js Express Framework**

  Express is a simple and flexible node.js web application framework that provides a set of powerful features to help developers create a variety of web applications and rich HTTP tools (Express Nodejs). Use Express to quickly build a full-featured website.

  Core features of the Express framework:

  ✓ It can set up middleware to respond to HTTP requests.
  ✓ It defines a routing table for performing different HTTP request actions.
  ✓ It can dynamically render HTML pages by passing parameters to the template.

  We choose this framework to implement different functions of the backend by importing different modules.

- **Node.js Express: Body parser**

  Body parser is a very common express middleware, the role is to parse the request body of the post request (body-parser of Express Node.js README File). In our program, this function we used to convert the acquired information into a unified format for parsing.

- **Node.js Express: CORS**

  CORS is a W3C standard, the full name is "Cross-origin resource sharing". It allows the browser to issue XMLHttpRequest requests to cross-origin servers, overcoming the limitations of AJAX's only homologous use (cors of Express

Node.js). Currently, all browsers support this feature, and IE cannot be lower than IE10. When a resource requests a resource from a different domain or port than the server on which the resource itself resides, the resource initiates a cross-domain HTTP request. For security reasons, the browser restricts cross-origin HTTP requests originating from within the script, or it may be that cross-site requests can be initiated normally, but the results are intercepted by the browser. This means that web applications that use these APIs can only request HTTP resources from the same domain that loads the application, unless a CORS header file is used.

This is one of the main aspects of our project. In order to achieve cross-domain of different pages, CORS supports this part of the project.

3.6 Mongo DB

MongoDB

- **Instruction**

  MongoDB is an open source database system based on distributed file storage. In the case of high load, it can add more nodes to guarantee server performance. MongoDB is designed to provide a scalable, high-performance data storage solution for web applications (MongoDB Docs).

  MongoDB stores the data as a document, and the data structure consists of key-value (key=>value) pairs. A MongoDB document is similar to a JSON object. Field values can contain other documents, arrays, and document arrays.

- **Database in the project**

  We chose MongoDB because it is a database for document storage that is simple and easy to operate. Developers can create data mirrors either locally or on the network, which makes MongoDB more scalable. We have stored three databases in this project, which are “student”, “employer” and “jobs”. The reason for this is not only to make the data easier to store, but also to speed up the database loading, retrieve data faster, and increase the query speed.

3.7 Beautiful Soup 4

- **Instruction**

Beautiful Soup 4 is an HTML/XML parser whose main function is to parse and extract HTML/XML data (Beautiful Soup 4.2.0 Documentation). Beautiful Soup automatically converts input documents to Unicode encoding and output documents to UTF-8 encoding, which eliminates the need for developers to consider encoding.

- **Data Crawling**
  The main information in our current database comes from the Internet, so we need to get this information. Thus, we chose Beautiful Soup 4 to capture the information we need on the web, such as the type of work, the skills required for the job. Beautiful Soup 4 enabled our project to get the specific information we needed more accurately and quickly in data crawling. At the same time, it saves us time and reduces the amount of code to a certain extent.

4. Challenge

4.1 Challenge of collecting existing data from a website

The main purpose of collecting existing data from our existing job search website is to build our database. To achieve this, we use a web crawler which is called beautiful soup. However, the information that we collect from the internet is very complicated and confusing. It is a difficult challenge to extract the data we need from these.

This is due to the following factors. Firstly, there is no standard format for each website and employer. Hence, different website and different employer maybe describe the same requirement in different style. Secondly, there is no clear label for different skills. For example, the HTML developer could be described as web developer, front-end engineer or internet engineer.

According to our knowledge, we decided to focus only on the IT field. Base on the basic major knowledge, we could accurately express the label of professional skills. At the same time, we could match each specific label and position correctly. For instance, in our database, CSS and JavaScript could be a label for HTML developer.

In order to solve these problems, we established a keyword library for IT industry professional skills and extracted the skills keywords from the crawling recruitment information.

4.2 Challenge of Recommend Algorithm Choice

We choose K-Nearest Neighbour (KNN) as our recommendation algorithm.

To achieve an efficient and effective recommendation system, it is important to get the relationship between employability skills and position requirement. According to our design, the employer users could post the required employability skills, and the candidate users could update the employability skills they mastered. The difficulty is defining the level of employability skills.

Among the machine learning and deep learning algorithms, we first consider that the recruitment information we collected from existing websites do not have labels and is difficult to set suitable labels for them. Therefore, we consider the unsupervised learning algorithm. We tried semantic analysis on recruitment information, but it is failing to give us a specific correspondence between a position to skill level.

In consequence, we decide to let employer users and candidate users post skill level by themselves. We provide them with the introduction of skill level and what we need to solve is matching by the KNN algorithm.

4.3 Microservice

In the project, each functionality is used as functionality as a light-weight and fast way of setting up the project. Currently, most of the websites are required for different functionalities, fully integrated service design will bring negative effects during updating the functions for the whole system. In the programming designing, we separate all functionalities into different routing and interface for avoiding interfering.

4.4 Architecture design.

This is a hard issue in all programming designing and it is always solved in different ways. In this project, the hard issue is to integrate all functions into one interface file with a clear structure and appropriately designed. Since different roles, including admin, employee, employer, and visitor, in the project, the difficulty level increases significantly. Also, the searching speed and the user experience need to care about by developers simultaneously. So, we redesign the structure and database. For example, jobs are separated into a different collection in the database, so searching speed will increase 14% when data comes to large scale.

4.5 Give the correct ranking

In our system, to get the more accuracy matching result, we have handled different ranking operation for candidate and employer.

- **Candidate**
  For the candidate, they could update and edit their skills to the e-portfolio. The candidate could determine the proficiency ranking 1 to 5 by their self. The number 1 indicate that the user just know this technology, but don't have the complete theory knowledge. The number 2 indicate that the user just have the basic theory knowledge. The number 3 indicate that the user has a relatively complete theoretical knowledge base, but do not much actual operation experience. The number 4 indicate that a user not only has the relatively systematic theoretical knowledge but also has many programming experiences. The number 5 indicate that a user has a deep understanding of this technology and experiences actual project developing. Base on the user's personal condition, they could add or delete any skills at any time.

- **Employer**
  For employers, they could set a specific skill requirement in their recruitment information. It is feasible that set multiple skills in one position. Also, the employer could prioritize demand technology. For example, a company wants to hire a student to be a website engineer with three specific programming skills which are HTML, Java and python. In this situation, this company could set the priority like this: HTML > java > Python. This setting can help employers find candidates who meet their needs more accurately.

- **Method Step**

  - ✓ Recommend candidates to employer user (Figure 3):
    1. Find all candidate users who have all required skills.
    2. For each candidate, sum the level number of all required skills, then use the sum of the level number of all unrequired skills to multiply a low weight, which is 0.01. Sum these two parts.
    3. Use find the candidate who has the highest score.
    4. Recommend the candidate use to the employer user timely.

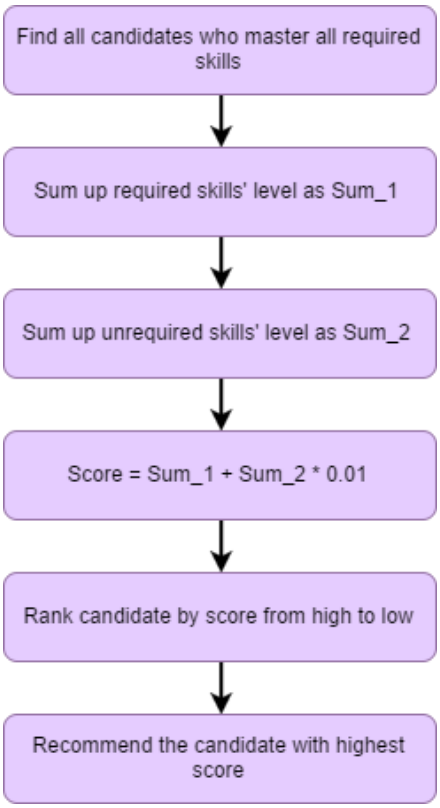

Figure 3 Flow chart of algorithm about recommending candidates to employer users

- ✓ **Recommend positions to candidate user (Figure 4):**
  1. Find all positions which the candidate masters all the position's required skills.
  2. Calculate the distance for all skills of each position, use Manhattan distance.
  3. Find the position with the closest distance.

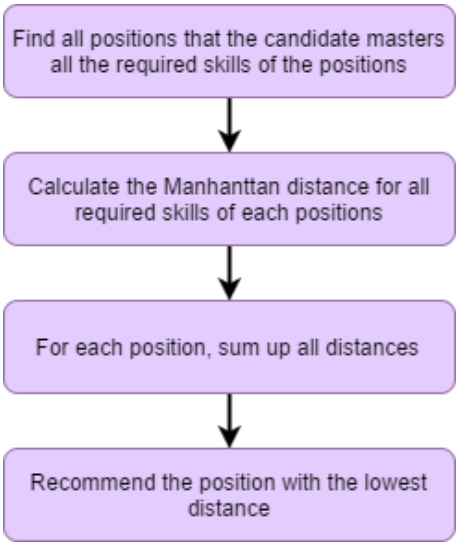

Figure 4 Flow chart of algorithm about recommending jobs to candidates

4.6 Time – performance evaluation

Furthermore, the performance of the server will affect the processing time and variate the user waiting time consequently. We got the Figure 5 by testing massive data. It can be seen that the relationship between search time and data size is roughly proportional. Therefore, with more memory of the server, the speed for receiving and computing is more efficiently. This will bring unpredictable effect to our recommending system.

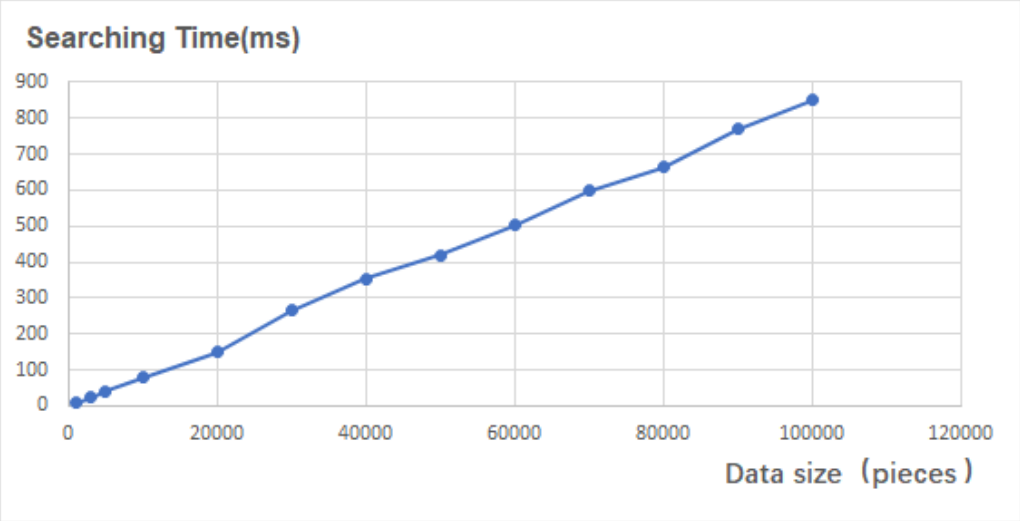

Figure 5 Searching time – Data size

5. User documentation

5.1 Install instruction

- **Please make sure the local machine has the NodeJS, and the Angular CLI install appropriate in the machine.**

➢ **Backend**

1. First clone both frontend and backend files into local machine.
2. Cd into the frontend directory, run "**npm install**", then it will start to install all modules automatically.
3. Run "**node app.js**", then the backend system will start immediately.

- **Frontend**
  1. Cd into the frontend, then run "**npm install**", it will install the required package for support the front end appropriately.
  2. Run "**ng serve**".
  3. Open browser, open a new blank webpage, enter "**localhost: 4200**" in the address bar of the browser.
  4. Then the frontend server will run at the browser.

Reference

1. Granovetter, M. (1995) *Getting a job: A study of contacts and careers*. University of Chicago press, pp.3 - 7
2. Angular.io (2018) *Angular Home page*. Available at: https://angular.io (Accessed: 15th October 15, 2018).
3. Angular.io (2018) *Angular Docs*. Available at: https://angular.io/api/common/http/HttpClientModule (Accessed: 15th October 15, 2018).
4. Node.js (2018) *Node.js About*. Available at: https://nodejs.org/en/about/ (Accessed: 15th October 15, 2018).
5. Node.js Express (2018) *Node.js Express Resources Middleware: body-parser*. Available at: https://expressjs.com/en/resources/middleware/body-parser.html (Accessed: 15th October 15, 2018).
6. Node.js Express (2018) *Node.js Express Resources Middleware: cors*. Available at: https://expressjs.com/en/resources/middleware/cors.html (Accessed: 15th October 15, 2018).
7. MongoDB (2018) *MongoDB: What's MongoDB*. Available at: https://www.mongodb.com/what-is-mongodb (Accessed: 15th October 15, 2018).
8. MongoDB (2018) *MongoDB: Docs*. Available at: https://docs.mongodb.com/?_ga=2.164547835.576043434.1539576330-102675203.1539576330 (Accessed: 15th October 15, 2018).
9. Beautiful Soup 4.2.0 (2018) *Beautiful Soup 4.2.0: Documentation*. Available at: https://www.crummy.com/software/BeautifulSoup/bs4/doc/ (Accessed: 15th October 15, 2018).