



# CV-CLEANER

Technical Report

Enhancing time and cost management with Natural Language Processing and Artificial Intelligence.  
Enhancing the quality of student's CVs with Machine Learning.

**Barry Lougheed, BSc in Computing**  
x14342501

## Contents

Abbreviations .....	4
Executive Summary .....	6
1. Introduction .....	7
1.1. Background .....	7
1.2. Aims .....	9
1.3. Technologies .....	10
1.3.1. Technologies for Statistical Analysis .....	10
1.3.2. Technologies for Application .....	11
1.4. Project Structure .....	12
2. System .....	13
2.1. User Requirements .....	13
2.2. Functional Requirements .....	13
2.3. Use Cases .....	15
2.3.1. Requirement 1 <Allowing the user to paste the content of their CV to the system>....	15
2.3.2. Requirement 2 <Allowing the user to upload their CV to the system>.....	16
2.3.3. Requirement 3 <The system will score the user's CV through a weighted scoring algorithm> .....	17
2.3.4. Requirement 4 <The system will detect what industry the user's CV is most suited to>	18
2.3.5. Requirement 5 <The system will provide analysis of how to enhance the quality of the user's CV> .....	19
2.3.6. Requirement 6 <The system will allow the user to filter CVs by its score> .....	20
2.3.7. Requirement 7 <The system will allow the user to filter CVs by its content> .....	21
2.4. Non Functional Requirements.....	20
3. Architecture and Design.....	24
4. Graphical User Interface.....	40
5. Testing.....	45
5.1. TDD (Test Driven Development) .....	45
5.2. Unit Testing .....	47
5.3. Usability Testing .....	48
5.3.1. Five Second Test .....	48
5.3.2. System Usability Scale.....	49

<b>6. Conclusion</b> .....	51
<b>7. References</b> .....	52
<b>Appendix A - Survey</b> .....	53
<b>Appendix B – CV Cleaner Invention Disclosure Form</b> .....	56
<b>Appendix C - Monthly Journals</b> .....	61
<b>September</b> .....	61
<b>October</b> .....	62
<b>November</b> .....	62
<b>December</b> .....	63
<b>January</b> .....	63
<b>February</b> .....	64
<b>March</b> .....	64

## Declaration Cover Sheet for Project Submission

### SECTION 1 *Student to complete*

<b>Name:</b>  Barry Lougheed
<b>Student ID:</b>  14342501
<b>Supervisor:</b>  Ms. Frances Sheridan

### SECTION 2 Confirmation of Authorship

*The acceptance of your work is subject to your signature on the following declaration:*

I confirm that I have read the College statement on plagiarism (summarised overleaf and printed in full in the Student Handbook) and that the work I have submitted for assessment is entirely my own work.

Signature: Barry Lougheed Date: 13/05/2018

## Abbreviations

<b>Abbreviation</b>	<b>Description</b>
CV	Curriculum Vitae
AI	Artificial Intelligence
NLP	Natural Language Processing
ML	Machine Learning
GUI	Graphical User Interface
API	Application Programming Interface
REST	Representational State Transfer
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
NLTK	Natural Language Toolkit
UI	User Interface
SQL	Search Query Language
XSS	Cross Site Scripting
HTTP	Hypertext Transfer Protocol
CDN	Content Delivery Network
MTFB	Mean Time Between Failure
JSON	JavaScript Object Notation
XML	Extensible Markup Language
PHP	Hypertext Preprocessor
IT	Information Technology

CORBA	Common Object Request Broker Architecture
DRY	Don't Repeat Yourself
TDD	Test Driven Development
SUS	System Usability Scale

## Executive Summary

This purpose of this report is to provide the reader with a comprehensive understanding of the design and development of CV Cleaner. The report contains an in-depth analysis and evaluation of the project. This includes the project concept, aims, and results. CV Cleaner is an innovative concept, with a unique algorithm that is based on a statistical analysis. CV Cleaner is a recruitment processing application used to improve an organization's time and cost management. Furthermore, computer science students can use CV Cleaner's complex algorithm to enhance the quality of their CV (Curriculum Vitae). The web application filters CVs through a weighted scoring algorithm, therefore enhancing an organization's time and cost management. A recruiter can upload multiple CVs to the system, which will grade the CVs for them. This is achieved by multiple techniques, including industry analysis, spelling error detection, and content analysis. The application showcases the project's potential for commercial success, as the project concept targets multiple audiences. This is further supported by the lack of competitive organizations in this market. The application provides the user with a slick front end, and the operations are seamless due to the well-coded API.

The algorithms are based on a complex statistical analysis, carried out on two datasets. The datasets contain CVs from software development students. These CVs are further sub-categorized by CVs that warranted responses from companies, i.e. obtaining an interview, versus CV's that did not warrant a response. With the use of machine learning, there are clear distinctions found between the datasets. This has allowed me to create algorithms that achieve the objectives of this tool.

# 1. Introduction

## 1.1. Background

The idea to create this application came from a combination of a general interest of artificial intelligence, and having friends ask me for assistance in the creation of a CV. A couple of my friends have mentioned to me that they have applied to multiple companies in the past where they did not receive a response. This became frustrating for them. Upon investigation, it was discovered that they had omitted crucial elements from their CV. I searched for applications that could assist them with this issue and were also free to use. However, I could not find an application that could provide this service both on a desktop, or mobile. The idea then sparked for me to create an application that will enhance the quality of CVs. As there was already a service where humans assist people, I thought an artificial intelligent agent would be a better alternative. The AI could assist you at any time you please, compared to waiting for a human to respond. I was already interested in artificial intelligence and had nonchalantly watched youtubers create applications involving AI before. As a result, I decided to apply a general interest to an issue I have personally had in the past.

Another reason for wanting to create this application is the demand for people with knowledge of artificial intelligence. Major companies such as Google, IBM and Facebook have acquired startups who focus on artificial intelligence in the last 2 years such as Dark Blue Labs, EMU and Cognea. Many applications featuring artificial intelligence have become synonymous with mobile devices like Prisma, Siri & Google Now. In the future, many more applications will implement advanced AI systems to their OS. AI will enhance the constructive techniques and capabilities of applications. Artificial Intelligence and cognitive computing is becoming extremely popular in genres of apps such as gaming, streaming and for data analysis. As such, I decided to go even further and apply artificial intelligence to data analysis, as well as a web application.

Before deciding to commit to this project, I had to carry out market research to ensure it was both unique, and potentially commercially successful.

I developed a system for ranking the threat of competitors in order to categorize level of threat:

<b>Level</b>	<b>Type</b>	<b>Monitored</b>
<b>1</b>	No threat, not monitored	No
<b>2</b>	Low threat/Not in market	No
<b>3</b>	Potential threat (Different app in other market) – monitored	Yes
<b>4</b>	Medium threat (Similar app in other market)	Yes

5	High threat (Direct competitor in market)	Yes
---	---	-----

**[Interview.ie:](#)**

This is a web application that allows users to interact with a human consultant that assists them in the creation of their CV. The user has to pay a fee for the consultancy and is at a disadvantage in terms of availability compared to my application as the user will have to wait for the human to respond. As it is available on the same platform, it is within the same market as my application and as such will be a high threat. Level 5.

**VisualCV Resume Builder:** Google Play Store

This is a mobile application that allows users to create CVs from scratch. This is simply a template builder, and as such does not have the same purpose or functionality as my application. While it is a similar app, it is not in the same market as my application. Level 2.

There is no other online competitor for my application. However, there is a real-world service in the college that provides consultancy on CVs. National College of Ireland holds CV clinics for their students. While they do not charge, it is still a competitor, as potential users of my application may opt to attend this clinic instead of using the application.

**CV Cleaner’s Market Potential**

This application addresses multiple target markets, thereby increasing its potential for commercial success. One of these audiences are universities/colleges with work placement programs. Colleges and universities such as National College of Ireland, Dublin City University and University City Dublin all have work placement programs for their students. These programs require students to seek employment in their relative fields. As such, the programs require students to send their CVs to multiple organizations. Workshops to enhance their CVs are set up. This application could benefit these programs greatly as it could speed the process of employment by lightening the workload for the career officers.

Another target audience is any organization that receive CV applications online. The recruitment officers may have over one-hundred CVs to run through before deciding who to contact for interviews. With this application, they can lower this number dramatically, therefore assisting the recruitment process.

## 1.2. Aims

The aims of the project are to develop an application that assists students in enhancing the quality of their CV, and to improve an organization's time and cost management through CV filtration. Another aim is to complete a statistical analysis on relative datasets to support the logic for the algorithms.

1. The analysis and backend will be coded in Python.
2. The project will incorporate NLP (Natural Language Processing), artificial intelligence (AI), and machine learning.
3. The AI algorithm will be developed in the Python language.
4. The program will clean the user's data, so it is readable.
5. The algorithm will be able to detect what industry the CV is suited to.
6. It will also be able to detect spelling errors, key words, and consider the word count.
7. The application will be able to filter CVs by spelling errors, score and content.
8. Overall, the algorithm should be accurate.
9. The system is a web application that will be developed using the Django framework.
10. The GUI (Graphical User Interface) will comply with modern industry standards. It will be both professional and easy to use. The visualization of an application can affect the user's perspective of the system, regardless of the functionality.
11. The API (application programming interface) will be hosted by a REST framework. It will send POST and GET requests to the server to be able to read the user's CV and return the results. The user will also be able to trigger DELETE requests in order to delete their data from the database.
12. The technologies used to complete the project will be free. An agile approach will be applied to the development of this project. Extensive documentation and planning will occur before coding the application. The application is developed with the intention of being commercially successful.

## 1.3. Technologies

Multiple technologies were used to develop this application. These technologies were carefully chosen to not only achieve the aims of the project, but to ensure the project was completed within the deadline. The focus of the project is around Natural Language Processing and artificial intelligence. The technologies mentioned below allowed me to showcase these skills and complete the project on time.

### 1. Research – Google Scholar, Books

Google scholar is a search engine specifically used for scholarly literature.

Artificial Intelligence: A Modern Approach – Peter Norvig and Stuart J. Russel (*Found in references*).

### 2. Documentation and Planning – Microsoft Word, Project, Lucidchart

Microsoft Word allowed me to document the project. Microsoft Project was used to create the project plan. [Lucidchart](#) is a web application that I used to create project diagrams.

### 3. Prototype – HTML, CSS, JavaScript, Bootstrap

The project prototype was created using these technologies.

## 1.3.1. Technologies for Statistical Analysis

The analysis was carried out on two data-sets. These data-sets consisted of CVs from software developers. This was further sub-categorized into CVs that received responses from companies e.g. an interview/phone call, against CVs that did not receive a response.

### 1. Text editor – IDLE

IDLE for Python 3.6 (32 bit) was the editor used to carry out the analysis. It comes pre-installed with the installation of Python.

### 2. Language – Python

Python 3.6 .4 was used to clean the data and develop the analysis

### 3. Libraries – NLTK, Scikit-Learn

[NLTK](#) (Natural Language Toolkit) was used to prepare the data for analysis. This is a library for NLP. Techniques such as tokenization, stopping, and stemming (these techniques will be explained in detail).

[Scikit-Learn](#) was used to carry out the machine learning operations. This includes industry, keyword analysis.

### 4. Data-sets

There were 40 CVs in each data-set. They are categorized as “neg” (negative) and “pos” (positive). “Pos” contains CVs that received responses from organizations and “neg” contains CVs that did not.

The data-sets contain personal information such as phone numbers, names and emails. For data protection compliance, the data will not be disclosed.

## 1.3.2. Technologies for Application

### 1. Text editor – Atom

[Atom](#) was the editor used to create the API and the client framework.

### 2. Language – Python

Python 3.6 .4 was used to create the API and the client framework.

### 3. Framework – Django 1.9

[Django](#) is a RESTful framework that I have used to develop the application. The API and the client are developed on this framework.

### 4. Version Control – GitHub

The application and analysis are stored on [GitHub](#). This cloud platform allowed me to revert to the latest version in the event of losing the project locally.

## 1.4. Project Structure

A high-level overview of the development of the project is explained in this section. The project architecture will be explained in detail later in the report.

### 1. Data Gathering

I gathered two datasets of 40 CVs tailored for the software development industry. One dataset contained CVs that received responses from organizations. The other dataset contained CVs that did not. These will be referred to as positive and negative datasets throughout this report.

### 2. Data Cleaning

Once the dataset was obtained, the data had to be cleaned and formatted. Multiple techniques were used to clean the data such as tokenization, removing stop words and stemming. Stop words are irrelevant words to the analysis such as “the”. Removing them will improve the performance of the analysis. Tokenization is the process of splitting text in documents by words and stemming essentially chops off the ends of words to group multiple derivatives into the one group.

### 3. Statistical Analysis

Upon completion of data cleaning, the statistical analysis was executed. The purpose of the analysis is to set criteria for the algorithm in the API. For example, the analysis found that LinkedIn accounts appeared 34 times in the positive dataset, compared to just 20 in the negative dataset. It can be concluded that it is beneficial to include your LinkedIn account on a CV according to this statistic. Another example is the average word count of the successful dataset was 400 words. From this, the section of the algorithm that scores the user’s CV based on their word count has statistical evidence to support the scoring. A total of 31 cases were found for the criteria of the algorithm.

### 4. Student CV

Students can paste their CV in the UI and run it through the algorithm. The algorithm will analyze the content of their CV and score it from a weighted scoring system. The algorithm can tell the user what industry the CV is best suited to, with the use of machine learning. The algorithm will also detect spelling errors within the user’s CV. The score is then returned to the user, along with automated advice on how to enhance the quality of the CV. For example, if the API did not detect the user mentioning what class of degree they have, the API will advise them to do so.

### 5. Recruitment

Recruiters can upload multiple CVs to the system. The API will score these CVs and store them in the database. The recruiter can then filter the CVs by score, or by searching for a specific value, such as “senior developer”. Every filter query will filter CVs with spelling errors.

## 2. System

### 2.1. User Requirements

A survey was carried out by 17 clients (computer science students) to assist in the design of the web application. The clients answered questions to define the importance of certain features of the application. The full survey can be found in the Appendix A section of the report. The following questions formed the foundation for the core functionality of the application:

1. Have you applied for jobs and not received a response?
2. If available, would you use an application that would predict the probability of an employer contacting you based on your CV?
3. If yes, would you be interested in receiving advice on tailoring your CV to a specific role?

Of the 17 clients, just one voted “No” to question 1. All 17 clients voted “Yes” to every other question above. This confirms there is a market for this application for this feature alone.

### 2.2. Functional Requirements

1. The system will allow the user to paste the content of their CV for analysis, so the content is not stored.
2. The system will allow the user to upload their CV to the system, so the content is stored.
3. The system will score the user’s CV through a weighted scoring algorithm.
4. The system will provide analysis of how to enhance the quality of the user’s CV.
5. The system will allow the user to filter CVs by its score, including spelling errors.
6. The system will allow the user to filter CVs by its content (e.g. “java developer”), including spelling errors.

The context diagram below entails a visual representation of the student’s experience with the application:

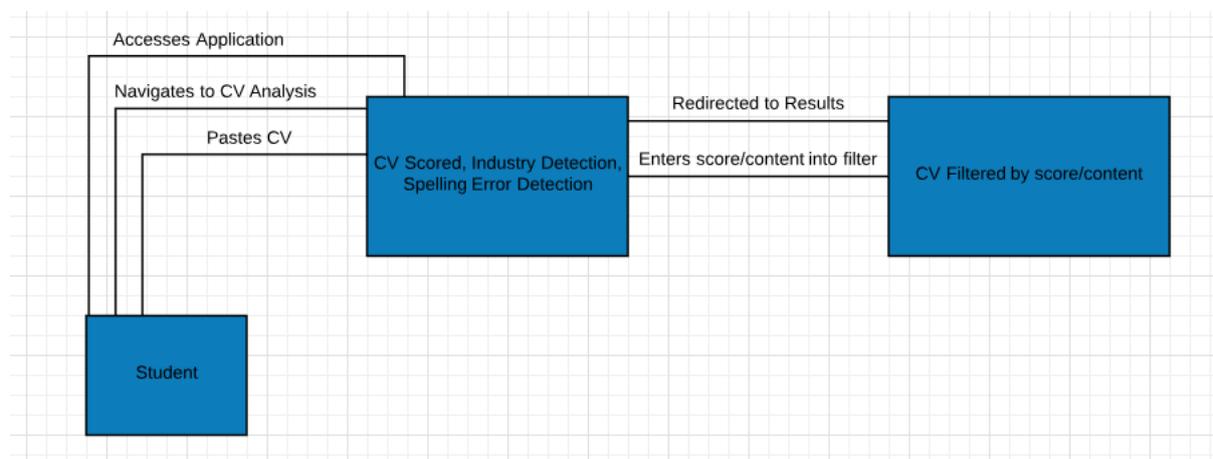


Fig 1: User Context Diagram for Student

The context diagram below entails a visual representation of the employer’s experience with the application:

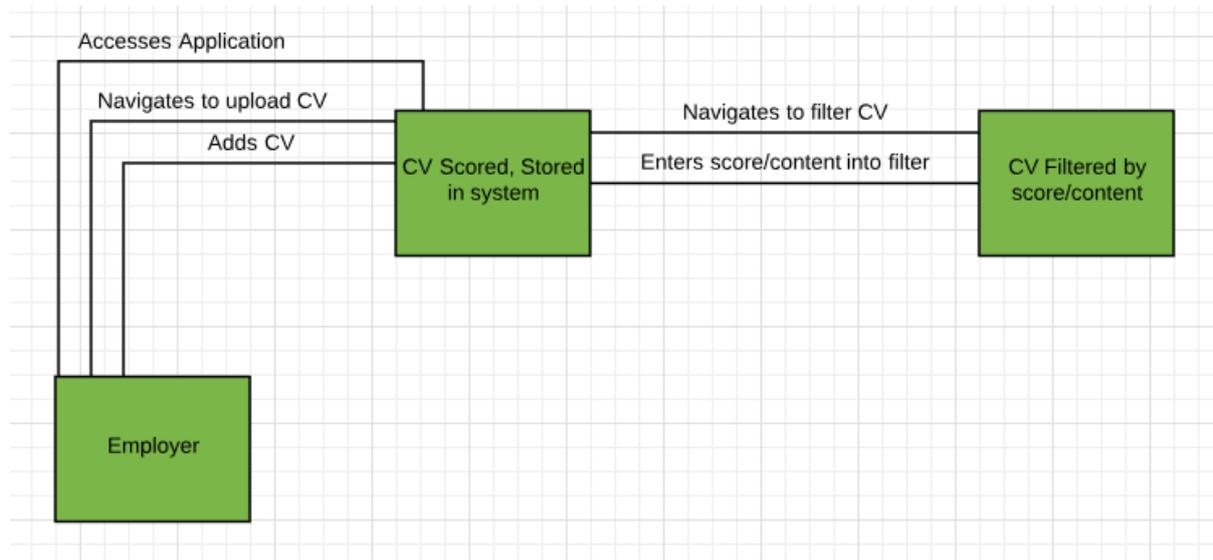


Fig 2: User Context Diagram for Employer

The use case diagram below showcases the steps to execute the core functionality of the application:

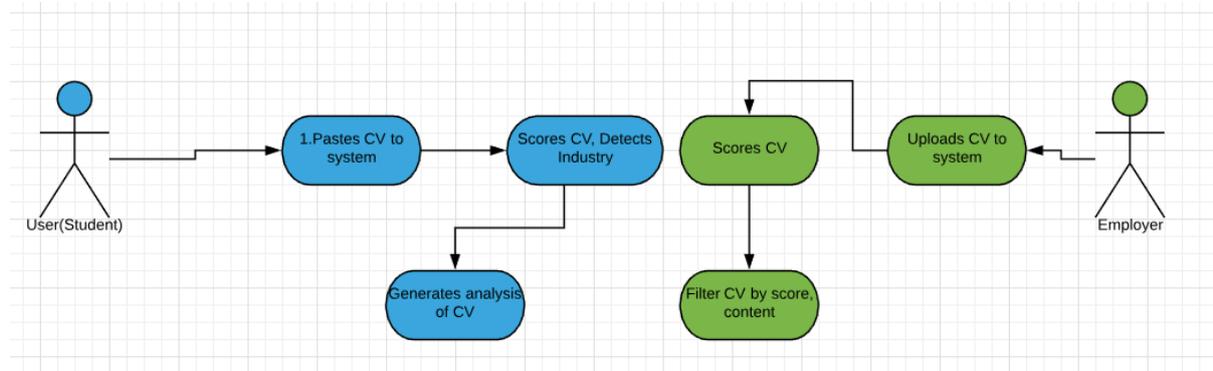


Fig 3: Use Case Diagram for CV Cleaner

## 2.3. Use Cases

### 2.3.1. Requirement 1 <Allowing the user to paste the content of their CV to the system>

#### **Description & Priority**

This requirement will allow the user to paste the content of their CV in a text field, which is retrieved by the API. The content of the text field is not stored as a security and performance measure.

#### **Use Case**

#### **Scope**

The scope of this use case is to allow the user to paste the content of their CV in a text field on the User Interface.

#### **Description**

This use case describes the process of the user pasting the content of their CV to the system.

#### **Flow Description**

#### **Activation**

This use case starts when the user navigates to the CV Analysis activity in the system and submits the text field.

#### **Main flow**

1. The user accesses the application
2. The user navigates to the CV Analysis activity
3. The user pastes the content of their CV to the text field
4. The user clicks the "Analyse" button

#### **Post condition**

The system redirects the user to the Results page and returns the results to the user.

## 2.3.2. Requirement 2 <Allowing the user to upload their CV to the system>

### **Description & Priority**

This requirement will allow the user to upload their CV file to the system.

### **Use Case**

### **Scope**

The scope of this use case is to allow the user to upload their CV to the system.

### **Description**

This use case describes the process of the user uploading their CV to the system.

### **Flow Description**

### **Activation**

This use case starts when the user navigates to the Add CV activity in the system and submits their file.

### **Main flow**

1. The user accesses the application
2. The user navigates to the Add CV activity
3. The user fills in the CV name field
4. The user selects their file to upload
5. The user clicks the "Submit" button

### **Post condition**

The system redirects the user to the Search page where a list of CVs is available.

### 2.3.3. Requirement 3 <The system will score the user's CV through a weighted scoring algorithm>

#### **Description & Priority**

This requirement will allow the system to score the user's CV.

#### **Use Case**

#### **Scope**

The scope of this use case is to allow the system to score the user's CV.

#### **Description**

This use case describes the process of the system scoring the user's CV.

#### **Flow Description**

#### **Activation**

This use case starts when the user uploads or submits their pasted content to the system

#### **Main flow**

1. The user accesses the application
2. The user navigates to the Add CV activity or to the CV Analysis activity
3. The user fills in required fields
4. The user clicks the "Submit" button or the "Analyse" button.
5. The system runs Keyword Analysis and assigns a score
6. The system runs Wordcount Analysis and assigns a score
7. The system calculates a total score
8. The system returns the score to the user

#### **Post condition**

The system redirects the user to the Search page where a list of CVs is available, or the results page.

## 2.3.4. Requirement 4 <The system will detect what industry the user's CV is most suited to>

### **Description & Priority**

This requirement will allow the system to detect what industry the user's CV is most suited to

### **Use Case**

### **Scope**

The scope of this use case is to allow the system to detect what industry the user's CV is most suited to.

### **Description**

This use case describes the process of the system detecting what industry the user's CV is most suited to.

### **Flow Description**

### **Activation**

This use case starts when the user navigates to the CV Analysis activity in the system and submits the text field.

### **Main flow**

1. The user accesses the application
2. The user navigates to the CV Analysis activity
3. The user fills in the required field
4. The user clicks the "Analyse" button.
5. The system runs Industry Analysis algorithm and assigns an industry

### **Post condition**

The system redirects the user to the Results page where the suited industry is returned.

### 2.3.5. Requirement 5 <The system will provide analysis of how to enhance the quality of the user's CV>

#### **Description & Priority**

This requirement will allow the system to provide analysis of how to enhance the quality of the user's CV.

#### **Use Case**

#### **Scope**

The scope of this use case is to allow the system to provide analysis of how to enhance the quality of the user's CV.

#### **Description**

This use case describes the process of the system providing analysis of how to enhance the quality of the user's CV.

#### **Flow Description**

#### **Activation**

This use case starts when the user navigates to the CV Analysis activity in the system and submits the text field.

#### **Main flow**

1. The user accesses the application
2. The user navigates to the CV Analysis activity
3. The user fills in the required field
4. The user clicks the "Analyse" button.
5. The system runs analysis of how to enhance the quality of the user's CV.

#### **Post condition**

The system redirects the user to the Results page where the analysis of how to enhance the quality of the user's CV is returned.

## 2.3.6. Requirement 6 <The system will allow the user to filter CVs by its score>

### **Description & Priority**

This requirement will allow the user to filter CVs in the system by its score.

### **Use Case**

### **Scope**

The scope of this use case is to allow the user to filter CVs in the system by its score.

### **Description**

This use case describes the process of the user filtering CVs in the system by its score.

### **Flow Description**

### **Activation**

This use case starts when the user navigates to the Search CV activity.

### **Main flow**

1. The user accesses the application
2. The user navigates to the Search activity
3. The user inputs an integer value to the search field
4. The user clicks the "Search" button.
5. The system returns CVs that have a score greater than or equal to the input value.

### **Post condition**

The system filters CVs that have a score greater than or equal to the input value.

## 2.3.7. Requirement 7 <The system will allow the user to filter CVs by its content>

### **Description & Priority**

This requirement will allow the user to filter CVs in the system by its content.

### **Use Case**

### **Scope**

The scope of this use case is to allow the user to filter CVs in the system by its content.

### **Description**

This use case describes the process of the user filtering CVs in the system by its content.

### **Flow Description**

### **Activation**

This use case starts when the user navigates to the Search CV activity.

### **Main flow**

1. The user accesses the application
2. The user navigates to the Search activity
3. The user inputs a string value to the search field e.g. ("java developer")
4. The user clicks the "Search" button.
5. The system returns CVs that contain the input value.

### **Post condition**

The system filters CVs that contain the input value.

## 2.4. Non-Functional Requirements

### 2.4.1. Security Requirement

Security is a major issue in today's world of computing. Huge companies like Sony have shown that you can be vulnerable to [cyberattacks](#) and data penetration. There are multiple security measures in place to ensure the application is secure.

#### 1. Student's CVs are not stored in the database

To protect student's personal data, the application is designed in a way that their information is not stored in the system. The student can simply paste the content of their CV into a text field, which is handled as a string. Once the user leaves or refreshes the CV analysis on the result page, the result is removed. This way no data can be exploited through common cyberattacks like SQL injections.

#### 2. File upload types are limited

Employers are required to upload the CVs to the system. The application enhances their time and cost management as they don't have to open the files. To prevent cyberattacks such as XSS (Cross Site Scripting), only .txt files can be uploaded to the system. This prevents malicious users from uploading malicious files to the system.

#### 3. Datasets for analysis separate to application

The analysis is a separate entity to the application. The datasets used for the statistical analysis will not be accessible to users as they contain confidential information. The datasets are separate to the application's files. The user will only be able to see results based of analysis of the datasets, not the physical data.

#### 4. Single file uploads

The application only allows single file uploads, which prevents the user from overloading the system. It also limits the file size to 10,000 characters.

### 2.4.2. Performance/Response Time Requirement

Response time can depend on multiple factors such as WIFI latency, geography and server response time. With this in mind, the industry standard of online applications tends to aim for an average response time of below 1000ms. A study was carried by Nielsen Norman Group found that most users leave pages after 59 seconds. As such, the user must be able to upload their CV and have the algorithm return the results within this time-frame. Aside from the calculation of the algorithm, the application must have a maximum response time between user click and response of 2000ms.

#### 1. REST API

Results are returned to the user of this application over HTTP (Hyper Text Transfer Protocol) and is almost instant. REST APIs have stateless clients, meaning it doesn't need to remember any previous state to work. In other words, cookies and caching is not implemented by default.

## 2. Content Delivery Network for Bootstrap

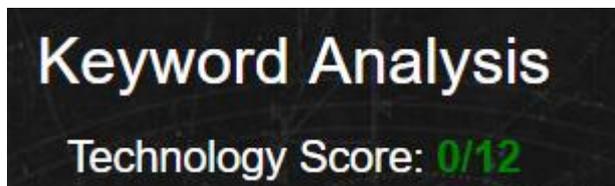
Bootstrap is used for the layout of the application. A CDN (Content Delivery Network) is used to load in the bootstrap properties. This increases the performance of the application. A CDN is a geographically distributed network of servers that connects to the closest server to them. This will increase the application's performance.

### 2.4.3. Scalability Requirement

The addition of features to the application and application growth was acknowledged in the development of the application.

#### 1. Adding to Keyword Analysis

For example, the User Interface states how many technologies are in the Keyword Analysis:



*Fig 4: Length of technology list*

The variable displaying this is not a static integer. It equals the length of the technology list. This means that if more technologies are added to the analysis, it will automatically change.

#### 2. Primary keys in database

The database is based on models. Each time the user uploads a file and title, they are assigned the same primary key. This means every model will be unique.

### 2.4.4. Availability Requirement

The application has been tested and proven it is compatible across all major browsers, such as Chrome, IE11, and Firefox. The application should have a minimum uptime of 99.9%. The CDN used to load bootstrap increases the availability as it is a distributed network. This means that if one server fails, it will be affected as it can connect to another server. The application is also usable from a mobile device. In the below figure, note the navbar change:

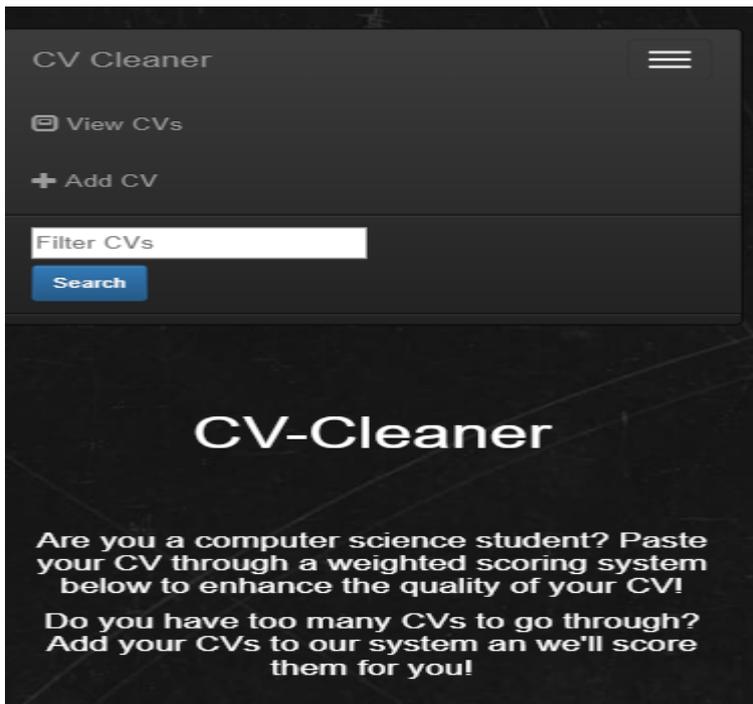


Fig 5: CV Cleaner accessed from mobile

#### 2.4.5. Robustness Requirement

To ensure robustness, programming is implemented to handle unexpected errors or termination. The application provides sufficient error messaging to assist the user to debugging their issue easily. The system is concise and unambiguous to create a stable and smooth environment. One example of this implementation is ensuring the user uploads their file as a text document. See the error handling below when the user does not upload the correct file type:

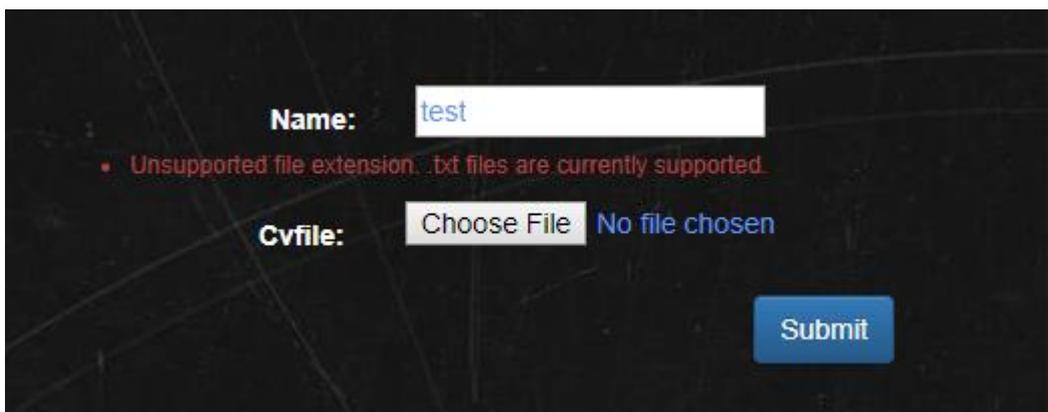


Fig 6: Error handling in CV Cleaner

## 2.4.6. Reliability Requirement

The stability of the system is essential to the success of the project. The application cannot be broken by the user and requests are retrieved within industry standard timing. The MTBF (Mean Time Between Failures) calculation system can be used to measure the reliability of the application. This is measured by the total up time divided by the number of crashes. This computation allows me to understand the stability of the application. After executing this calculation, I can then determine what crashes and bugs require elimination. This requirement ties in closely with the performance. It is imperative that the system runs smoothly to encourage user engagement. As the system has not crashed, the MTBF for CV Cleaner is zero.

## 2.4.7. Maintainability Requirement

Maintenance will be required to control the performance and stability of the system. The application should not be subjected to downtime due to maintenance. This includes updates, and bug fixing. Should maintenance affect the uptime of the system, updates should be carried out during off peak hours on average per user. This is essential to ensure the uptime target stays at 99.9%. Maintenance will be completed in test environments before it is pushed live to minimize downtime.

## 2.5. Data Requirements

Four datasets have been used to complete this project. Two datasets were used for the statistical analysis, and two datasets were used for the industry analysis. User data is also required to execute the core functionalities of the application.

### 1. Datasets for Statistical Analysis

Each dataset contained 40 CVs from software developers. They are categorized as “neg” (negative) and “pos” (positive). “Pos” contains CVs that received responses from organizations and “neg” contains CVs that did not. The data-sets contain personal information such as phone numbers, names and emails. For data protection compliance, the data will not be disclosed.

### 2. Datasets for Industry Analysis

The remaining two datasets were used to detect what industry the user’s CV will be most suited to. This is a machine learning algorithm and requires data to learn from. To showcase this, industries have been split by “Dev” (Software Development), and “Network” (IT Networking).

The Dev dataset contains over 850 software development terms, while the Network dataset contains over 850 networking terms. These datasets can be accessed with the file upload.

### 3. User Data

The system requires input from the user to execute the core functionalities. Student’s must paste the contents of their CV into a text field to execute the analysis. Employers must upload CVs to the system to execute filtering functionality.

### 3. Architecture and Design

This section will describe the architecture of the statistical analysis and the application in detail.

#### 3.1. Architecture and Design of Statistical Analysis

This section describes how the datasets were cleaned for analysis, the execution of the analysis, and the analysis results. The purpose of the statistical analysis was to set criteria and rules for the algorithm. Without the analysis, there is nothing to base the algorithm on. Another reason was to have statistical evidence to prove the algorithm in the application is accurate. For example, I must provide a reason as to why a user's CV gets more points the closer their word count is to 400.

The analysis was executed using IDLE and the results were printed in the terminal.

##### 3.1.1. Cleaning Data

To execute this analysis accurately, the data had to be cleaned for the following reasons:

###### 1. The CV had to be split into words

Natural Language Processing is a branch of artificial intelligence that allows machines to understand human language. To analyse the content of the CV, the CV had to be split by words.

###### 2. Identical words must be removed

Identical words must be removed from the data as it would misrepresent the number of times a specific word appeared in a CV. The purpose of the Keyword analysis is to understand what words appear in the positive dataset, compared to the negative. For example, the word "java" could appear in each of the 40 successful CVs once, meaning it appeared 40 times. It could also appear 40 times in the negative dataset across 20 CVs. Furthermore, the Wordcount Analysis represents how many different words are used in the CVs. These features would be misrepresented if identical words were not removed.

###### 3. Similar words had to be grouped together

The analysis had to account for points being articulated in different ways. One CV could say they are a "driver", while another could say they have a "driver license". These points are identical however they are said in a different way. Words can also be used in different tenses. The machine needs to understand this for the analysis to be accurate.

###### 4. Irrelevant words had to be removed

After the Wordcount Analysis, irrelevant words had to be removed from the data. If they weren't, words such as "the" or "and" would appear in the results. Another reason to remove irrelevant words was to increase the performance of the analysis.

Multiple techniques were used to clean the data:

###### 1. Tokenization

Tokenization is the process of demarcation of text by words. Words are split by white space, full stops, question marks etc. Each word is then stored in an array, rather than the complete string.

## 2. Stopping

Stopping is the process of removing words that are irrelevant to the analysis such as “the”. To remove stop words, the text is tokenized. The words are then compared against a stop word list, using a sequential search technique. If a word matches, the word is removed from the text. This process continues until it has reached the end of the stop word list.

## 3. Lemmatization

Lemmatization is the process of grouping similar words into the one group e.g. “working”, and “worked” can be lemmatized into the word “work”. This is done by analyzing the presence of consonants and vowels at a very low level. It has a list of ending sequences, such as “ing” or “ed”. It tokenizes the text and sees if each word contains these at the end. If it does, it can be removed.

As the data cannot be disclosed. I have provided an example below on my CV to showcase what the cleaned data looks like. This can be replicated by running “dataProcessing.py” in the analysis folder.

```
Linkedin
:
http
:
//www.linkedin.com/in/barry-lougheed/
Barry
Lougheed
Profile
:
Final
year
BSc
Computing
student
specialising
software
development
-
current
grade
:
1:1
7
Months
experience
Software
Analyst
Fenergo
.
```

*Fig 7: Representing cleaned data*

### 3.1.2. Analysis Design

The Wordcount Analysis was carried out to find how many unique words were used in the positive dataset.

The Keyword Analysis was carried out to discover what key words appeared in the positive dataset compared to the negative dataset.

#### 1. Wordcount

This analysis was executed by the WordCount.py file, located in the analysis folder.

This was a simple process that involved binding the CVs into one file, tokenizing, getting the word count and then dividing by the number of CVs in the dataset.

The CVs were bound to one file by navigating to the directory on the command line and running the following command:

```
copy /positiveCVs *.txt > positivebinded.txt
```

```
copy /negativeCVs *.txt > negativebinded.txt
```

Once the CVs were bound, they could be tokenized and run a word count by the command:

```
Print(len(tokenizedlist))
```

The analysis found that the average word count of the positive CVs were 406 words. From this, rules can be set in the algorithm to score CVs relative to these findings.

#### 2. Keywords

This analysis is executed by the KeyWord.py file, located in the analysis folder. This is done by establishing how many times a specific word appears in the positive dataset against the negative dataset. A Naïve Bayes Classifier algorithm was used to determine this. This will be explained in detail in the application architecture and design. Some of the most informative features (words) in the dataset were as follows:

<b>Informative Feature</b>	<b>Appearances in Positive Dataset</b>	<b>Appearances in Negative Dataset</b>
Email	40	40
References	40	32
Profile	40	34
Interests/achievements	40	24
Experience	36	20

Java	32	20
SQL	28	18
Github	26	6
JSON	26	10
XML	26	8
PHP	24	16
Excel	20	10
python	16	4
Ruby	15	4
Drive	10	1

Find below of how the program outputs this to the terminal:

```

ruby = True           pos : neg   =    3.4 : 1.0
python = True        pos : neg   =    3.4 : 1.0
xml = True           pos : neg   =    3.2 : 1.0

```

*Fig 8: output of keyword analysis*

This output means the word “ruby” appeared in the positive dataset 3.4 times more than the negative one. Rules can be set in the algorithm based on these results e.g. the CV receives 1 point for every informative feature mentioned.

Find an architecture diagram below for the analysis:

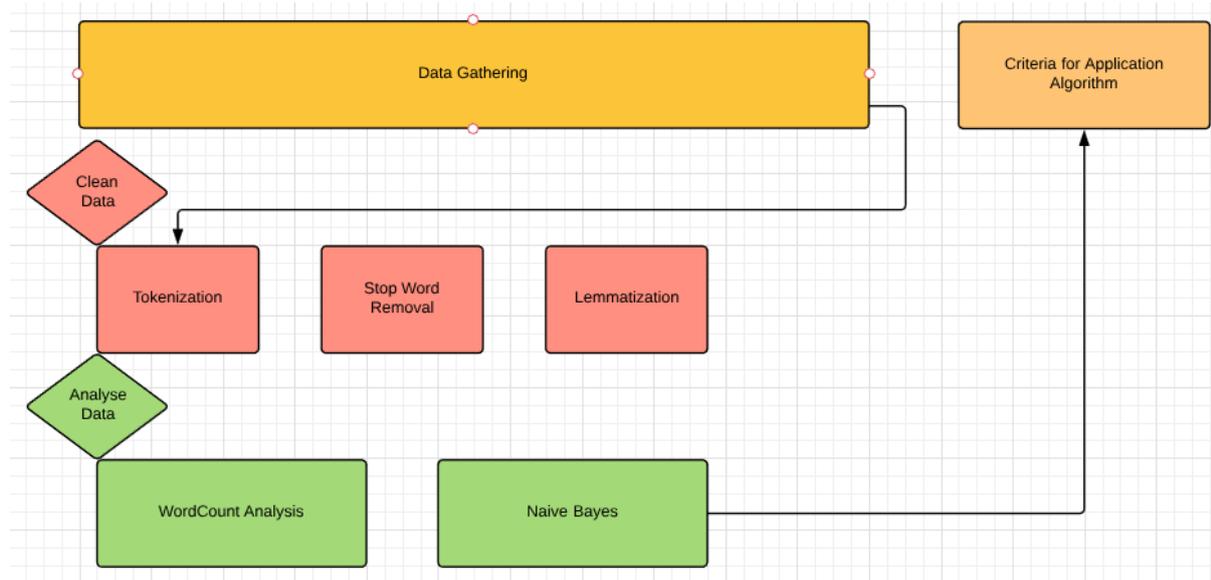


Fig 9: Analysis Architecture Diagram

## 3.2. Architecture and Design of Application

This section describes the architecture and design of the application. As mentioned above, the statistical analysis is the logic behind the weighted scoring system. This section entails how the weighted scoring system is incorporated into the application.

### 3.2.1. Uploading CVs

CVs are uploaded to the database via models. The model consists of four fields: "name", "cv", "cvfile", spellerrors, and "cvscore". This is found in the models.py class. See the model below:

```

class Clean(models.Model):
    name = models.CharField(max_length=100, blank = False)
    #User doesn't see when creating model
    cv = models.TextField(max_length=10000, blank = True, null = True)
    cvfile = models.FileField(validators=[validate_file_extension])
    #User doesn't see when creating model
    cvscore = models.IntegerField()
    spellerrors = models.CharField(max_length=100, blank = False)
  
```

Fig 10: Model for uploading CVs

The "name" is a charField(short string) that can only be 100 characters long. It cannot be blank.

“cv” is a text field that will contain the content of the uploaded CV. It can only be 10,000 characters long and is blank by default. It is blank by default as the cvfile will overwrite it. This field is not available to the user when uploading their model.

“cvfile” is a file field which stores the CV file in the system. It calls a validator function from validators.py that limits what file type can be uploaded to the system. It only allows .txt files as it prevents malicious files from being uploaded to the system.

“cvscore” is an integer field that stores the score of the CV. This field is not available to the user when uploading their model as the algorithm will write the score to this field.

“spellerrors” is a charField that stores whether the CV contains spelling errors or not. This field is not available to the user and the content of this field will either be “yes” or “no”.

The save method is overridden to upload this model as it is not a simple field validation form.

The text from the file is be parsed to the “cv” field, which allows the user to view the CV from the application. The text from the file is firstly cleaned by converting to lower case and ignoring utf-8 characters. This process can be seen below:

```
#Decoding unicode and converting to lower case|
get_text = self.cvfile.read().lower().decode("utf-8", 'ignore')
#Parsing the file's content to charfield cv
self.cv = get_text
```

*Fig 11: Parsing file content to cv field*

Name, cv, and cvfile is now ready to be saved to the database. “cvscore” is now the only field that needs to be completed. Once the CV is scored, the integer is saved to “cvscore” as a percentage. The percentage is calculated by dividing the score of the CV by the total score a CV can get, then multiplying by 100. The scoring system will be explained in the next section. See below:

```
cv_percentage = score / potential_score * 100
self.cvscore = cv_percentage
super(Clean, self).save(*args, **kwargs)
```

*Fig 12: How cvscore is saved*

### 3.2.2 Analysing CVs

The CVs are analysed in the following areas: Industry Analysis, Spelling Error Detection, Wordcount Analysis, and Keyword Analysis.

#### Industry Analysis

This operation identifies what industry the CV is suited to. As the analysis is related to computer science students, the available industry classes are software development and IT/networking.

The industry is detected by machine learning. The code for this can be found in `industry_analysis1.py` and `industry_analysis2.py`, where a Naïve Bayes algorithm is implemented.

#### Naïve Bayes Algorithm

Naïve Bayes is a machine learning algorithm used for text classification. It requires high dimensional data training sets. The algorithm calculates the probability of an object with certain features belonging to a class. In this case, the algorithm is used to calculate the probability of the user's CV being suited to an industry (software development or Networking). It follows Baye's Theorem, which is:

**$P(A/B) = P(B|A)P(A)/P(B)$** . This is read as: the probability of event B given A is equal to the probability of the event A given B, multiplied by the probability of A upon probability of B.

This means the posterior = likelihood x by the prior probability / the evidence of the prior probability. Naive Bayes requires trained datasets in which it can base the probability from.

The two datasets trained for this algorithm were a dataset consisting of 1200 software development terms and a dataset consisting of 1200 networking terms.

Half of each dataset is defined so the machine can base probability from prior evidence.

The remaining data is then shuffled to mitigate statistical bias and fed to the algorithm, which predicts what category the words belong to. As mentioned above, it calculates this from having prior evidence which is the trained dataset.

If the word "java" appeared significantly more in the software development dataset, it can will be classified as such. The algorithm then looks through each word in the CV (as it is tokenized) and computes the probability this way.

#### Enhancing the performance of the algorithm

The algorithm would be significantly slower if the data had to be retrained every time the algorithm was ran. To solve this, the trained algorithm can be saved and then called each time it is ran. This is done using a python library called pickle. "Pickling" is the process where a python object hierarchy is converted into a byte stream. "Unpickling" is the reverse operation. This is essentially what is done in the CORBA service that was covered in the Distributed Systems module (serialization and marshalling). Once the algorithm is trained, it can be "pickled" so it can be later "unpickled" which is a faster process than re training:

```

classifier = nltk.NaiveBayesClassifier.train(training_set)
print("Original Naive Bayes Algo accuracy percent:", (nltk.classify.accuracy(classifier, testing_set))*100)
classifier.show_most_informative_features(25)

#####
save_classifier = open("pickled_algos/originalnaivebayes.pickle", "wb")
pickle.dump(classifier, save_classifier)
save_classifier.close()

```

Fig 13: Naïve Bayes Algorithm being Pickled

```

#First 850 words are used to train against, remaining used for testing
#When training, you tell the algorithm what words appear in the Dev vs Networking
training_set = featuresets[:850]
#Testing set means the machine is not told what category the words are,
#and the machine compares it to the known category
testing_set = featuresets[850:]

open_file = open("pickled_algos/originalnaivebayes.pickle", "rb")
classifier = pickle.load(open_file)
open_file.close()

```

Fig 14: Naïve Bayes Algorithm being Unpickled

The accuracy of the probability can also be presented with a vote classifier:

```
class VoteClassifier(ClassifierI):
    def __init__(self, *classifiers):
        self._classifiers = classifiers

    def classify(self, features):
        votes = []
        for c in self._classifiers:
            v = c.classify(features)
            votes.append(v)
        return mode(votes)

    def confidence(self, features):
        votes = []
        for c in self._classifiers:
            v = c.classify(features)
            votes.append(v)

        choice_votes = votes.count(mode(votes))
        conf = choice_votes / len(votes)
        return conf
```

Fig 15: Returning accuracy of algorithms

Here is an example of my CV being fed to the algorithm:



Fig 16: Result of Naive Bayes

### 3.2.3. Spelling Error Detection

An available API or library could not be used to detect spelling errors in this application as they would flag technologies such as “xml”, or “json” as errors. It would also flag some names as errors. As such I have built a custom spelling error detection API. To do this, I designed and executed the following in spellErrorPrep.py:

1. I downloaded a file containing ten thousand “English” words. This included names and common technologies such as “xml” (10000-english.txt).
2. I downloaded a file containing the eight thousand most common misspellings(originalErrors.txt).
3. The originalErrors.txt file was in the format of “correct spelling: misspelling”. As such, I removed any word that had “:” attached to it.
4. The results were then written to “errors2.txt”.
5. To ensure accuracy, I compared the misspelling file with the correct spelling file. I found that there were some correct spellings in the misspelling file. These had to be removed.
6. The results were written to “misspellingDictionary.txt” to complete the misspelling dictionary.

```
each_line = ""

def removeStr(val):
    if ":" not in val:
        return val

file1 = open("errors2.txt", "w")

#File containing original spelling errors
with open("originalErrors.txt") as the_file:
    for each_line in the_file:
        #Removes correct spellings and splits by word
        each_line = "".join(filter(removeStr, each_line.split()))
        each_line = each_line.replace(","," ")
        #print(each_line)

        #Writing result to errors2.txt
        file1.write(each_line)

file1.close()

file = open ("misspellingDictionary.txt", "w")
#Analysing misspellings
with open('errors2.txt') as a, open('10000-english.txt') as b:
    b=b.read().split()
    #Removing correct spellings
    r='\n'.join([e for e in a.read().split() if e not in b])
    print(r)
    file.write(r)

file.close()
```

Fig 17: Creating misspellings dictionary

The user's CV is then compared against this file. If a word from the user's CV is also present in the misspelling dictionary, then there is a spelling error.

```
#####  
#                               Spell Check  
  
file_ = open(os.path.join(settings.BASE_DIR, 'misspellingDictionary.txt'))  
diction = file_.read().split()  
spell_error_li = '\n'.join([e for e in cv if e in diction])  
number_spell_errors = len(spell_error_li)  
file_.close()  
  
if number_spell_errors > 0:  
    spell_error = True  
elif number_spell_errors == 0:  
    spell_error = False  
  
context_dict['spell_error_li'] = spell_error_li  
context_dict['spell_error'] = spell_error  
context_dict['number_spell_errors'] = number_spell_errors
```

*Fig 18: Spelling error detection*

### 3.2.4. Scoring CVs – Wordcount Analysis

The CVs are scored through a weighted scoring algorithm. The criteria for the scoring system which is set by the statistical analysis is the wordcount and keywords. Wordcount is defined as how many unique words are used.

The analysis found that the average word count for the positive data was 406 words. I created a function to score the user's CV based on how close the word count was to 400 words. See below:

```
#Tokenizing to analyse and score cv
tokened = nltk.word_tokenize(get_text)

cv_words = set(word for word in tokened)
cvCount = len(cv_words)

*****
#
#           Wordcount score
#Between 375 and 425 words (within 25 words of 400)
if cvCount >= 375 and cvCount <= 424:
    wordcount_score = 5
#Between 350 - 374 words or 426-450 words (within 50 words of 400)
elif cvCount >= 350 and cvCount <= 374 or cvCount <= 450 and cvCount >= 426:
    wordcount_score = 4
#Between 325-349 words or 451-475 words (within 75 words of 400)
elif cvCount >= 325 and cvCount <= 349 or cvCount <= 475 and cvCount >= 451:
    wordcount_score = 3
#Between 300-324 words or 476-500 words (within 100 words of 400)
elif cvCount >= 300 and cvCount <= 324 or cvCount <= 500 and cvCount >= 476:
    wordcount_score = 2
#Between 200 - 323 words or 501-600 words (within 200 words of 400)
elif cvCount >= 200 and cvCount <= 323 or cvCount <= 600 and cvCount >= 501:
    wordcount_score = 1
elif cvCount > 199 or cvCount > 601:
    #If its not within 200 words of 400
    wordcount_score = 0
total_wordcount_score = 5
```

Fig 19: Wordcount function

Identical words are removed in the set cv\_words. The word count is then obtained by getting the length of cv\_words.

If the word count is within 25 words of 400, they CV will receive the maximum score, which is 5.

If the word count is within 50 words of 400, the CV will receive 4 points.

If the word count is within 75 words of 400, the CV will receive 3 points.

If the word count is within 100 words of 400, the CV will receive 2 points.

If the word count is within 200 words of 400, the CV will receive 1 point.

If the word count is under 200 words or over 600 words, the CV will not receive any points.

### 3.2.5. Scoring CVs – Keyword Analysis

Keywords were words that consistently appeared across the successful dataset. These were categorized as follows:

1. Technologies – consists of technologies such as (but not restricted to) java, python, ruby.
2. Frameworks – consists of frameworks such as Django, rails, asp.net.
3. Key Components – consists of key elements such as email, references, LinkedIn.
4. Level of Degree – sub categorized into 3 levels. The top level consists of first class degrees, the second level consists of second class honour degrees, and the third consists of pass degrees.
5. Colleges – sub categorized into 2 levels. The top level is universities such as UCD, and Trinity College Dublin, and the other level consists of colleges such as NCI or DIT.
6. Bonus – Consists of bonus keywords found such as being a driver or having personal projects.

These words were added to array lists. The content of the user's CV is then compared against them. The CV receives a point for every word that exists in these lists. However, degrees, universities, and bonus words will get more point. See an example below:

```
#                               KeyComponent score
keycomponent_words = set(word for word in cv if word in keyComponent_list)
missing_key_words = set(word for word in keyComponent_list if word not in cv)
keycomponent_score = len(keycomponent_words)
keycomponentlist = len(keyComponent_list)

context_dict['missing_key_words'] = missing_key_words
context_dict['keycomponent_score'] = keycomponent_score
context_dict['keycomponentlist'] = keycomponentlist
#*****

#                               Degree score
for i in degree_level_1:
    if i in cv:
        degree_score = degree_score + 10

for i in degree_level_2:
    if i in cv:
        degree_score = degree_score + 5

for i in degree_level_3:
    if i in cv:
        degree_score = degree_score + 3
#Stops score being given multiple times
if degree_score == 15 or degree_score == 18:
    degree_score = 10
```

Fig 20: Scoring the CV by keywords

Everything that is returned to the user is added to a dictionary, titled "context\_dict". This is then passed as a return parameter. The score is calculated and returned to the user along with the percentage. The words that the user is missing will also be returned, suggesting the user to add them to their CV. The code for this can be seen below:

```
# Total score
score = wordcount_score + degree_score + keycomponent_score + tech_score + framework_score + driver_score + personal_score + university_score
context_dict['score'] = score

potential_score = universityli + degreeli + driverli + personalli + wordcountli + techlist + frameworklist + keycomponentlist
context_dict['potential_score'] = potential_score

cv_percentage = score / potential_score * 100
context_dict['cv_percentage'] = cv_percentage
#*****
#ensure the CV is not blank
if details != '':
    return render(request, self.template_name, context_dict)
```

Fig 21: Calculating score, returning score and missing words to user

The architecture diagram below entails a visual representation of the process:

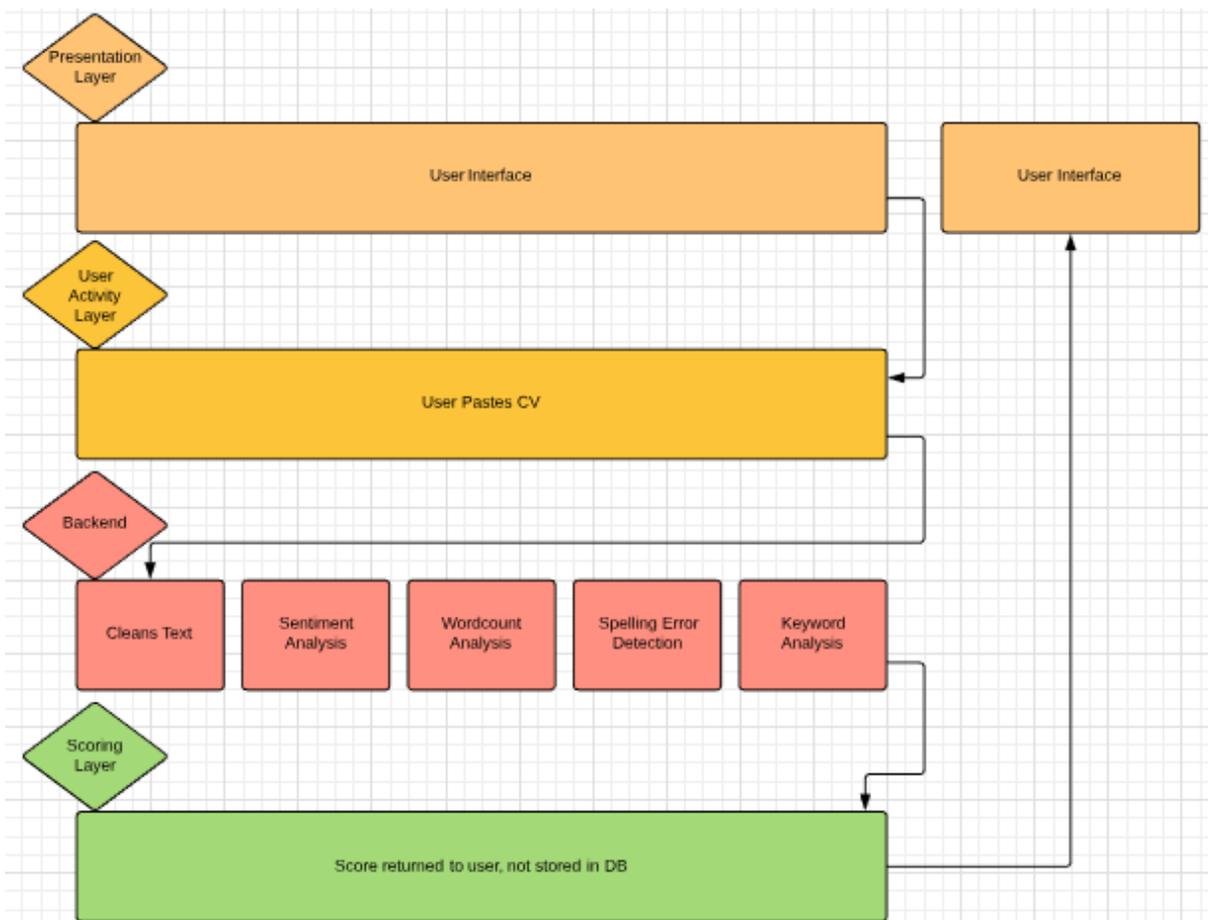


Fig 22: Architecture Diagram for Student CV Analysis

### 3.2.6. Scoring CVs – Wordcount Analysis

Users have the ability to filter uploaded CVs by its score, or by specific content in the CV. The purpose of employers uploading CVs to the system is to reduce their time spent evaluating CVs. This is achieved by uploading CVs to the system, filtering them by score. This can eliminate the poor scoring CVs for the user. The query will also eliminate CVs with spelling errors. See the flow diagram below:

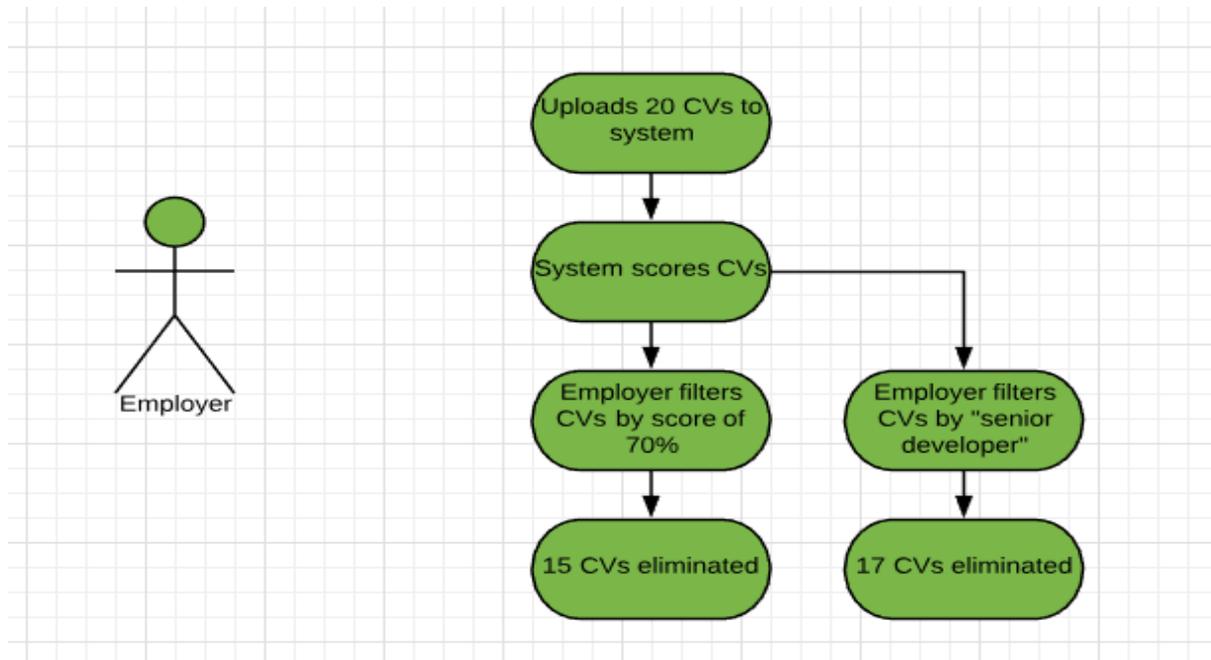


Fig 23: Flow diagram for CV Filtration

The user can filter CVs by score or by content, depending on their query. If the user enters an integer value to the form, a GET request will retrieve and filter CVs by score. The CVs will be sorted in ascending order, starting with the closest score to the query. If the user enters a string value e.g. “National College of Ireland” to the form, a GET request will retrieve and filter CVs by that query. The user can then open these CVs and read them. The function can be seen below:

```
class SearchView(LoginRequiredMixin, generic.ListView):
    login_url = '/accounts/login/'
    redirect_field_name = 'redirect_to'
    model = Clean
    template_name = 'cleaner/search.html'

    #Get request for filtering
    def get_queryset(self):
        cvs = Clean.objects.all()
        query = self.request.GET.get('q','')

        if query.isdigit():
            cvs = cvs.filter(Q(cvscore__gte=int(query)) & Q(spellerrors__icontains="no")).distinct()
            cvs = cvs.order_by('cvscore')
        else:
            cvs = cvs.filter(Q(spellerrors__icontains="no") & Q(cv__icontains=query) | Q(name__icontains=query)).distinct()
        return cvs
```

Fig 24: Filtering CV

## 4. User Interface

The application is spread across five pages. The front end was developed in the Django framework (version 1.9). The languages used to style and create the user interface were HTML, CSS, Bootstrap, and Python. DRY principles are used here as each .html class extends the “Base.html” class. This class contains the nav bar for the application.

### 4.1. Styling

The styling is consistent throughout the application. A dark background image is complimented with a contrasting colour scheme. The text colour is primarily white, which stands out against a dark background. All buttons are blue, and anything that is clickable changes colour when the user hovers over it.

### 4.2. Layout

The application uses a bootstrap layout which creates a responsive design. The layout also responds to mobile use. Each section is easily navigable from every page of the application. A navigation bar at the top of the page allows the user to navigate to the home page, add a CV, or view all CVs from each page. The navbar highlights what page the user is on. The language used is simple, allowing the user to understand. The language is also simple, which negates any wasted space.

The URLs are human-orientated which makes navigating by domain name easy.

The URLs and pages are as follows:

#### 1. Home page - /cleaner/

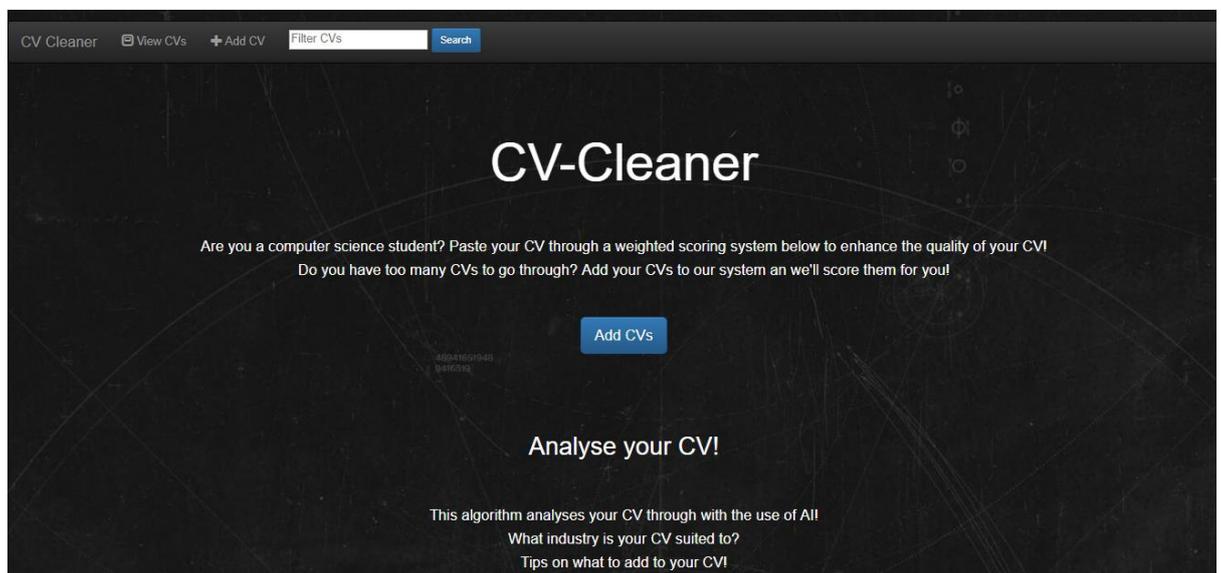


Fig 26: Home page part 1

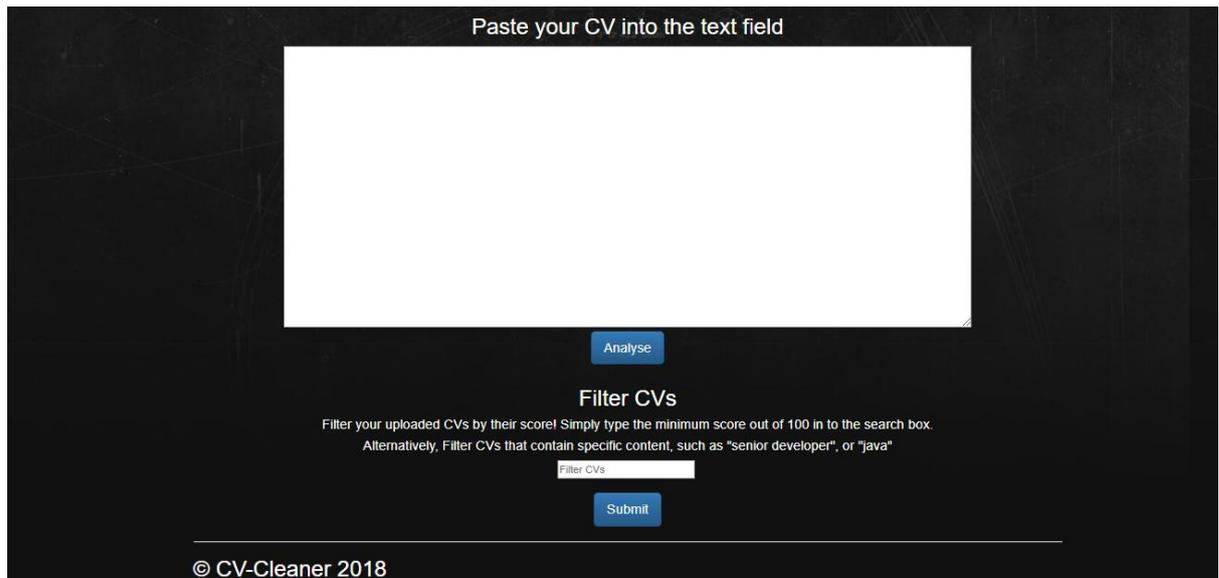


Fig 27: Home page part 2

The home page simultaneously achieves the user needs and the business goals. The user immediately knows the name of the application and understands what purpose the application serves. This is done by having large text of the application name along with “Analyse your CV”. There is a button in the jumbotron to direct the employer to add CVs, and there is also large text “Paste your CV into the text field” under the “Analyse your CV” heading.

## 2. Add CV page - /cleaner/clean/add/

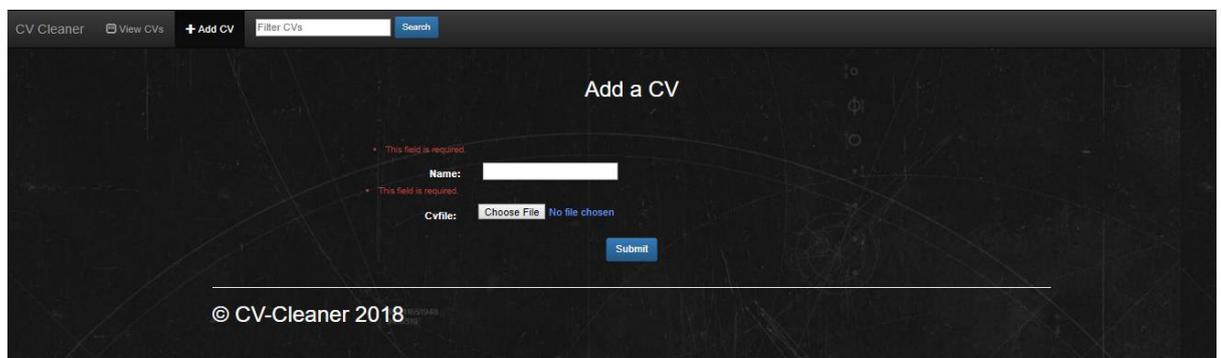


Fig 28: Add CV page

The Add CV page is simple and concise. It allows the user to upload a name and file to the system. The “Submit” button sends a POST request to the server. The nav bar highlights that the user is on this page. Note the error handling. The colour is red to show the user that it is not primary text. The score field is not visible to the user, so they cannot manipulate the score.

### 3. Filter CV page - /cleaner/clean/search/

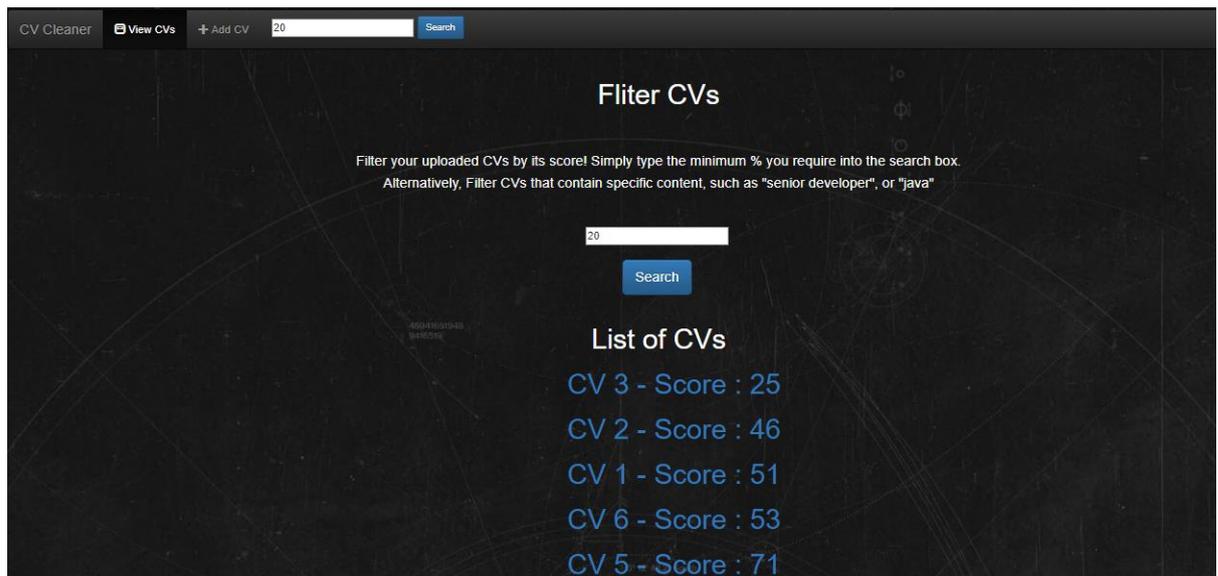


Fig 29: Filter CV page

The Filter CV page allows the user to view all the CVs in the system. Before search/filter, the list of CVs is sorted by ID. This means the CVs are in order by upload time. However, when the user filters by score, they become filtered by the closest score to the user input value. The "Search" button sends a GET request to retrieve the user's query. The nav bar highlights that the user is on this page. Note that the value the user entered has stayed after the CVs were filtered and it is present in the nav bar.

### 4. View CV page - /cleaner/ID/



Fig 30: View CV page

The View CV page allows the user to view a specific CV in the system. The CV name is followed by the content of the CV. Rather than have the user download the content of the CV, the CV is viewable from the application. This is an incentive to keep the user on the application. The score is in a strong green colour, so it stands out on the page.

## 5. Analysis/Result page - /cleaner/clean/result/

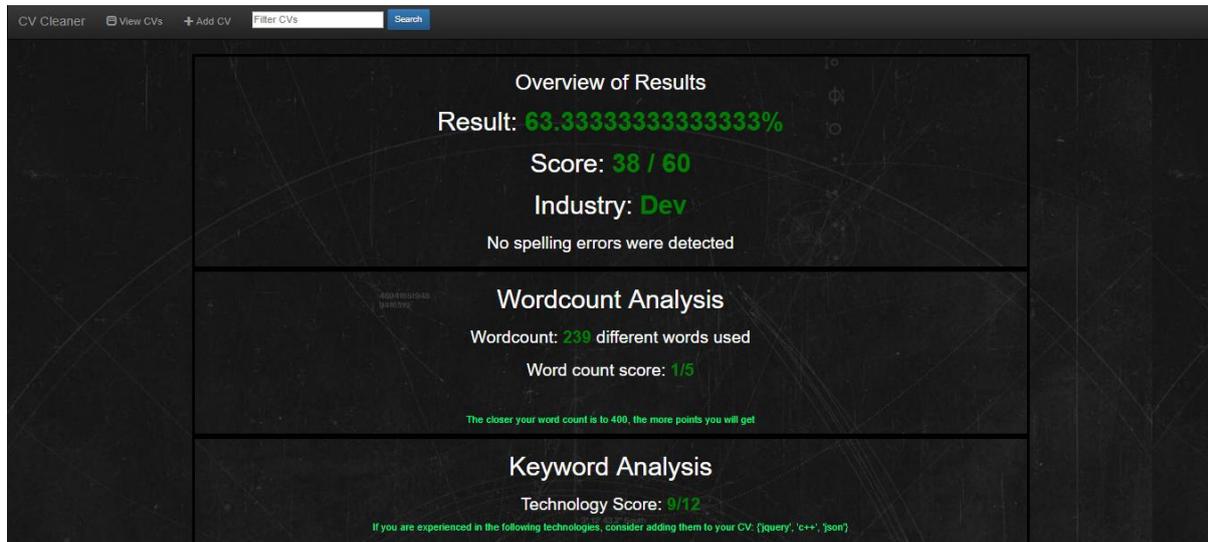


Fig 31: Analysis/Result page part 1

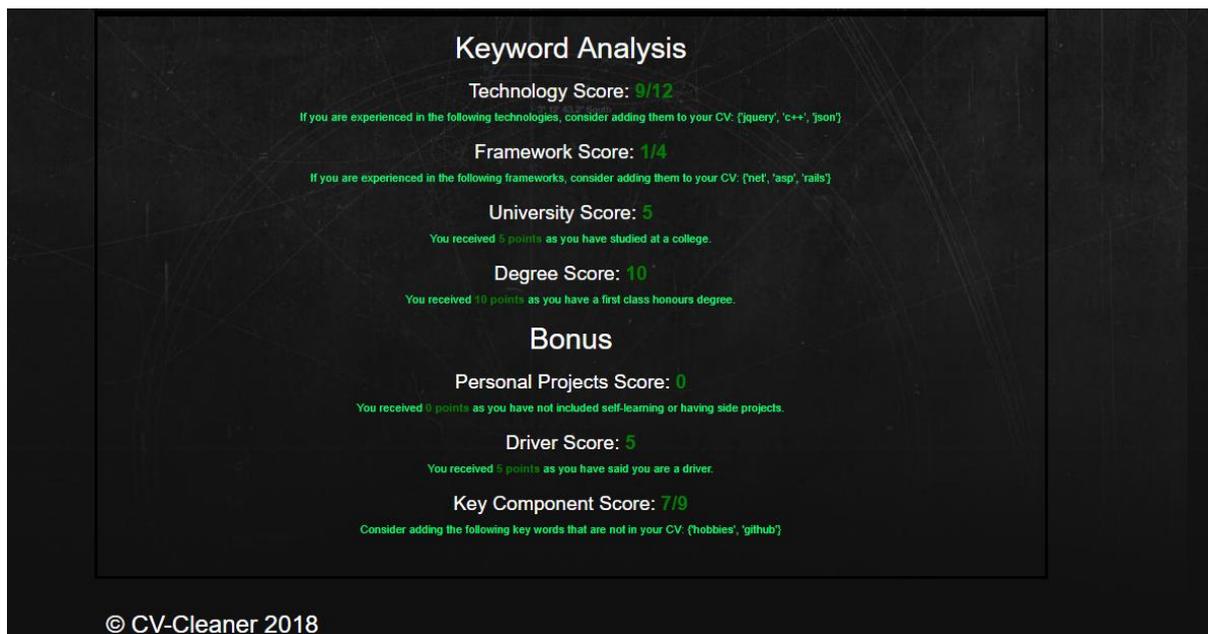


Fig 32: Analysis/Result page part 2

The user is redirected to this page after they click the “Analyse” button on the home page. This sends a POST request which activates the algorithm to score the student’s CV. The score, suited industry and spelling error detection are all presented to the user at the top of the page. The colour is notably different on this page. If the user refreshes this page, their information will be removed.

## 5. Testing

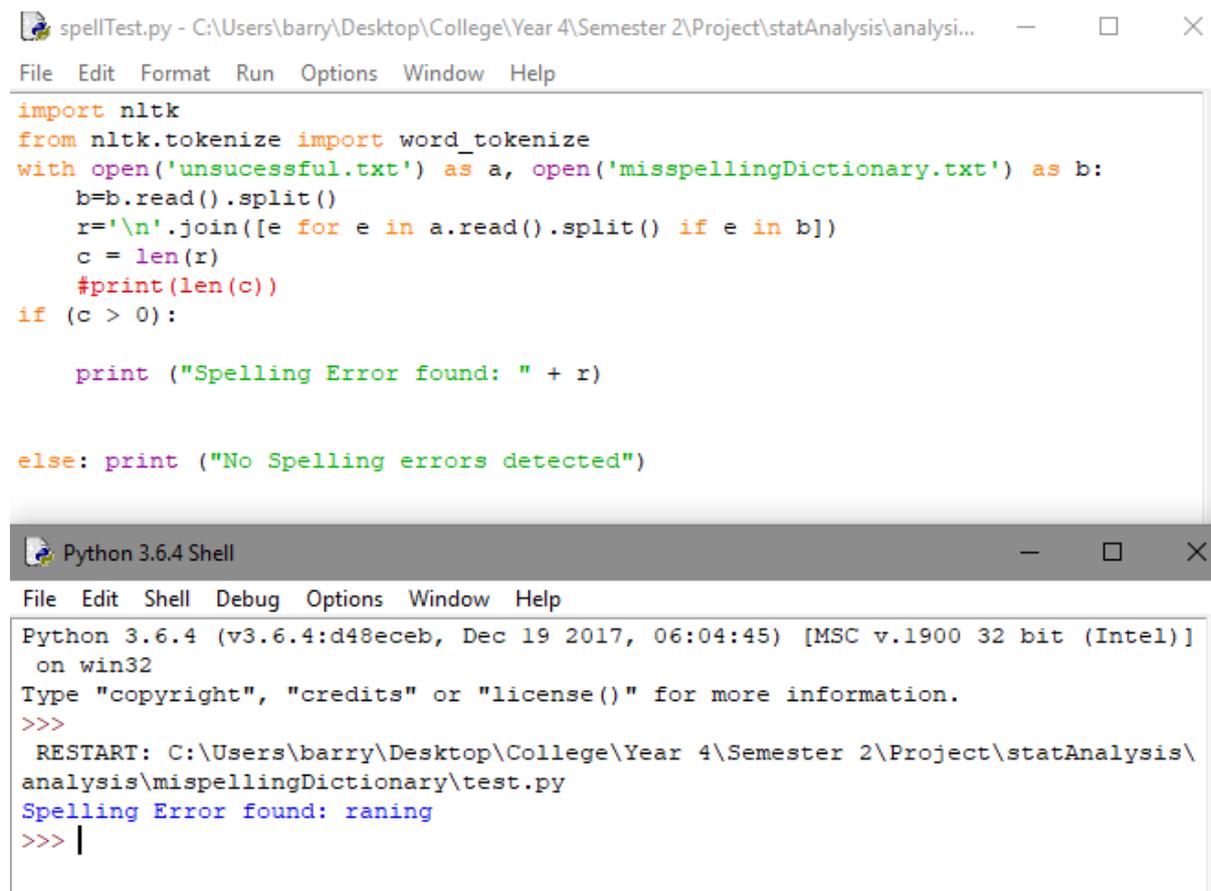
As someone who was formally employed as a Quality Assurance Analyst, I am experienced in creating and executing test scripts on a system. I have created automated tests for this application as well as executing end user testing. As I adopted an agile approach to project, testing occurred throughout the development of the application.

### 5.1. TDD (Test Driven Development)

Test-driven development is an agile development process. It is a process of writing tests of code and ensuring it passes, before implementing it into an application. This form of development provides a detailed guideline of how my functions will be developed when implemented in a framework. Test classes were produced for testing the spelling error detection logic, and the industry analysis logic.

The two tests were tested with multiple files. Once completed, the logic could be adapted to the API

The spelling error detection test code can be found in spellTest.py See the below figure:



```
spellTest.py - C:\Users\barry\Desktop\College\Year 4\Semester 2\Project\statAnalysis\analysi...
File Edit Format Run Options Window Help
import nltk
from nltk.tokenize import word_tokenize
with open('unsuccessful.txt') as a, open('misspellingDictionary.txt') as b:
    b=b.read().split()
    r='\n'.join([e for e in a.read().split() if e in b])
    c = len(r)
    #print(len(c))
if (c > 0):

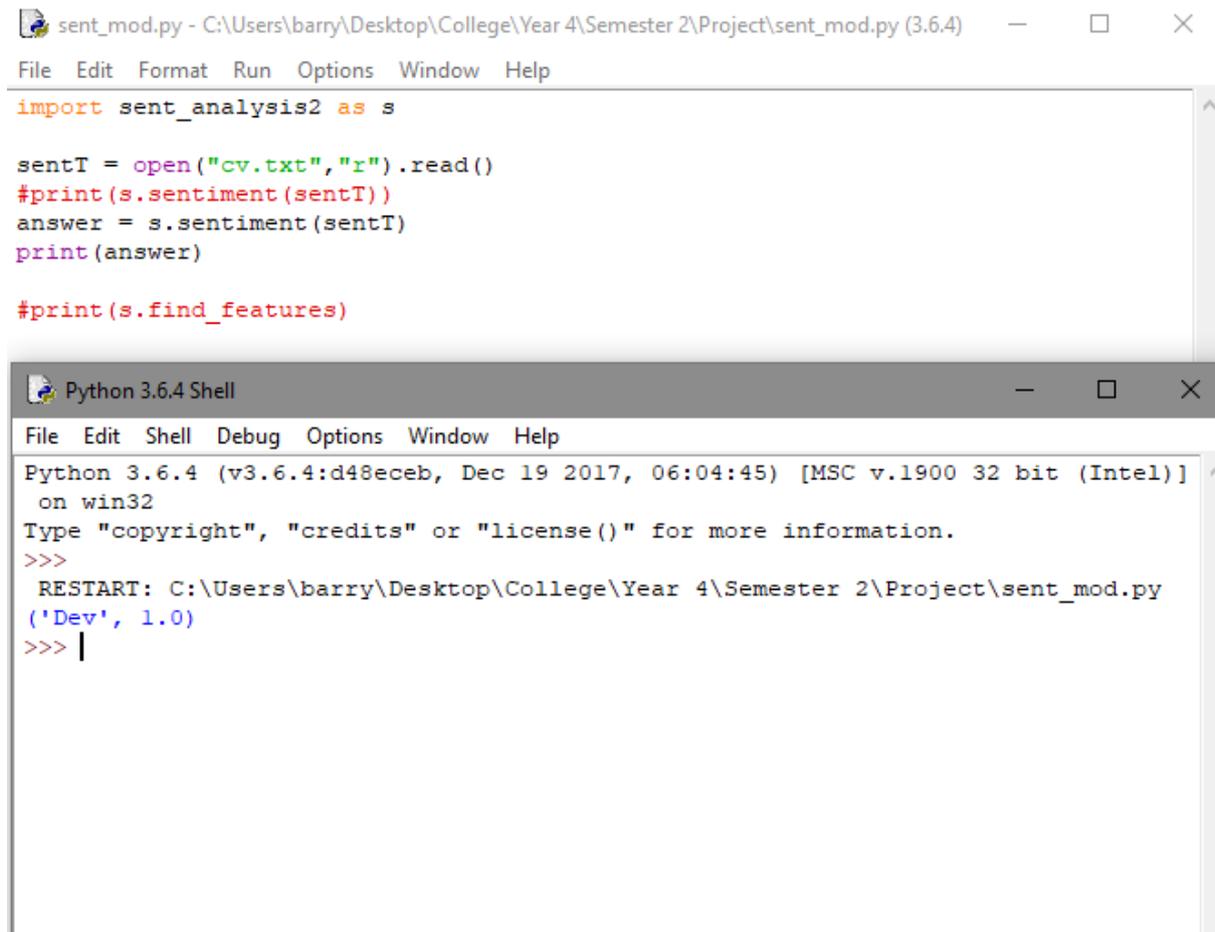
    print ("Spelling Error found: " + r)

else: print ("No Spelling errors detected")

Python 3.6.4 Shell
File Edit Shell Debug Options Window Help
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\Users\barry\Desktop\College\Year 4\Semester 2\Project\statAnalysis\
analysis\misspellingDictionary\test.py
Spelling Error found: raning
>>> |
```

Fig 33: Testing for spelling error detection

The Industry Analysis was also created before being implemented to the Django framework. This can be tested in sent\_mod.py:



The image shows two windows from a Python IDE. The top window is a text editor titled 'sent\_mod.py - C:\Users\barry\Desktop\College\Year 4\Semester 2\Project\sent\_mod.py (3.6.4)'. It contains the following Python code:

```
import sent_analysis2 as s

sentI = open("cv.txt","r").read()
#print(s.sentiment(sentI))
answer = s.sentiment(sentI)
print(answer)

#print(s.find_features)
```

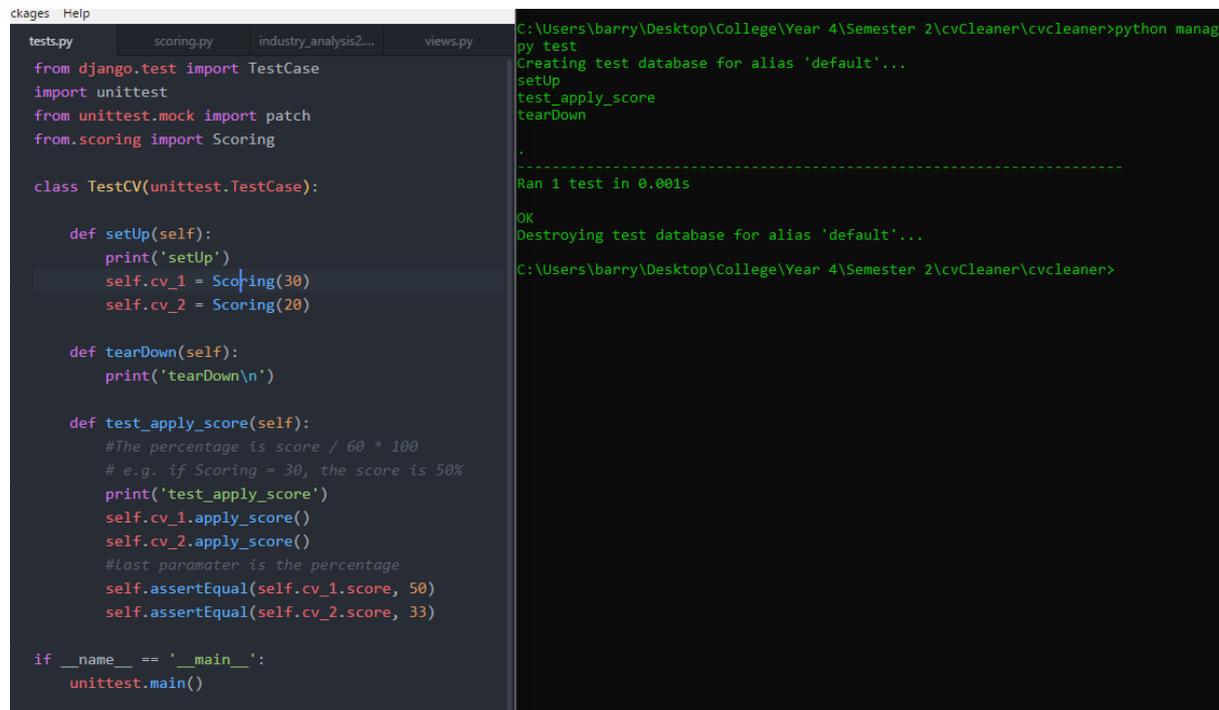
The bottom window is a 'Python 3.6.4 Shell' with the following output:

```
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\Users\barry\Desktop\College\Year 4\Semester 2\Project\sent_mod.py
('Dev', 1.0)
>>> |
```

Fig 34: Testing Industry Detection

## 5.2. Unit Testing

Unit tests were written within the Django framework to test the logic for calculating the score for CVs. This was executed using the tests.py and scoring.py classes. This test allowed me to ensure the logic for the CV percentage calculation was correct. The test was run in the command line with the following command: `python manage.py test`.



```
tests.py | scoring.py | industry_analysis2... | views.py
from django.test import TestCase
import unittest
from unittest.mock import patch
from scoring import Scoring

class TestCV(unittest.TestCase):

    def setUp(self):
        print('setUp')
        self.cv_1 = Scoring(30)
        self.cv_2 = Scoring(20)

    def tearDown(self):
        print('tearDown\n')

    def test_apply_score(self):
        #The percentage is score / 60 * 100
        # e.g. if Scoring = 30, the score is 50%
        print('test_apply_score')
        self.cv_1.apply_score()
        self.cv_2.apply_score()
        #Last parameter is the percentage
        self.assertEqual(self.cv_1.score, 50)
        self.assertEqual(self.cv_2.score, 33)

if __name__ == '__main__':
    unittest.main()
```

```
C:\Users\barry\Desktop\College\Year 4\Semester 2\cvCleaner\cvcleaner>python manage.py test
Creating test database for alias 'default'...
setUp
test_apply_score
tearDown
-----
Ran 1 test in 0.001s

OK
Destroying test database for alias 'default'...
C:\Users\barry\Desktop\College\Year 4\Semester 2\cvCleaner\cvcleaner>
```

*Fig 35: Test CV percentage passing*

The above test passed as the correct parameters were inserted. In the setup function, cv\_1 was given a score of 30. The total score a CV can get is 60. As such, the correct percentage should be 50%. This is assigned in the test\_apply\_score function. If the parameter passed was not 50, the test would fail.

The if statement at the end of test.py allows the examiner to run the test outside of Django.

### 5.3. Usability Testing

It was important to have multiple testers to test the application's usability. It was also important to have external testers, to ensure every area of the app is tested. I gathered five willing participants within CV Cleaner's target audience to test the application's usability. I ensured participants had never seen the application before to avoid bias. The testing occurred under my supervision, however to ensure the integrity of the testing, I did not assist the testers in testing the application. The goal of this testing was to ensure the application meets the user needs, and the application's goals seamlessly. The following techniques were used to test the usability of the application:

1. Five Second Test
2. System Usability Scale

#### 5.3.1. Five Second Test

A five second test is a technique that documents the user's first impression of an application. The user will look at a page on the application for 5 seconds and is then asked questions regarding the user interface. This technique was executed on CV Cleaner's home page by five testers. To ensure the tests were accurate, the testers have never seen the page prior to the test. The following questions were asked after the testers views the page for five seconds:

1. What is the application about?
2. What was the company name?
3. What grabbed your attention?
4. Was there anything you didn't like?

The results of the five second test is available below:

Question	Tester 1	Tester 2	Tester 3	Tester 4	Tester 5
<b>1</b>	Analysing CVs	Analysing and Filtering CVs	Analysing CVs for computer scientists	Scoring CVs	Analysing CVs for Computer Science students
<b>2</b>	CV Cleaner	CV Cleaner	CV Cleaner	CV Cleaner	CV Cleaner
<b>3</b>	The colours	The layout	The layout was slick	The name and the add CV button	The colours scheme stood out
<b>4</b>	Nothing really	No	There should be a filtration form in the nav bar	No	Nothing

These results confirm that the application achieves its goals. All testers immediately knew the name and the purpose of the application. Only one user provided an idea of how to improve the design which was to include a filtration form in the navigation bar. This has now been implemented.

### 5.3.2. System Usability Scale

After the five second test, I provided the testers with the use cases of the application and I asked them to execute them. After completing the use cases, they filled out the SUS.

This technique is a way of statistically measuring usability. It consists of 10 questions which are as follows:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated
6. I thought there was too much inconsistency in this system
7. I would imagine that most people would learn to use this system very quickly
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

These questions were measured on a scale of one to five. One meaning the tester “strongly disagrees” and five meaning the tester “strongly agrees”. The system usability score is calculated by subtracting one from the score of the odd numbered questions and subtracting the value from five for the even numbered. The sum of these values is then multiplied by 2.5, which returns a score out of one hundred.

Question	1	2	3	4	5	6	7	8	9	10	Score
<b>Tester 1</b>	5	1	5	1	5	1	5	1	4	1	97.5
<b>Tester 2</b>	5	1	5	1	5	2	5	1	5	1	97.5
<b>Tester 3</b>	5	1	5	1	4	1	5	1	4	1	96
<b>Tester 4</b>	5	1	5	1	5	1	5	1	5	4	97.5
<b>Tester</b>	4	1	5	2	5	1	5	1	5	4	94.5

The application scored extremely high on the system usability scale. This confirms the application is easily usable and aesthetically pleasing.

Overall, the usability testing was extremely beneficial to me as I received constructive feedback. I have implemented the tester's suggestion to add the filter form into the nav bar, which contributes to the application's usability.

## 6. Conclusion

Planning and developing this app was a complex and time-consuming process. A lot of time was spent researching and learning new technologies to ensure the functionality I proposed could be accomplished. Despite this, I feel this experience has equipped me with vast knowledge and skills that I can apply to a role in the IT industry.

I have utilized many techniques and technologies throughout this project which I feel was beyond what was required as I completed a statistical analysis on top of developing a software application. I have used my knowledge from the course to incorporate emerging technologies that are not elements of the course content (machine learning, natural language processing).

I believe the application possesses commercial potential as it is a unique concept with minimal competitors in this market. The application can expand into new markets with future releases to include more industries, such as the business industry. This would not be difficult as scalability was considered when developing the application.

The use of core coding and data analysis principles such as DRY and referencing code was essential to completing this project within the deadline. The use of Pickling the machine learning algorithms contributed to the performance of the statistical analysis. Developing the project with scalability, security and performance in mind has contributed to the tool's finesse. The application is highly scalable, robust, and minimizes the possibility of cyber-attacks with multiple techniques such as file limitations.

Test Driven Development was a great approach to the development of this project and I would recommend future students to adopt this approach. Having the code and logic planned out before implementing it to the application allowed me to focus my attention on learning the Django framework. Unit testing allowed me to quickly test the equations behind my functions. Usability testing from end users was extremely beneficial as external testing identified a feature that would enhance the application (filtering form on the nav bar). The feature has since been implemented.

Going forward, I would like to expand this application into other markets. I would like to increase the intelligence of the algorithm by adding more types of analysis, such as structure analysis. The application could expand to provide scoring systems for other industries. It could also expand to learn other languages e.g. Spanish or French.

## 7. References

\*Code references are included in class files.

1. Anadiotis, G. (2017). *Predictive analytics and machine learning: A dynamic duo* | ZDNet. [online] ZDNet. Available at: <http://www.zdnet.com/article/predictive-analytics-machine-learning/> [Accessed 28 Nov. 2017].
2. analytics?, H. (2017). *How Machine Learning can boost your predictive analytics?*. [online] Maruti Techlabs. Available at: <https://www.marutitech.com/machine-learning-predictive-analytics/> [Accessed 28 Nov. 2017].
3. Future of Life Institute. (2017). *Benefits & Risks of Artificial Intelligence - Future of Life Institute*. [online] Available at: <https://futureoflife.org/background/benefits-risks-of-artificial-intelligence/> [Accessed 28 Nov. 2017].
4. Harvard Business Review. (2017). *The Business of Artificial Intelligence*. [online] Available at: <https://hbr.org/cover-story/2017/07/the-business-of-artificial-intelligence> [Accessed 28 Nov. 2017].
5. Interview. (2017). *Enquiries - Interview*. [online] Available at: <http://www.interview.ie/enquiries/> [Accessed 28 Nov. 2017].
6. Métais, E., Meziane, F., Saraee, M., Sugumaran, V. and Vadera, S. (n.d.). *Natural Language Processing and Information Systems*.
7. Pereira, F. and Grosz, B. (1994). *Natural language processing*. Cambridge (Mass.): The MIT Press.
8. Perner, P. (n.d.). *Machine learning and data mining in pattern recognition*.
9. Russell, S. and Norvig, P. (2016). *Artificial intelligence*. Boston: Pearson.
10. YouTube. (2017). *sentdex*. [online] Available at: <https://www.youtube.com/channel/UCfzlCWGWYyIQ0aLC5w48gBQ> [Accessed 28 Nov. 2017].

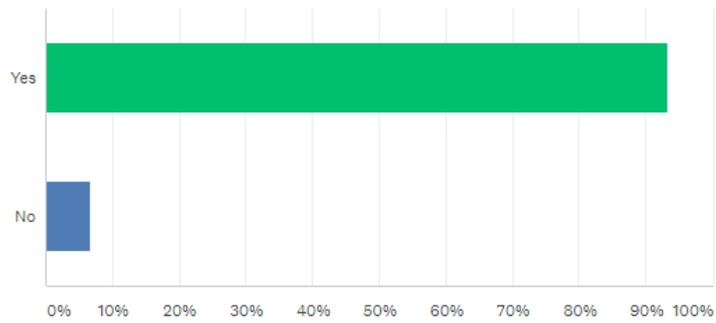
## Appendix A - Survey

Q1

Customize Export

Have you acquired an interview/phone call from sending your CV to said operation before?

Answered: 15 Skipped: 0



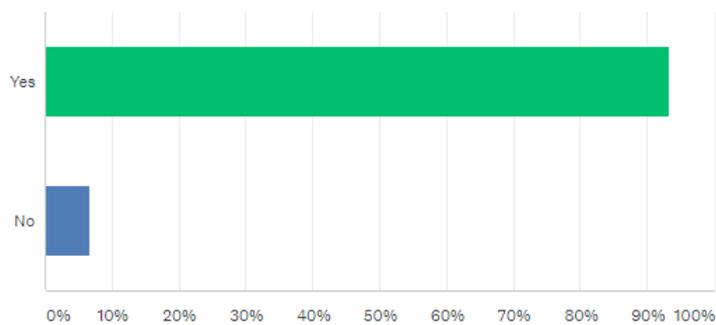
ANSWER CHOICES	RESPONSES
Yes	93.33% 14
No	6.67% 1
Total Respondents: 15	

Q2

Customize Export

Have you applied for jobs and not received a response?

Answered: 15 Skipped: 0



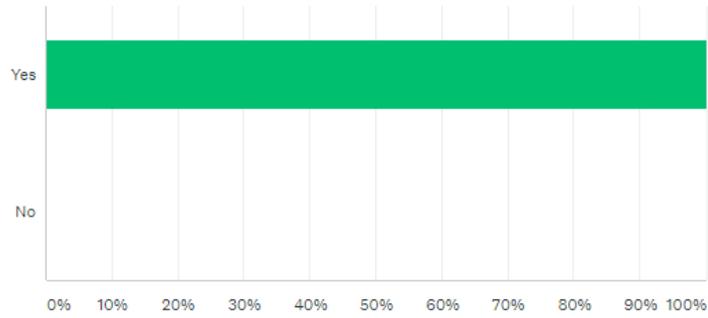
ANSWER CHOICES	RESPONSES
Yes	93.33% 14
No	6.67% 1

Q3

Customize Export

Have you ever received assistance in creating your CV?

Answered: 15 Skipped: 0



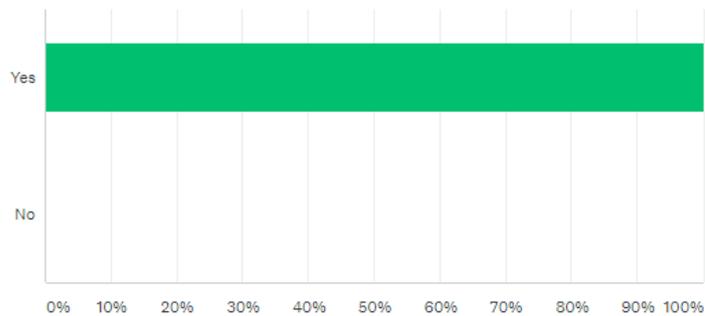
ANSWER CHOICES	RESPONSES	
Yes	100.00%	15
No	0.00%	0
<b>TOTAL</b>		<b>15</b>

Q4

Customize Export

If available, would you use an application that would predict the probability of an employer contacting you based off your CV?

Answered: 15 Skipped: 0



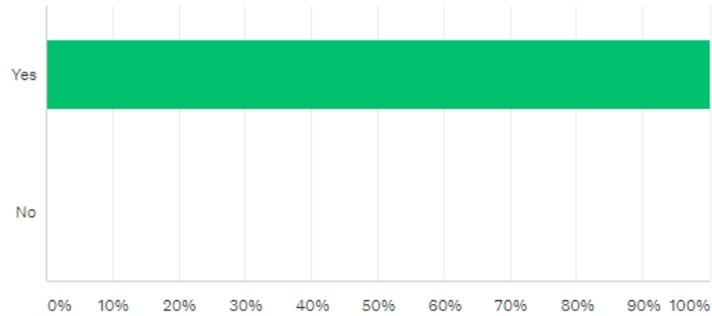
ANSWER CHOICES	RESPONSES	
Yes	100.00%	15
No	0.00%	0
<b>TOTAL</b>		<b>15</b>

Q5

Customize Export

If yes, would you be interested in receiving advice on tailoring your CV to a specific role?

Answered: 15 Skipped: 0



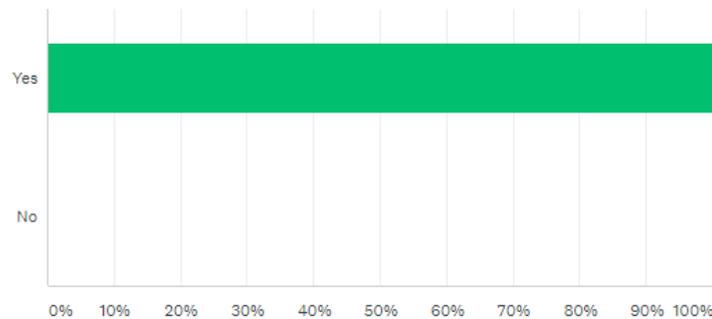
ANSWER CHOICES	RESPONSES	
Yes	100.00%	15
No	0.00%	0
<b>TOTAL</b>		<b>15</b>

Q6

Customize Export

Would you be interested in a visual representation of the analysis? (e.g. graphs based off the AI algorithm)

Answered: 15 Skipped: 0



ANSWER CHOICES	RESPONSES	
Yes	100.00%	15
No	0.00%	0
<b>TOTAL</b>		<b>15</b>

## Appendix B – CV Cleaner Invention Disclosure Form

### NCI INVENTION DECLARATION FORM

#### 1. Title of Invention

CV Cleaner

#### 2. Inventors

Name	School/Research Institute	Affiliation with Institute (i.e. department, student, staff, visitor)	Address, contact phone no., e-mail	% Contribution to the Invention
Barry Lougheed	National College of Ireland	Student	25 Knocksedan Drive, Swords, Co. Dublin 0851339264 <a href="mailto:Barrylougheed43@hotmail.com">Barrylougheed43@hotmail.com</a>	100%

#### 3. Contribution to the Invention

Each contributor/potential inventor should write a paragraph relating to his/her contribution and include a signature and date at the end of the paragraph.

I have researched and developed the entire invention.

Signed: Barry Lougheed

Date: 12/05/2018

#### 4. Description of Invention

(Please highlight the novelty/patentable aspect. Attach extra sheets if necessary including diagrams where appropriate). What is novel, the ‘inventive step’? For more information on patents, please look at <http://www.patentsoffice.ie/en/patents.aspx>

The innovative aspect of this invention is the ability to analyse CVs through artificial intelligence. It is also innovative in the aspect of providing a tool for employers by scoring CVs and filtering the CVs for them.

#### 5. Why is this invention more advantageous than present technology?

What are its novel or unusual features? What problems does it solve? What are the problems associated with these technologies, products or processes? Explain how this invention overcomes these problems (*i.e.* what are its advantages).

It vastly minimizes employer’s time spent evaluating CVs. There is currently no technology that evaluates CVs for employers via automation. The invention can also evaluate student’s CVs via automation.

#### 6. What is the current stage of development / testing of the invention?

First version is complete (updates such as new features will follow).

7. List the names of companies which you think would be interested in using, developing or marketing this invention

Any company that receives many CVs for a role. For example, if Google advertises a role that requires the person to have experience as a “senior java developer”, they can use the application to eliminate CVs that do not meet this requirement. Other companies that could use this tool:

Also, any academic institution that provides software development courses for students would be interested in using this tool as it will help their students increase the quality of their CV. With a better CV, the students have a higher probability of receiving an interview and therefore a higher chance of being offered a job. This will reflect well on the academic institution.

#### 8. Funding Partner(s)

Government Agency & Department	
% Support	
Contract/Grant No.	
Contact Name	
Phone No.	
Address	

Industry or other Sponsor	
% Support	
Contract/Grant No.	

Contact Name	
Phone No.	
Address	

9. Where was the research carried out?

All research was carried out on my personal laptop from home on the internet.

10. What is the potential commercial application of this invention?

The invention has huge commercial potential. The invention can be sold as an individual tool to each company and it can be tailored to their requirements.

11. Was there transfer of any materials/information to or from other institutions regarding this invention?

If so please give details and provide signed agreements where relevant.

--

12. Have any third parties any rights to this invention?

If yes, give names and addresses and a brief explanation of involvement.

13. Are there any existing or planned disclosures regarding this invention?

Please give details.

14. Has any patent application been made? Yes/No

If yes, give date: \_\_\_\_\_ Application No.: \_\_\_\_\_

Name of patent agent: \_\_\_\_\_

Please supply copy of specification.

15. Is a model or prototype available? Has the invention been demonstrated practically?

Yes.

**I/we acknowledge that I/we have read, understood and agree with this form and the Institute's *Intellectual Property and Procedures* and that all the information provided in this disclosure is complete and correct.**

**I/we shall take all reasonable precautions to protect the integrity and confidentiality of the IP in question.**

Inventor: Barry Lougheed

Signature: Barry Lougheed

Date: 12/05/2018

## Appendix C - Monthly Journals

### September

I had my project pitch today. This involved me going into a dragon's den environment with Lisa Murphy, Frances Sheridan and Michael Bradford. I pitched my idea for my fourth-year project to them. My idea was a mobile application that allows users to offer labour jobs to carpenters, plumbers, builders, roofers, painters etc. in auction form. Labourers will have an account set up and will receive notification for their field of work providing it is close by. They will then post a price in the auction for the job. The user who posts the auction can then choose which labourer to do the job at the agreed price. With some constructive criticism and feedback, the project was ultimately approved.

I am yet to come up with a name for the application, but I have fleshed out what the project will look like in low fidelity. I have to consider all of the following technologies while creating my application:

1. Web/Mobile? Mobile
2. C#/Java? Java
3. Text editor – android studio
4. Login functions (security)
5. GPS?
6. Emails/Firebase (notifications)
7. Messenger?
8. Cloud hosted?
9. Database integration?
10. Google static api?
11. Documenting when offline?
12. Concurrency

## October

After receiving feedback from my project supervisor and peers, I had decided to change my project idea that I had originally pitched. My original idea was to create a mobile application that allows users to post adverts for jobs for tradesmen. My main reason for changing is due to this being done before in the form of a web application. There was a lot of competition and the application was more of a good business idea, rather than a project that can showcase my computing skills. As such, I decided to create a machine learning, deep neural network that allows users to identify infections on their skin via an image processor.

I have completed the project proposal and uploaded it to moodle. Within the proposal, I have outlined my objectives for the project, and the background into me choosing to do this as my project. I have explained my technical approach and details, which contains what technologies I will use as well as how I will manage the project. There is also a gantt chart to help me visualise the time and cost management of the project, aswell as an evaluation to ensure I keep testing the project in an agile approach.

## November

The first week of November I began to flesh out my new project idea. I like the idea as well as the computing skills I can execute. After more research, I have a few concerns regarding this idea. I've a meeting with Frances in the second week so I will discuss them with her. My main concerns are the following:

1. I would need a huge dataset of infections to be able to train a neural network. This means I'd have to have thousands of pictures of images I can label eczema, wart, etc. If the dataset isn't large enough, the image recognition won't be accurate.
2. Acquiring the dataset will be time consuming, which I won't get any marks for.
3. I originally thought the neural network would be very complex, but it turns out there is an API that does most of the work. I don't think this will fly for a final year project.

I met with Frances this week and told her my concerns. She agrees with what I'm saying and called Irina over for another perspective. Irina flat out said I'll have to make it way more complex to receive a 1:1, and it would be very hard to accurately distinguish similar skin infections. While the meeting was very productive, I need to think long and hard whether to get a new project or keep this one. The mid-point report is due in 2 weeks, so I needed to sort this, but I didn't. I'm starting to worry now.

I've pretty much ruled out this project idea. This is my second one that I've decided not to go ahead with. I think if I keep being too much of a perfectionist I'm gonna end up missing deadlines. I have another meeting with Frances on the 24<sup>th</sup> of November. We plan to just sit down for an hour, pick a project and just go with it. I have to at this stage as the mid-point report is literally due in less than a week.

The meeting with Frances was great. I now finally have a solid project idea. Frances basically asked questions on what I'm technologies or fields I'm interested in and we then brainstormed. I told her I

want to do something relating to artificial intelligence as I believe it's the future and I'm really enjoying the AI class. I got 91% in the chess assignment CA1, so I figured I'm pretty decent at coding AI. We were going through the lecturers' ideas and an idea struck me. I was researching another CA this week regarding private planning on houses. E.g. building a conservatory in a garden. From research, I understood how the process of seeking planning permission is a hassle. It then sparked an idea for me to create an application that can predict the probability of whether you'll get planning permission or not. It seemed like a decent idea. We wrote it down and kept brainstorming. From this, I thought it would be more relative to predict the probability of a student receiving an interview based purely on their CV. Frances preferred the planning permission prediction, but I preferred the CV as it was more relative to me and I felt it would be easier to obtain the data. I now have a project that has potential to get a first. I also have to re-write my proposal as the last one useless.. on top of that I've to upload the midpoint report in 6 days. I'll have to knuckle down.

This was an eventful week. I've spent at least 30 hours solely on researching and writing this technical report. It's 7k words long and I'm actually really proud I got the work uploaded on time. I think it's good work also. Now that I've done thorough research on the project, I'm looking forward to presenting my work for this presentation. I can imagine Frances will be impressed with the amount of work I've put into this considering I only had the project idea last Friday.

## December

This month is gonna be so rough. I've my Project presentation, Chess project, Mobile project, Data app project, and Web API project all due this month. I'm stressed just looking at all this work ahead.

I completed the project presentation on the 4<sup>th</sup>. Just my luck to be one of the first to do it. I was a bit pissed off the way we didn't have a deadline to upload the presentations. It meant people that were a couple of days after my presentation had more time to finish theirs. Ah well, I think it went really well. Frances and Lisa were quite impressed with the amount of work I completed in such a small-time frame. It's my own fault at the end of the day though so it's not as if I'm gonna get more marks for changing my project idea twice.

My life is now coding my AI and API project. Caffeine is fuelling me at this stage. I've decided to leave my Data App and Mobile project til last for different reasons. I've done really well in Data Application CA's so I'll basically pass of a project getting 10% and I'll do mobile project in a day. I find mobile app development really easy. It makes the most sense logically as I'm under so much pressure.

I wasn't able to write into this journal until I finished those projects. That process really took a toll on me. I'm confident I'll do decent in all my projects tbh. I slept for 14 hours after completing my mobile project. I'm gonna take a day off then start studying for the exams.

## January

Now that the holidays are over, I'm back in the groove of studying. I've only two exams whereas the other streams have at least 3 so I'm happy with that. It's almost fair considering the amount of work we had just before Christmas.

I got a couple of results back from my projects. I got 69% in the project midpoint. It's not bad considering I did it in less than a week. I'm content with it but disappointed in myself for not getting the first. I should have been more prepared regarding a project idea.

I got a first in my Web API and AI project which is really good news. I'm delighted with that. I don't need much to pass the AI module now.

Thank god those exams were over. They were terrible. Some of the questions on that AI paper were ridiculous. We were asked a mandatory 20-mark question that appeared in one slide on a moodle powerpoint in week 1. It's not even relative to my degree. We should be getting asked questions relative to "Computer-Science", not the philosophy or history on how one guy swayed humans from thinking consciousness was in the heart, to the brain.

At least my strategic management exam went relatively ok. I still think these exams are gonna bring my grades down massively.

I'm finally able to get back into working on my project. This semester doesn't seem as heavy as last one thankfully. This week I pretty much interviewed two former colleagues of mine regarding what they look for in CV's. Both of them have been hiring people for years. They gave me really good insight into what rules I should have in my algorithm. I can incorporate their advice as well as the results from the statistical analysis on the datasets I acquired. I have two different sets of CV's. whether they received an interview or not.

## February

I've got a decent grasp on the NLTK. I think I'm gonna use scikit learn for part of the machine learning aspect of my analysis. I completed my Usability Design CA1 this week. I'm really happy with the work.

I cleaned the datasets, so I can now undergo the analysis. I have an idea of what I'm gonna find but I'm interested to see if there's anything unexpected.

I've completed 1/5 sections of the analysis. I completed the key word analysis, it was really interesting. This basically seen how many times words occurred in the successful datasets versus the unsuccessful ones. I got to showcase machine learning in this. I found expected information like getting a 1:1 was favourable, but I found a lot of new information such as UCD being more favourable than any other college. I've emailed Frances with an update of my project progress

I got the word count analysis done this week. This was pretty simple. It involved me compiling the CV's from each dataset into one file, categorized by their set. So, the successful software CV's made one file, the unsuccessful software CV's made another file and so on. I was then able to get the average word count of these sets by finding the length of this compiled data. I can now incorporate this as a rule into my algorithm. It will be something like the closer the user is to the average word count of the successful CV's, the more points they will get.

## March

I spent a couple of days tackling the grammar detection part of my analysis. After thorough research, I decided to remove it from my project scope. To detect grammar, each sentence is parsed through a

tree, which is extremely costly. If this is implemented into my application, it will add minutes onto returning a result to a user. The user won't be bothered if it takes too long, which it will already take longer than a normal function on an application.

I got the spelling error detection complete. This is tricky to tackle in the context of my application as many technologies will be used that won't exist in an API such as "xml" or "javaScript". Even names would be flagged. There is another issue with existing API's is that most of them correct the spelling, whereas I just want to find if there is a spelling error in the file.

As such, I acquired a data dump of over 8k common misspellings. It had the format of "correctspelling:" <incorrectspellings> so some cleaning was involved to remove the correct ones.

Now I can run the user's CV against this file and see if a word exists in both of them. If it does, this will equal to a spelling error.

Obviously this won't be 100% accurate, but I can run tests on it to provide a statistic to prove it is a reliable way to detect this. This will be pretty solid to add to my unit testing section. I updated Frances on my progress.

My Distributed Systems test was postponed to this week. I spent all week studying for this test. I did the test on Thursday, then had to complete Usability Design CA2 before Friday eve. I got the CA completed and I'm confident I'll do well.

I got my Usability Design CA1 results back. I got 80% so I'm delighted. On top of that, I completed the code for the final section of my analysis which is the industry analysis. This basically determines whether a CV is suited to a IT or software development role. Similarly to the spelling error detection, I'm gonna run it a few times to calculate the accuracy of it.

