

National College of Ireland

Business Information Systems

2020/2021

Abigail Boyle

X17323943

x17323943@student.ncirl.ie



CONNECT2

4th Year Software Engineering Project

Technical Report

Executive Summary

This report was submitted in May 2021 as part of the final year software engineering project for Business Information Systems. In this report, there is a detailed explanation outlining the motivation behind and implementation of a new recruitment software, Connect2. This project focuses on the issues around recruitment discrimination in Ireland and proposes an idea for how technology could be used to tackle this issue. Furthermore, this report outlines how this idea was implemented to produce a web application candidates and recruiters can use for their hiring needs. It explains the functional requirements of the web application and how such requirements were achieved. It also discusses the testing stages the project went through to validate performance. Finally, further research and development for this project are discussed in the final sections.

1. Background

The idea for Connect2 sparked from the discussion around discrimination in recruitment, specifically in Ireland. In 2009, an experiment was conducted where researchers recorded employers' responses to job applications from candidates who had identical attributes except their ethnic or national origin. It was found that that job applicants with Irish names were twice more likely to be invited for interview than those with identifiably non-Irish names, despite submitting identical CVs (Lunn and Quinn, 2009). It must be noted that the study only investigated the first stage of the recruitment process, interview invitations. The authors comment on how if candidates had completed further stages of the recruitment processes, the likelihood of being faced with discrimination would be even higher (Lunn and Quinn, 2009).

In a 2019 survey, it was found that 10.6% of persons from non-white ethnic background reported discrimination while looking for work in Ireland. It was also reported that one in twelve of persons who identify as LGBTI+ experienced discrimination prior to employment (Central Statistics Office Ireland, 2019). In the same year, Studio Graphene conducted an online survey surrounding the gender bias in STEM-related career fields. It uncovered that half of women (49%) have experienced some form of discrimination in the workplace and that 54% are in support of anonymising CVs during the recruitment process to prevent bias prior to interview (Agarwal,2020).

Action has been taken by leaders in Ireland to combat workplace discrimination with the Irish Employment Equality Acts 1998-2015 prohibiting a range of workplace discrimination, including legislation around the hiring and interview process. However, recruitment discrimination problems remain with cognitive biases being identified as a prime cause. There are 140 possible cognitive biases one can have, ranging from confirmation bias (interpreting new information as a vindication of prior beliefs) or anchoring (relying on the initial piece of information whilst making decisions) (Pozniak, 2020). Humans have limited control over their ability to manage these biases, therefore when a recruiter is asked to review multiple CVs within a short time frame, it is more than likely that these biases will affect the final choice of candidates.

In short, it is not determinable if recruiters are being consciously or unconsciously biased in their choice of candidates. We cannot understand the exact reasoning behind the decisions made but we can see from the statistics that high rates of individuals from minority groups are suffering from recruitment discrimination. From this revelation, the inspiration for Connect2 began. It was questioned if there could be a technology which directly tackled the problem and attempted to alleviate it.

When one discovers a problem, the processes which occur before the problem are investigated to try and determine the root of the issue. When it comes to recruitment, applicant tracking software systems (ATS) have become the norm. Recruiters typically issue job listings to their company website or third-party employment website (Indeed, Monster Jobs etc) and applications are passed to custom ATS systems implemented by the company. These pieces of software are utilised to accept, rank, and manage applications made to a job listing. The employers specify which skills, experiences, and education they are in search of, and the ATS software does the work.

Considering that job listings can attract thousands of applications, ATS software popularity is understandable. ATS systems are successful in reducing the time required to screen thousands of applications but with such statistics as previously outlined, it is evident that these ATS systems are in some form enabling recruitment discrimination. This further inspired the idea for Connect2 as it was examined whether it would be possible to build an applicant tracking software system which was specifically designed to protect against human cognitive biases but capable of finding the recruiter the type of candidate they are searching for. Additionally, this ATS software could promise candidates their application would be solely ranked based on **how they are** over who they are. It was further questioned if this ATS software could be implemented on an employment website, meaning all applicants and similarly all recruiters would use the same centralised ATS software.

From such, the idea for Connect2 was drawn. The project focused on building a website which acted as an employment website for candidates and recruiter. Recruiters could post job listings whilst candidates could apply. Connect2 would be able to understand what the employer is searching for and find ideal candidates. It would have its own built in ATS software designed to protect against cognitive biases and an application management system for recruiters to use. Most importantly, it would employees an equal opportunities job application service.

2. Aims:

Connect2 aims to offer users a range of features to help with their recruitment needs. Since Connect2 is focused on reducing recruitment discrimination, there are several distinct characteristics of the website which set it apart from other employment websites available on the market:

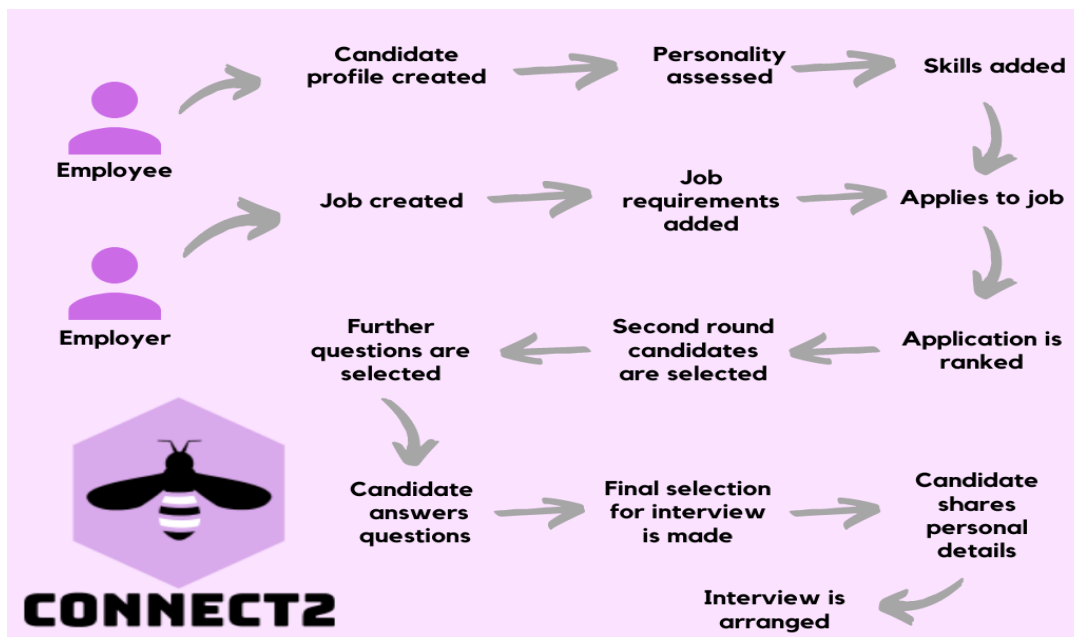


Figure: Diagram outlining the flow of actions user take on Connect2

There are four distinct groups of functions which collectively make up Connect2. They are:

1. An employment website for recruiters to post jobs and candidates can apply directly.
2. Built in applicant ranking software.
3. Application management system recruiters can use.
4. Built in direct messaging and interview scheduling system.

2.1. Employment Website Aims

Connect2 is designed as an employment website where users can create accounts as either candidates or recruiters. Users create profiles with their personal/company details. User profiles of the employee users are to be completely anonymised and kept private from other users. These profiles are used to create applications to jobs but the personal details of the candidate are kept private, even from employers. Candidate users will include their skills and work experience on their profile but more specific details, such as names of companies they have worked for, will not be required.

Connect2 will offer the following features to users in the form of an employment website:

1. Ability to create an account as either a recruiter or candidate.
2. Candidate users can upload details of their CV to their user profile, personal and more specific details are kept private.
3. Recruiters can create job listings with the required skills and experience specified.
4. Candidates can view custom recommended jobs based on their listed skills and interests.
5. Candidates can apply directly to jobs using their user profile and unique cover letter.
6. Recruiters receive applications.

Candidate users will also be asked to complete a unique personality test based on a new framework this project will introduce. The personality framework is outlined in the next section.

2.1.1. Personality Frameworks (MBTI and Keirsey)

The animal work-type framework is a four-personality group framework derived from the existing Myers & Briggs Foundation MBTI personality framework and the Keirsey Temperament Sorter. The MBTI personality type reveals how one sees and interacts with the world. It provides a strong foundation for personal growth and development, underpinning enhanced personal effectiveness. In terms of the professional world, MBTI types can offer insight into what a person can contribute to a company and how they may act in team environments. Reports show that the MBTI framework is used by nearly 90% of Fortune 100 companies in the hiring process or as a team-building exercise (Colossus Media Group, 2019). When used correctly, MBTI type can give employers a broad idea of a person and how they will perform in certain role.

The MBTI framework consists of 16 personality types. The MBTI framework uses questions to understand how an individual scores in 4 categories:

1. Are they an introvert or extrovert? Extrovert (E) vs Introvert (I)
2. How do they process information? Sensing (S) vs Intuition (N)
3. How do they make decisions? Thinking (T) vs Feeling (F)

4. How do they organise their world? Perceiving (P) vs Judging (J)

Once the questionnaire is complete, the persons preferred type is determined in each of the four categories, resulting in a four-letter code. For example, an individual who scored highly in extraversion, sensing, thinking and judging would be classified as an ESTJ.

The Keirsey Temperament Sorter is similar to the MBTI framework; however, it uses just 4 temperament behaviours. Keirsey was inspired by Hippocrates and Plotes and their ancient study of personality. In his works, Keirsey used the names given by Plato: Artisan (iconic), Guardian (pistic), Idealist (noetic), and Rational (dianoetic). These groups were further split into two types, each type with two role variants. The resulting 16 types match with the 16 personality types outlined by Briggs and Myers.

	Temperament	Role	Role Variant	
Concrete or Abstract?	Cooperative or Utilitarian?	Informative or Directive?	Expressive or Attentive ?	
Observant (S)	Guardian (SJ) <i>Logistical</i>	Conservator (SFJ) <i>Supporting</i>	Provider (ESFJ): <i>Supplying</i> Protector (ISFJ): <i>Securing</i>	
		Administrator (STJ) <i>Regulating</i>	Supervisor (ESTJ): <i>Enforcing</i> Inspector (ISTJ): <i>Certifying</i>	
		Artisan (SP) <i>Tactical</i>	Entertainer (SFP) <i>Improvising</i>	Performer (ESFP): <i>Demonstrating</i> Composer (ISFP): <i>Synthesizing</i>
			Operator (STP) <i>Expediting</i>	Promoter (ESTP): <i>Persuading</i> Crafter (ISTP): <i>Instrumenting</i>
	Introspective (N)	Idealist (NF) <i>Diplomatic</i>	Advocate (NFP) <i>Mediating</i>	Champion (ENFP): <i>Motivating</i> Healer (INFP): <i>Conciliating</i>
			Mentor (NFJ) <i>Developing</i>	Teacher (ENFJ): <i>Educating</i> Counselor (INFJ): <i>Guiding</i>
Rational (NT) <i>Strategic</i>			Engineer (NTP) <i>Constructing</i>	Inventor (ENTP): <i>Devising</i> Architect (INTP): <i>Designing</i>
			Coordinator (NTJ) <i>Arranging</i>	Fieldmarshal (ENTJ): <i>Mobilizing</i> Mastermind (INTJ): <i>Entailing</i>

Figure: Chart explaining how the personality types are broken down into 16. The colours represent the 4 temperaments proposed by Keirsey. (Zalcastin, 2013)

2.1.2. Questionnaire:

Connect2 will use these existing frameworks to create its own personality types. Employee users will be asked to complete a questionnaire which will categorise them into one of the 16 MBTI types. Once they have received their type, they will be placed into one of the four Keirsey types, which have been altered by this project.

The Connect2 project created its own custom questionnaire. The content employee users are presented with are of similar design to the existing MBTI questionnaire, however they have been altered slightly to focus more specifically on behaviour at work. The questionnaire consists of sentences and users select how highly they agree or disagree with the statement.

Examples from the questionnaire:

- You feel more energised after spending time with people. (Extraversion)
- You find it difficult to approach others for help at work unless you are very familiar with them. (Introversion)
- You are particularly skilled at being detail oriented. (Sensory)
- You enjoy problems which require thinking of new ways to solve something. (Intuition)
- You place a lot of emphasis on the effect a decision may have on others before making it. (Feeling)
- You feel more comfortable using logic to make decisions rather than emotion. (Thinking)
- You consider your organisation skills to be one of your top strengths. (Judgement)
- You would rather improvise than plan. (Perceiving)

2.1.3. Use of Four Animals

Four animals (Fox, Elephant, Beaver and Owl) have been selected to represent the four temperament types presented by Keirsey for Connect2. The animal groups represent these temperaments in a working environment, rather than how these temperaments act as parents, friends, partners etc.

The Owl represents the NT temperament, accounting for ENTP, INTP, ENTJ and INTJ personalities from the MBTI framework. Owl people are known for being extremely logical in their thinking and attracted to careers where they are free to explore their ideas independently and use these ideas to solve complexed problems. They have a talent for conceptualising and are strategic planners/ goal setters. Owls begin building mechanisms as young children and as adults they will turn their inventiveness to many kinds of organizations, social as well as mechanical. With their innovative, entrepreneurial spirit, Owls are always on the lookout for a better way, always eyeing new projects, new enterprises, new processes.

The Beaver represents the SJ temperament which in MBTI are ESFJ, ESTJ, ISTJ and ISFJ. Beavers are known for being loyal, patient, and gentle. They are often the glue that holds a team together as they seek to nurture and support others. Beavers take pride in being dependable and trustworthy; if there's a job to be done, they can be counted on to put their shoulder to the wheel. Beavers are informative and reactive. They tend to work to spot problems before they occur and act in advance. When they communicate with others, they tend to direct them. They are highly motivated to use their organizational skills and follow procedures as for Beavers in the long run loyalty, discipline, and teamwork get the job done right.

Foxes represent the SP temperament which in MBTI are ISFP, ISTP, ESFP and ESTP. Foxes are known for being action oriented and able to think quickly on their feet. Due to having an innate ability to stay in the moment, Foxes are natural problem solvers, finding solutions to crises quickly. Foxes have a special sense of immediate needs and can use their resourcefulness to find solutions. Foxes are bursting with creativity meaning they are skilled at turning creative ideas into reality. Foxes tend to wind up in technical and creative fields as they provide outlets for their creativity and curiosity about how the world works.

Finally, Elephants represent the NF temperament which in MBTI is INFJ, INFP, ENFP and ENFJ. Elephants want to understand people, they see the potential and possibilities in themselves and

others and wish to collaborate to progress. They are team builders and through encouragement and leading by example, they strive to make their world a better place. They are deeply sensitive to flaws in a system, particularly flaws that infringe on a human's rights and values and can come up with innovative solutions to solve these problems.

(Kiersey, 2021) (Kroeger, Thuesen and Rutledge, 2002) (Zalcastin, 2013)



Figure: The four animal personality types with their symbols.

2.1.4. Personality Classification Model

Users save their animal work-type to their profile once they have been categorised. Similarly, when employers post a job listing, the job is be categorised into an animal work-type too. The difference however is that employers are asked to input text data regarding the specifics of the job, requirements, responsibilities etc. Connect2 runs the data through a **custom machine learning classification model** which will predict the animal work-type.

The machine learning model was created using compiled online data of the MBTI and Keirsey types. Multiple sources were used to compile descriptive information on the different types and a custom CSV file was created to store the data. Originally, a pre-existing dataset from Kaggle was to be used to train the model however its content was largely unsuitable for training the classification model used in this project.

The dataset used to train the model consisted of 20,916 rows of data, each row representing one of four temperaments followed by descriptive information. Building the dataset consisted of web-scraping for categorical data on the four types. The sources used for this dataset are listed in the appendix.

The Connect2 ML model uses properties of the natural language processing kit to analyse the employer inputted data and make a prediction on which animal work type is best suited. Connect2 introduces the idea of using machine learning for understanding job postings to help recruiters from their own cognitive biases. Recruiters input the job details and Connect2 decides what they are looking

for. More details on the creation and use of the personality prediction model is specified in the implementation section.

2.1. Job/Applicant Ranking Software

Connect2 features built in job and application ranking systems. The job ranking software takes into consideration the employee users' skills, experience and location and makes recommendations on what jobs they should apply to. When the employee user visits the "Find Job" page there are several different recommendations made:

1. Jobs which share common skills or experiences with the employee users are recommended. Jobs with more matched attributes are listed first.
2. The most in demand skill or experience of an employee user in their city will be found. Jobs which require this skill will be recommended because if there is a high demand for the skill in that location, the likelihood of recruitment is greater.
3. Jobs which match the employee users work animal type.
4. The employee user can use a filter system to find other jobs.

The applicant ranking software is used to rank the applicants which are made to a job. When employer users view the applications, they will appear in descending order. Applications with higher rates of matched skills, experience or personality will appear first. A list of common attributes between the job requirements and application is also outlined for the user.

2.2. Application Management System

Connect2 has an application management system which employers can use to manage applications made to their jobs.

1. The system features the ability to highlight or delete applications.
2. Employers can view the report from the ranking system and view the unique cover letter for each application.
3. Favoured applications can be highlighted whilst others can be deleted.
4. Employers can create new rounds, which will forward the highlighted applications to another round and delete the remaining.
5. Employers can create a final round. This will forward all remaining applications to an interview round.

2.3. Messaging, Interview Scheduling and Notification Systems

Connect2 has a built-in direct messaging system which allows recruiters and applicants to interact with one another. Employers can send direct message from the application management system if they have further questions for the candidate. Furthermore, when an application is forwarded to an interview round, an automatic message is sent to the candidate stating they have been selected for interview and to await further details.

Connect2 also can schedule interviews with selected candidates. When applications have been forwarded to the interview round, employers select the time, date, and location of the interview. When the employer issues the interview, a request is sent to the candidate with the details.

Interview requests can be either accepted or rejected by the candidate. If the employee users choose to accept the interview request, an automatic link to the applicants personal Connect2 user profile will be issued to the recruiter. **It is only when an interview has been issued and accepted that the applicants' personal details are revealed.**

Lastly, when new applications, messages or interview requests are sent, the recipient receives a notification. The notification states the update with a link that leads the user to it.

3. Technologies

3.1. Development Technologies

- Visual Studio Code

3.2. Frontend Technologies

These technologies will be used to produce the visual front-end of the web application. The front-end of Connect2 will be a collection of pages to take the users through the various processes and functions.

- HTML
- CSS
- Bootstrap
- JavaScript
- JQuery

3.3. Backend Technologies

These technologies will be used to connect the data to the front-end of Connect2. Django is a Python-based free and open-source web framework which will be used to create an Admin dashboard.

- Python
- Django

3.4. Databases

These technologies will provide a relation database to store user data.

- SQL

3.5. Machine Learning

These technologies will be used to work with human language data and implement machine learning algorithms to analyse the user job description to predict a personality type.

- Python
- Pandas
- Scikit-Learn
- Natural Language Processing Toolkit (NLTK)
- Numpy

3.6. Version Control

Used to maintain versions of Connect2 as development continues.

- Github
- Git

3.7. Testing Technologies

Used to test the applications functionality.

- PyTest
- Visual Studio Code

4. Structure

For the remaining parts of this report, the structure is as follows.

Section 5 will outline the key stakeholders of Connect2, the Use Case diagram and the Functional Requirements. There are a total of 14 functional requirements which collectively cover Connect2's functionality.

Section 6 will explain the data requirements.

Section 7 explains the User, Environmental and Usability Requirements.

Section 8 will outline Connect2's system design and architecture.

Section 9 will discuss the technical implementation of Connect2's key features.

Section 10 contains images describing the graphical user interface.

Section 11 will illustrate how Connect2's functionality was tested using several different types of testing styles.

Section 12 will cover the system evaluation and project conclusions.

Section 13 will discuss potential further research the project could take.

Section 14 include the references.

Section 15 includes the Project Proposal. Section 16 contains the reflective journals from September 2020 – May 2021.

5. Functional Requirements

Connect2 will have three stakeholders:

Employee Users – denoted with **green**

Employer Users – denoted with **red**

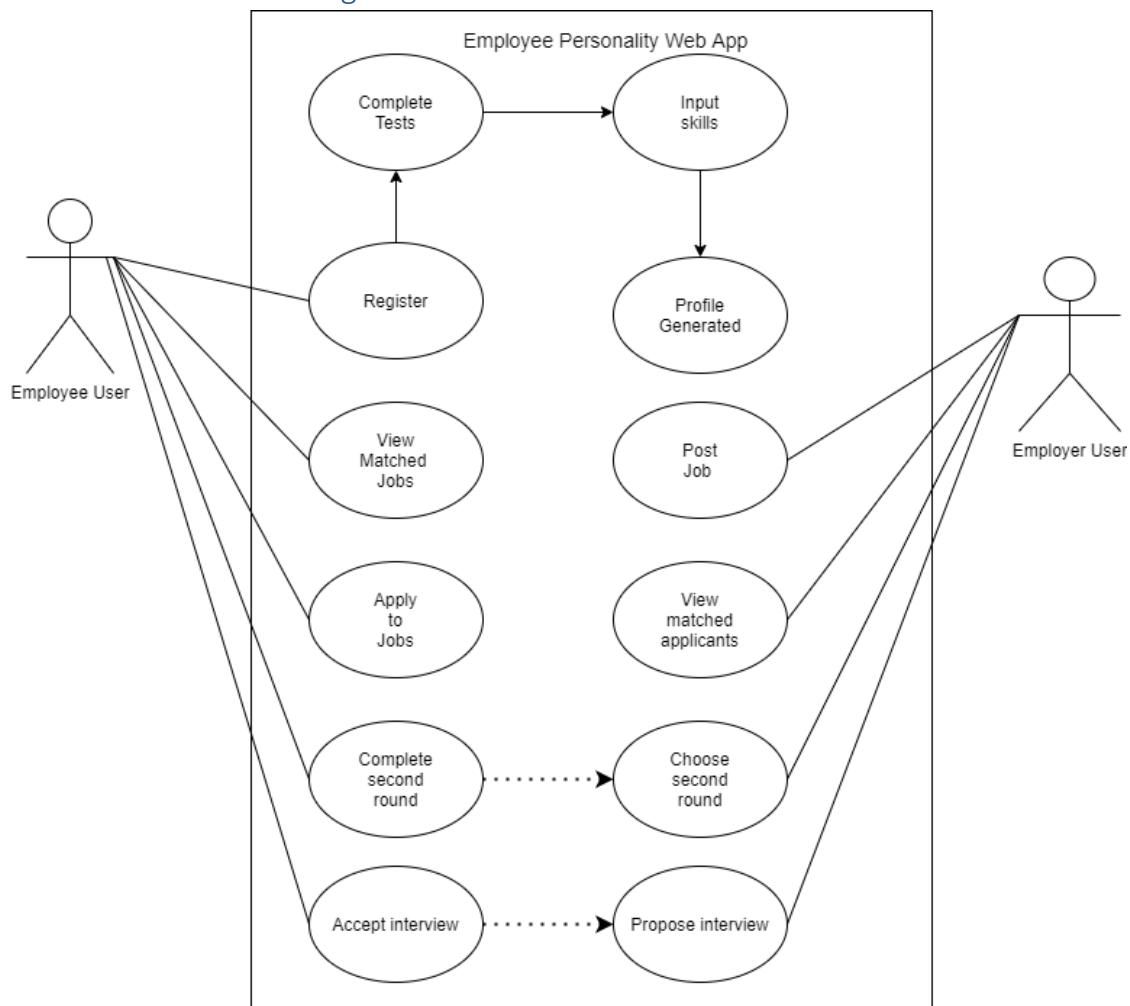
Admin – denoted with **blue**

5.1 List of Functional requirements

1. Users must create an account
2. Users must sign in to their account to access the application

3. **Employee** users can edit their personal details, knowledge area and skills section.
4. System enables personality testing for **employee** users
5. **Employer** users can post jobs
6. **Employee** users can view available jobs in an order of highly matched to less matched
7. **Employee** users can apply to job listings of their choice with their professional profile
8. System ranks all **employee** users who have applied to the position against one another
9. **Employer** user selects applications to advance/eliminate
10. **Employer** can select **employee** users to interview and request personal details
11. **Employee** user can approve the request for personal details to arrange an interview
12. **Admin** users can access the admin page and manipulate user data with CRUD tools
13. Users can delete their account

5.1. Use Case Diagram:



5.2. Termination/Post Condition Tags:

<NOTIFY>	User is notified of successful completion of use case
<HOME>	User is returned to homepage
<TUSER>	User is granted access to the website
<TWAIT>	System is waiting

5.3. Alternative/Exceptional Flow Use Cases:

<INVALID>	<ol style="list-style-type: none"> 1. User data is deemed as being invalid 2. User is notified of invalid data 3. User is required to alter invalid data 4. New entered data is validated
<CANCEL>	<ol style="list-style-type: none"> 1. User selects to cancel the process 2. Process is cancelled 3. Changes are unsaved

5.4. Functional Requirements

Functional Requirement 1: User Registration

Purpose: All users are required to create an account to access Connect2.

Priority

High

Use Case:

Title	User Registration
Requirements Covered	1,2
Tags	<FRACCOUNT> & <FRPROFILE>
Scope	The scope of this use case is to allow users to sign up to and create an account with a user profile to appear on the web application. Upon creation, these accounts will be used to store the user's data and make it viewable to other users.
Actors	Employee and Employer users
Precondition	The user enters the application for the first time.
Activation	User creates an account
Main Flow	<ol style="list-style-type: none"> 1. User indicates they are an employee/employer user. 2. Data is validated by the system. 3. The system generates either an employee or employer user profile. 4. The values for the User Account information are stored in the User's account. The system alerts the user that the account has been created. 5. The user is brought to their user profile.
Exceptional Flow:	<INVALID>
Termination	Success: <TNOTIFY> Failure: <THOME>

Post Condition	Success: <TUSER> Failure: <TWAIT>
----------------	--------------------------------------

Functional Requirement 2: Sign In

Purpose: Users must sign into their account to access the web application

Priority

High

Use Case:

Title	Sign In
Requirements Covered	3
Tags	<FRSIGN>
Scope	The scope of this use case is to allow Users to sign into the system to use the web application features. It ensures Users are validated and can only access the correct accounts.
Actors	Employee and Employer users
Precondition	The User has created an account with the web application prior to signing in
Activation	User signs in
Main Flow	<ol style="list-style-type: none"> 1. The system prompts the User for their login details. 2. The User enters the required information. 3. The system validates the inputted data, ensuring the password matches the email address. 4. The User is signed in. The User is taken to the web applications homepage.
Exceptional Flow:	<ol style="list-style-type: none"> 1. <CANCEL> 2. <INVALID>
Termination	Success: <TNOTIFY> Failure: <TNOTIFY>
Post Condition	Success: <THOME> Failure: <TWAIT>

Functional Requirement 3: Add Information to User Profile

Purpose: Employee users can add skills to their profile

Priority

High

Use Case:

Title	Add Information to User Profile
Requirements Covered	3

Tags	<FREDIT>
Scope	The scope of this use case is to allow employee users to edit/delete or add to the information/skills displayed on their user profile used to apply to open positions.
Actors	Employee users
Precondition	The user is logged in and is an employee user
Activation	User edits profile
Main Flow	<ol style="list-style-type: none"> 1. The system displays the existing data on user skills to the user. 2. The employee user adds the personal information, knowledge areas and skills to their profile attached with the level of their skill 3. Employee provides brief high-level description of how they came to acquiring a particular skill 4. System validates data 5. Profile is updated
Exceptional Flow:	<ul style="list-style-type: none"> • <CANCEL> • <INVALID>
Termination	Success: <NOTIFY> Failure: <HOME>
Post Condition	Success: <UPDATE> Failure: <WAIT>

Functional Requirement 4: Complete Personality Testing

Use Case:

Title	Personality Testing
Requirements Covered	5
Tags	<FRPERSONALITY>
Scope	The scope of this use case is to allow employee users complete a series of testing designed to extract information from the user based on their personality. This use case will be a process of answering situational based questions to produce results.
Actors	Employee users
Precondition	The user is logged in and is an employee user.
Activation	User completes personality testing
Main Flow	<ol style="list-style-type: none"> 1. Questions are displayed 2. User answers questions 3. Answers are validated by the system 4. The system determines which personality types to assign to the profile based on user answers 5. Each user is assigned a result in <ul style="list-style-type: none"> • Extraversion (E) or Introversion (I) • Sensing (S) or Intuition (N) • Thinking (T) or Feeling (F) • Judging (J) or Perceiving (P) 6. The user is assigned their animal-work type 7. The user is alerted of their results

	8. The user chooses to add results to their profile
Exceptional Flow:	<CANCEL>
Termination	Success: <NOTIFY> Failure: <HOME>
Post Condition	Success: <UPDATE> Failure: <WAIT>

Functional Requirement 5: System enables job posting and prediction of job animal type

Purpose: Employer user able to post and manage open jobs to the web application.

Extended Description

- Employer user can create job opening with relevant information included

Priority

High

Use Case: Create Job

Title	Create Job
Requirements Covered	6
Tags	<FRJOB>
Scope	The scope of this use case is to allow employer users to create job items which are designed to describe the ideal candidate to the system and employee users. The data entered into the job posting will be used by the machine learning model to perform machining analyses on
Actors	Employer users
Precondition	The user is logged in and is an employer user
Activation	User adds a new job
Main Flow	<ol style="list-style-type: none"> 1. The user enters job information. 2. The entered job data is validated by the system. 3. The personality classification model predicts the work animal type 4. The user confirms data is correct. 5. The user is alerted of the job creation. 6. The job is posted
Exceptional Flow:	<ol style="list-style-type: none"> 1. <CANCEL> 2. <INVALID>
Termination	Success: <NOTIFY> Failure: <HOME>

Post Condition	Success: <UPDATE> Failure: <WAIT>
----------------	--------------------------------------

Use Case: Edit Job

Title	Edit Job
Requirements Covered	5
Tags	<FRJOB>
Scope	The scope of this use case is to allow users to edit/delete information from their job listing. This use case will contribute to ensuring job openings are flexible to alter.
Actors	Employer users
Precondition	The user is logged in and is an employer user
Activation	User edits an existing job
Main Flow	<ol style="list-style-type: none"> 1. The user selects to edit jobs. 2. The system displays the existing data on the job listing to the User. 3. The User enters/edits/delete the desired information from the job listing and requests the system updates and saves such data.
Exceptional Flow:	<ul style="list-style-type: none"> • <CANCEL> • <INVALID>
Termination	Success: <NOTIFY> Failure: <HOME>
Post Condition	Success: <UPDATE> Failure: <WAIT>

Use Case: Close Job

Title	Close
Requirements Covered	5
Tags	<FRJOB>
Scope	The scope of this use case is to allow Users to close a job when it is no longer available. This use case will contribute to ensuring that only available jobs are listed to employee users and job openings which have been filled are closed
Actors	Employer users
Precondition	The user is logged in and is an employer user
Activation	User closes an existing job
Main Flow	<ol style="list-style-type: none"> 1. The User chooses to close the job opening. 2. The job opening status is changed to closed. 3. The job opening is updated. 4. The user is notified of the closure.
Exceptional Flow:	<ul style="list-style-type: none"> • <CANCEL>

Termination	Success: <NOTIFY> Failure: <HOME>
Post Condition	Success: <UPDATE> Failure: <WAIT>

Functional Requirement 6: Employee users can view open jobs

Purpose: Employee user can view open jobs and jobs which have been automatically ranked/matched to their skills/personality

Priority

High

Use Case:

Title	View Open Jobs
Requirements Covered	7
Tags	<FRMATCH>
Scope	The scope of this use case is to assist employee users with seeking open job positions available on Connect2. It will allow employee users to view positions which have been matchmade with their skills and personality automatically.
Actors	Employee users
Precondition	The user is logged in and is an employee user
Activation	Employee views open positions
Main Flow	<ol style="list-style-type: none"> 1. Open positions are matchmade with employee user 2. Job positions appear in a ranked order 3. Employee user can select interested jobs and view more information
Termination	Success: <HOME> Failure: <HOME>
Post Condition	Success: <UPDATE> Failure: <WAIT>

Functional Requirement 7: Apply to job

Purpose: Enables employee users to apply for desired positions and additionally assist employer users in finding suitable candidates for the position.

Priority

High

Use Case:

Title	Apply to Job
Requirements Covered	8
Tags	<FRAPPLY>

Scope	The scope of this use case is to assist users with hiring processes. The use case enables employee users to apply for desired positions and additionally assist employer users in finding suitable candidates for the position.
Actors	Employee and Employer users
Precondition	The user is logged in
Activation	Employee applies to job
Main Flow	<ol style="list-style-type: none"> 4. Employee user submits application 5. Application is validated 6. Application is received 7. Application is ranked 8. Employer can view ranked application
Exceptional Flow:	<ul style="list-style-type: none"> • <CANCEL> • <INVALID>
Termination	Success: <NOTIFY> Failure: <HOME>
Post Condition	Success: <UPDATE> Failure: <WAIT>

Functional Requirement 8: Rank Candidates

Purpose: Ranks applicants for employer users to view and analyse how matched each is

Priority

High

Use Case:

Title	Rank Candidates
Requirements Covered	9
Tags	<FRMATCH>
Scope	The scope of this use case is to give employer users the ability to view the employee users who have applied to their job listing. In addition to such, it will provide the analyses on candidates and allow the user to gain insight into highly matched individuals.
Actors	Employer users
Precondition	The user is logged in as Employer user.
Activation	Employer views candidates
Main Flow	<ol style="list-style-type: none"> 1. List of candidates who have applied to the job listing in a ranked order and with further data analyses displayed outlining common attributes 2. Employer selects employee users they are interested in 3. Employer confirms list of employees to advance 4. Request to further application is sent to chosen employees 5. Unsuccessful employees receive notification
Termination	Success: <NOTIFY> Failure: <HOME>
Post Condition	Success: <UPDATE> Failure: <WAIT>

Functional Requirement 9: Advance Candidates

Purpose: Allows employer to advance the candidate they are interested in to the second round.

Priority

High

Use Case:

Title	Advance Candidates
Requirements Covered	10
Tags	<FRADVANCE>
Scope	The scope of this use case is to give employer users the ability to advance employee users they are interested in to the next round of the hiring process, and allow the employer user to gain insight into highly matched individuals.
Actors	Employer and employee users
Precondition	The user is logged in as Employer user.
Activation	Employer views candidates they have selected
Main Flow	<ol style="list-style-type: none">1. Employer chooses set of questions to be given to the interested profiles2. Questions are sent to employee user to answer3. Employee answer types answers to questions and sends back to the employer4. Employer receives the answers in the job listing from each employee
Exceptional Flow:	<CANCEL> <INVALID>
Termination	Success: <NOTIFY> Failure: <HOME>
Post Condition	Success: <UPDATE> Failure: <WAIT>

Functional Requirement 10: Final Candidates

Purpose: Allows employer to select the candidates they wish to interview

Priority

High

Use Case:

Title	Final Candidates
Requirements Covered	10
Tags	<FRFINALE>
Scope	The scope of this use case is to give employer users the ability to advance select which advanced employee users answers they liked the most from the second round and wish to interview for the position.
Actors	Employer and employee users
Precondition	The user is logged in as Employer user.

Activation	Employer views answers from second round
Main Flow	<ol style="list-style-type: none"> 1. Employer views the answers from advanced employees in round two 2. Employer selects which profile they wish to send to the final round 3. Employer confirms profile/profiles they wish to interview 4. Employer provides interview information (time/date/location) 5. Employee users who have not been advanced are removed from the hiring process and notified of such 6. Employee users who have been selected for the final round are notified of their interview
Exceptional Flow:	<CANCEL> <INVALID>
Termination	Success: <NOTIFY> Failure: <HOME>
Post Condition	Success: <UPDATE> Failure: <WAIT>

Functional Requirement 11: Grant Access to Personal Details/Secure Interview

Purpose: Allows users to secure interview time/date. Sends employee personal details to employer user

Priority

High

Use Case:

Title	Grant Access to Personal Details
Requirements Covered	10
Tags	<FRPERSONAL>
Scope	The scope of this use case is to give employee users the ability to share their personal details with their potential new employer and secure an interview time and date.
Actors	Employee users
Precondition	The user is logged in as Employee user and has been selected for interview by a potential employer.
Activation	Employee receives notification of being selected for interview
Main Flow	<ol style="list-style-type: none"> 1. Employee users who have been selected for the final round are notified of such 2. Employee user views proposed interview information 3. Employee chooses to accept the interview invitation 4. Personal details are shared to the potential employer
Alternative Flow	<ol style="list-style-type: none"> 1. Employee users who have been selected for the final round are notified of such 2. Employee user views proposed interview information 3. Employee chooses to decline the interview invitation 4. Employee provides reason for declining 5. Employer is notified of the decline
Exceptional Flow:	<CANCEL>

Termination	Success: <NOTIFY> Failure: <HOME>
Post Condition	Success: <UPDATE> Failure: <WAIT>

Functional Requirement 12: Admin Access

Purpose: The purpose of this use case is to ensure admin users can access the admin page of the web application.

Priority

High

Use Case:

Title	Final Candidates
Requirements Covered	10
Tags	<FRADMIN>
Scope	The scope of this use case is to allow admin users perform CRUD actions and have control over information/profiles on the web application.
Actors	Admin users.
Precondition	The user is logged in as an Admin user.
Activation	This use case starts when a User selects to access the admin page.
Main Flow	<ol style="list-style-type: none"> 1. The user selects to view admin page. 2. The system verifies that the user is an admin user. 3. Access is granted to the admin page. 4. User can access CRUD tools to manage data
Exceptional Flow:	<UNAUTHORISED> <INVALID>
Termination	Success: <NOTIFY> Failure: <HOME>
Post Condition	Success: <UPDATE> Failure: <WAIT>

Functional Requirement 13: Delete account

Purpose: : All users can delete their account at any point in time. Account deletion removes user data from the database.

Priority

High

Use Case:

Title	Delete Account
Requirements Covered	14
Tags	<FRDELETE>

Scope	The scope of this use case is to give users control over their account status and delete their user account if they please.
Actors	Admin Users
Precondition	The user is logged into their existing account.
Activation	This use case starts when a User selects to delete their account.
Main Flow	<ol style="list-style-type: none"> 1. The user selects to delete their account. 2. The system verifies the user by asking them to re-enter their password. 3. The password is verified. 4. The user account and connected data is dropped from the data base.
Exceptional Flow:	<CANCEL>
Termination	Success: <NOTIFY> <UPDATE> Failure: <HOME>
Post Condition	Success: <UPDATE> Failure: <WAIT>

6. Data Requirements

User Data

When users create an account with Connect2, they are required to supply the website with certain data. In the case of recruiters, the company information, location, email and phone number are required. In the case of candidates, personal details and area of work are also required.

All users will be required to create a username and password to log in and out of the Connect2 system.

Personality Data

Since Connect2 has its own machine learning personality classifier model, a considerable amount of data on the personality types is required. Originally in the project's proposal, a dataset from Kaggle was identified to be used. This dataset however, was largely unsuitable to train the projects model due to the sort of content. Connect2 required descriptive data on the personality types. To obtain this data, the project had to obtain volumes of descriptive information from multiple sources.

When the data collection stage was complete, the personality dataset resulted in a CSV file containing 20,916 rows of data. Each row represents a certain animal type, i.e. Owl, Beaver, Elephant or Fox. This data was used to be fed into the machine learning model for both training and testing purposes.

Skills and Experience Data

In order for the application and job ranking systems to function, user data is required from both sets of users regarding skills and experience. Both recruiters and candidate users are required to enter a minimum of two for each category for each job listing or employee user profile. This requirement ensures the ranking systems are capable of scoring and making recommendations.

7. Environmental and Usability Requirements

The environment requirements for Connect2 are as follows:

Operating System:

The operating systems compatible with Connect2 are:

- Windows 8
- Windows 10
- MacOS 10.11
- MacOS 10.12
- MacOS 10.13

Browser:

Recommended latest versions of

1. Chrome
2. Firefox
3. Edge
4. Internet Explorer 11
5. Safari

Programming Language and Libraries:

- HTML
- CSS
- JavaScript
- JQuery
- Django
- Python
- Pandas
- Scikit-Learn
- Natural Language Processing Toolkit (NLTK)
- TensorFlow
- PyTest

Usability Requirement 1: Learnability

Connect2 must have an interface which provides the users access to the various functions with ease. The purpose of each page should be easily understood.

Usability Requirement 2: Error Management

The system must be designed to alert the user of any error that occurs.

Users will be able to report issues to [admin](#) users with a report function provided on the web application.

Usability Requirement 3: Performance

The activities users complete on Connect2 must be easy to complete and require little to no external assistance. Use Cases should be of ease to access and complete with exceptional flows navigating users to the correct parts of the web app.

Usability Requirement 4: Information Scent

Users must find Connect2 easy to find useful information regarding the application. Visual Cues must be utilised to provide information on different features. Dedicated web pages with information regarding the personality testing and results must be informative and structured.

8. Design & Architecture

Connect2 as outline previously will consist of a web application built with the Django Framework. Connect2 has four separate objectives:

1. An employment website for recruiters to post jobs and candidates can apply directly.
2. Built in applicant ranking software.
3. Application management system recruiters can use.
4. Built in direct messaging and interview scheduling system.

A personality classifier model was trained with the customer personality dataset to become proficient at detecting personality type from recruiters inputted text and job descriptions. The model is able to assess text to categorise the job into a specific group.

The matching making function displays jobs openings to employees which it calculates to be a good match on the basis of the job requirements and the employee user's personality, experience and skills. When employee users view open jobs, they will automatically see highly matched jobs at the top of the page, descending to jobs with lower matched rankings. Additionally, employee users can apply filters to the jobs they view such as:

- Line of work
- Location
- Post Date
- Company
- Office/Remote Work

Employers will view applicants to the open positions ranked against one another. With highly matched most suitable candidates appearing at the top and descending to less matched candidates. Employers can select which candidates they wish to advance to the next and final round. Employers can select candidates to issue interview requests to. Candidates can decide to accept or reject the interview request and share their user profile.

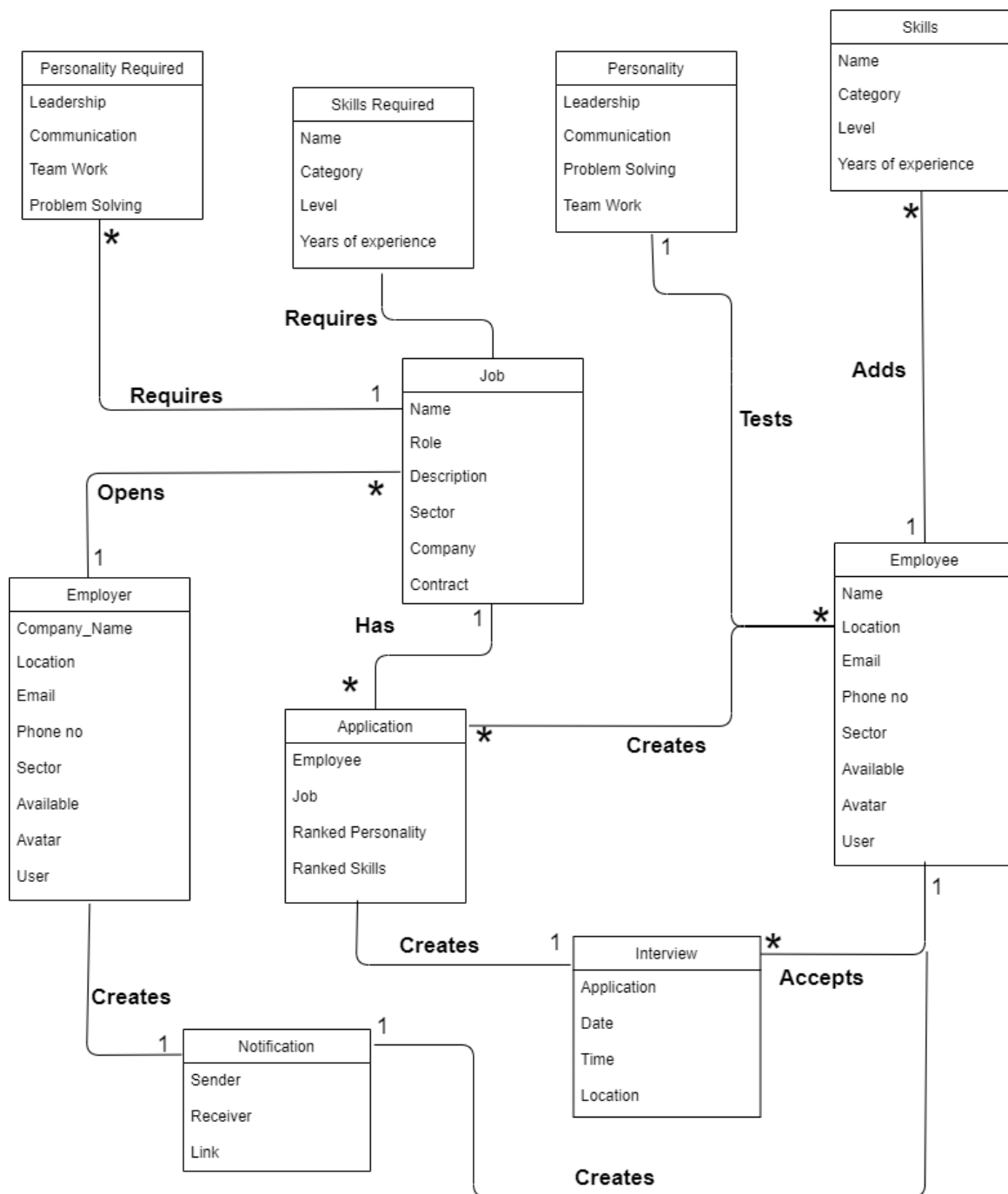


Figure: Class Diagram explaining the architecture of Connect2.

9. Implementation

Connect2 technical implementation was completed as follows:

HTML Templates

Several HTML pages were used for Connect2’s implementation. These templates were designed using CSS and the Bootstrap framework. The colour scheme for Connect2 followed a purple and white theme.

ACCOUNT		
Applications	+ Add	🔧 Change
Conversation messages	+ Add	🔧 Change
Employee skills	+ Add	🔧 Change
Employees	+ Add	🔧 Change
Employers	+ Add	🔧 Change
Experiences	+ Add	🔧 Change
Interview plans	+ Add	🔧 Change
Job rankings	+ Add	🔧 Change
Job skills	+ Add	🔧 Change
Job tags	+ Add	🔧 Change
Jobs	+ Add	🔧 Change
Location tags	+ Add	🔧 Change
Ltms	+ Add	🔧 Change
Matched skills	+ Add	🔧 Change
Shared things	+ Add	🔧 Change
Users	+ Add	🔧 Change

Recent actions

My actions

- 🔧 Smoothie Job
Job
- 🔧 Computer Job
Job
- 🔧 Teacher Job
Job
- 🔧 Rob-2021-05-09 15:19:12.678692+00:00
Employee
- 🔧 Josh-2021-05-14 16:04:06.544420+00:00
Employee
- 🔧 x17323943
Job
- + JobTag object (9)
Job tag
- + JobTag object (8)
Job tag
- + JobTag object (7)
Job tag
- 🔧 Computer Job
Job

Figure: Image of Connect2's admin dashboard.

Select employee to change

Action: Go 0 of 8 selected

- EMPLOYEE
- Josh-2021-05-14 16:04:06.544420+00:00
- testuser123-2021-05-13 10:05:25.632758+00:00
- pat-2021-05-13 09:56:16.040394+00:00
- Rob-2021-05-09 15:19:12.678692+00:00
- archie-2021-05-09 14:32:10.221293+00:00
- James-2021-05-03 11:25:01.417838+00:00
- amy-2021-04-12 16:17:10.926691+00:00
- hone-2021-04-12 16:16:53.518812+00:00

Figure: Image of Connect2's admin dashboard.

Machine Learning Implementation

The use of machine learning in this project is to use data supplied to the system from the employer used to systemically determine the appropriate personality type for specific jobs. A text classification model was created to predict which work animal is best matched with a job listed.

The prediction task involved the model predicting one of 4 types:

1. NT (OWL)
2. SP (FOX)
3. SJ (BEAVER)
4. NF (ELPEHANT)

To create the model, the bag-of-words approach was taken. The bag-of-words model is method oof representing text data when modelling text with machine learning algorithms. The text from the personality data set was converted into fixed-length vectors through counting the occurrence of each work. The process is also known as vectorisation (Zhou, 2019).

The first stage of creating the ML model involved uploading the personality dataset csv file. Pandas was used to convert the file into a data frame. The data frame was then shuffled.

```
data8 = pd.read_csv('test.csv')
data8 = shuffle(data8, random_state=22)

data8.head()
```

Figure: Creating a data frame from the CSV file.

The labels (NF,NT,SP and SJ) were encoded as four hot element arrays using the Scikit-learn label binarize function as part of the bag of words approach.

```
tags_split = [tags.split(',') for tags in data8['tags'].values]
print(tags_split, '\n')

tag_encoder = LabelBinarizer()
tag_encoded = tag_encoder.fit_transform(tags_split)
num_tag = len(tag_encoded[0])
print(data8['content'].values[0])
print(tag_encoder.classes_)
print(tag_encoded[0])
```

Figure: Encoding personality tags

The data frame was split using the 80/20 train test method. More information on such is provided in the testing section.

Next a keras tokenizer utility was used to create the vocab used in the bag of words.

```

%%writefile preprocess.py

from tensorflow.keras.preprocessing import text

class TextPreprocessor(object):
    def __init__(self, vocab_size):
        self._vocab_size = vocab_size
        self._tokenizer = None

    def create_tokenizer(self, text_list):
        tokenizer = text.Tokenizer(num_words=self._vocab_size)
        tokenizer.fit_on_texts(text_list)
        self._tokenizer = tokenizer

    def transform_text(self, text_list):
        text_matrix = self._tokenizer.texts_to_matrix(text_list)
        return text_matrix

```

Figure: Creating the vocab.

The bag of word matrices were created next. The bag of words size for this project was chosen to be 1000 after various different amounts were tested. Both the train and test data frames were converted into bag of word matrices.

```

from preprocess import TextPreprocessor

VOCAB_SIZE= 1000

train_qs = data8['content'].values[:train_size]
test_qs = data8['content'].values[train_size:]

processor = TextPreprocessor(VOCAB_SIZE)
processor.create_tokenizer(train_qs)

body_train = processor.transform_text(train_qs)
body_test = processor.transform_text(test_qs)

```

Figure: Creating the bag of word matrices.

The tokenizer was then saved for later use using pickle.

```

import pickle

with open('./processor_state.pkl', 'wb') as f:
    pickle.dump(processor, f)

```

Figure: Saving the tokeniser object to a file using pickle.

The model was created using Keras. The model accepted the previously set vocab size and the number of tags. The sequential model API of Keras was used to define the model as a stack of layers. The first layer of the model is a dense fully connected layer. The input shape in this case is the vocab size. Relu was used as the activation function. A second layer was added similar to the first layer. The final layer was set with 4 neurons (the 4 animal types) and the activation function used was sigmoid.

```
def create_model(vocab_size, num_tags):

    model = tf.keras.models.Sequential()
    model.add(tf.keras.layers.Dense(50, input_shape=(VOCAB_SIZE,), activation='relu'))
    model.add(tf.keras.layers.Dense(25, activation='relu'))
    model.add(tf.keras.layers.Dense(num_tags, activation='sigmoid'))

    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model

model = create_model(VOCAB_SIZE, num_tag)
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 50)	50050
dense_1 (Dense)	(None, 25)	1275
dense_2 (Dense)	(None, 4)	104
Total params: 51,429		
Trainable params: 51,429		
Non-trainable params: 0		

Figure: Creating the personality prediction model.

Training the model was completed using the model.fit() function. 10% of the data was taken to evaluate the model as it was training. The training model eventually was producing an accuracy of 90%.

```
model.fit(body_train, train_tags, epochs=20, batch_size=800, validation_split=0.1)
```

Figure: Training the model

accuracy: 0.9004 - val_loss: 0.0937 - val_accuracy: 0.9056

Figure: Model results

The model was further tested, which is outlined in the testing section. The model was then imported to the Django project and ran as part of the post job view found in the employer app.

A customer model prediction class was created which pre processes data. The imported model and tokenizer are instantiated in the class. Text data is taken in and transformed to return a sigmoid probability array.


```
[[7.166762952692807e-05, 0.9901820421218872, 0.02297857403755188, 3.19152204610873e-05]]
Predicted labels:
nt
nt
```

Figure: Example of model predicting NT (Owl) personality type for a job listing.

THE JOB IS NOW POSTED.

Thank you for choosing Connect2.

IDEAL PERSONALITY TYPE _____

OWL

The ideal personality type for this job is Owl. For more information on the different animal types click [here](#).

In the meantime, below is information regarding the predicted ideal personality type for the job you have listed. Candidates with this personality type will be granted bonus points in our application ranking system.

Character traits are not something you can teach, so when you need candidates with certain soft skills, your best bet is to hire based on personality and then teach the hard skills.



Goal Oriented

Owls see life as a system to be designed and redesigned. They learn by testing the system in a relentless pursuit of excellence. They have a talent for conceptualising and systems planning and are strategic planners and goal setters. When an Owl has a new idea, they set goals to achieve it.

Logical

Owls gather data consistently largely of abstractions and possibilities which they filter through their objectives decision making process. Their driving force in their never ending quest for competence is to theorise and intellectualise everything meaning they are some of the most logical beings.

Figure: Information page displaying after prediction of animal type for job listing. (Owl).

Ranking Systems

There are three ranking systems used in Connect2.

The first ranking system was used to score applications made to a particular job listing. The ranking system considered the personality, skills and experience required for a job listing and compared it to the employee of the application. Each application was assigned a score and their application was ran through the following code:


```

personality = Personality.objects.get(employee=employee)
person_type = personality.group
job_type = job.ideal_person

if request.method == 'POST':
    form2 = ApplyJobForm(request.POST)

    if form2.is_valid():
        application = form2.save(commit=False)
        application.job = job
        application.candidate = candidate
        application.status = 'applied'
        application.person_type = person_type
        application.job_type = job_type
        application.save()
        app = application
        job.no_applicants = job.no_applicants+1
        print(job.no_applicants)
        job.save()
        create_notification(request, job.employer.user, application.job, app,
                           'application', extra_id=application.id)

        application.save()
        score = 0
        print(job_type)
        print(person_type)
        if job_type == person_type:
            application.match = True
            score += 5

        else:
            application.match = False

        for x in intersection_as_list:
            MatchedSkills.objects.create(application=application, title=x)

            score += 1
            application.score = score


            print(application.score)
            application.save()

```

If the personality was a match, they earned a 5-point bonus. For every skill/experience in common, a matched skill object was created, and they earned an additional 1 point. The application was then saved with the score.

When an employer accesses their job dashboard, the applications appear in scored order, with common attributes listed. In the following example, the job was looking for skills in data science and Java. Furthermore, it was categorised as most suited to having an employee with an owl animal type.

Junior Developer
Employer: Skype
Number of Applicants: 6
Status: Open



We have matched owls with this listing!

Job Dashboard

#	Applied	Personality	Score	Common Skills	Cover Letter
1	May 16, 2021, 8:23 p.m.	✔	7	Data Science Java	Cover Letter
2	May 16, 2021, 8:19 p.m.	✘	1	Java	Cover Letter
3	May 16, 2021, 8:22 p.m.	✘	1	Java	Cover Letter

New Round

Interview

Figure: Job dashboard. Purple tick representing that there is a match in personality. The score is boosted due to this match.

The second ranking system implemented in Connect2 accounted for the job recommendations made to employee users. When an employee user wishes to find a job, the application will recommend open job listings according to their specific skills/experiences.

```
for job in jobs:
    if job.status == 'Open':
        jobrank = JobRankings.objects.filter(job=job)
        if jobrank:
            jobrank.delete()
        jobrank = JobRankings.objects.create(
            job=job, score=0, employee=employee)

        jobskills = job.jobskill_set.all().values_list('title', flat=True).distinct()
        employeeskills = employee.employeeskill_set.all().values_list('title',
                                                                    flat=True).distinct()

        jobtags = job.jobskill_set.all().values_list('title', flat=True).distinct()
        employeetags = employee.employeeskill_set.all().values_list('title',
                                                                    flat=True).distinct()

        employeeskills_as_set = set(employeeskills)
        employeetags_as_set = set(employeetags)
        intersection = employeeskills_as_set.intersection(jobskills)
        intersection2 = employeetags_as_set.intersection(jobtags)
        intersection_as_list = list(intersection)
        intersection_as_list2 = list(intersection2)

        matched_amount = len(intersection_as_list+intersection_as_list2)/2
        score = matched_amount
        if job.ideal_person == personality.group:
            score += 1
        else:
            hscore = 0
            for x in intersection_as_list:
                score += 1
                SharedThing.objects.create(
                    title=x, jobrank=jobrank, sort="Skill", employee=employee)
            jobrank.score = score
            jobrank.save()

checker = 0
jl = 0

jobrankings = JobRankings.objects.filter(
    job=job).order_by('-score')
print(jobrankings)
p = 0
checker = 0
for l in jobrankings:
    checker = checker+1
    if checker < 3:
        l.topthree = True
        l.save()
    else:
        p = 0
```

Figure: Code to rank jobs for employee users

The above code accounts for the top three jobs recommended to the specific user. Firstly the system checks if there is a personality match and awards jobs with matched personality scores a point. Secondly the list of skill requirements for the job is compared with the employee skills. Each matching skill warrants an additional point. The top 3 job rankings are then displayed for the user. These job rankings update as more skills/new jobs are added to Connect2.

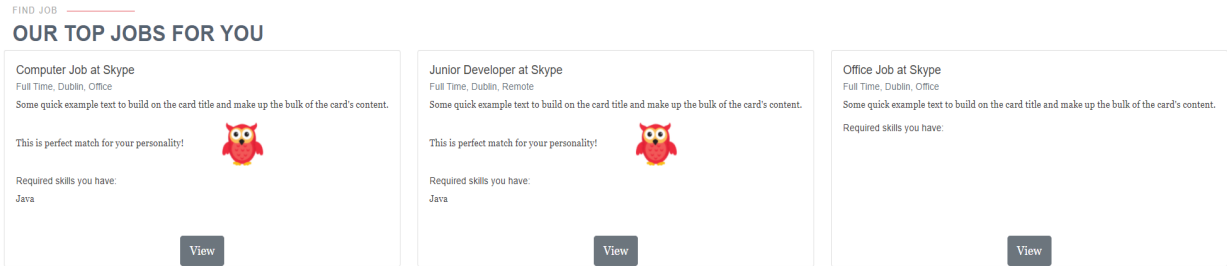


Figure: Job recommendations to the employee user based on their individual talents and personality. The work animal type symbol appears if it's a personality match.

The next recommendation system was more complex. Firstly, Connect2 would take the city of the logged in employee user and perform a check to see what the most in demand tags are in that city. Next it would find which tags from the employee profile are most in demand according to the city skills. The most in demand employee tag in the employee users city was saved. Jobs in the city with this tag were then recommended to the user.

The reason for this recommendation is since there is a high demand for a specific tag in that city, there is a likelihood employers are short in supply of individuals working in this area. Employee users have higher chances of securing an interview if they have highly in demand skills.

```

city = employee.city
score = 0
jobtags = JobTag.objects.all()

persontags = PersonTag.objects.filter(employee=employee)
matchedtags = []
for j in jobtags:
    if j.job.city == city:
        for n in persontags:
            if j.title == n.title:
                matchedtags.append(n.title)
                print(matchedtags)

                p = LocationTag.objects.create(
                    job=j.job, employee=employee, tag=j.title)

            hi = Counter(matchedtags)

most_common_element = hi.most_common(1)

key = most_common_element[0]

hi = key[0]

l1mn = LTM.objects.create(
    employee=employee, city=city, tag=hi)
l1mn = LTM.objects.get(employee=employee)
areatags = LocationTag.objects.filter(tag=hi).order_by("?")[:3]

```

Figure: Code which determines most in demand employee tag within their city.

Each time the user visits the find job page, 3 random jobs appear in their city which require this tag.

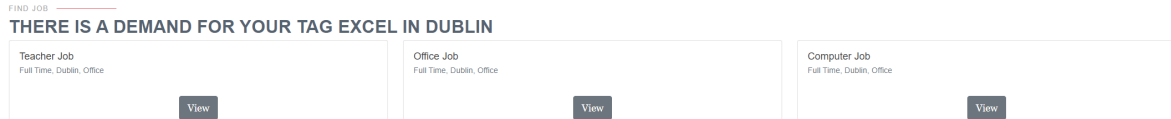


Figure: Jobs being recommended to user on the basis of their most in demand tag

Other Functionality

Connect2 allows user to filter for other jobs if they are uninterested in the recommended jobs. They can search by title, location, contract or office type. Similarly, employer users can filter their posted jobs by status and application amounts. These filters were implemented using Filter classes.

```
class JobFilter(django_filters.FilterSet):
    Title = CharFilter(field_name='title', lookup_expr='icontains')

    class Meta:
        model = Job
        fields = '__all__'
        fields = ('Title', 'status', 'no_applicants')

class JobFilter2(django_filters.FilterSet):
    Title = CharFilter(field_name='title', lookup_expr='icontains')

    class Meta:
        model = Job
        fields = '__all__'
        fields = ('Title', 'location', 'contract', 'office_type')
```

Figure: Example Filter classes

Employee users were categorised into their work type animal using python code to score their answers. The scores in each of the 8 MBTI categories was checked to give them their MBTI type and subsequently their animal work type.

```
for q in questions:
    a_selected = request.POST.get(q.text)
    if a_selected != "":
        question_answers = Answer.objects.filter(question=q)
        for a in question_answers:
            if a_selected == a.text:
                if q.category == "e":
                    if a_selected == "Always":
                        e_score += 10
                        print(e_score)
                    elif a_selected == "Often":
                        e_score += 7
                        print(e_score)
                    elif a_selected == "Sometimes":
                        e_score += 5
                        print(e_score)
                    elif a_selected == "Rarely":
                        e_score += 3
                        print(e_score)
                    elif a_selected == "Never":
                        e_score += 0
                        print(e_score)
```

Figure: Sample code from the quiz scorer

```

if e_score > i_score:
    if n_score > s_score:
        if f_score > t_score:
            if j_score > p_score:
                Result.objects.create(quiz=quiz, employee=employee, e_score=e_score, i_score=i_score, s_score=s_score,
                                     n_score=n_score, f_score=f_score, t_score=t_score, j_score=j_score, p_score=p_score)
                Personality.objects.create(
                    first_letter='e', second_letter='n', third_letter='f', fourth_letter='j', person_type='enfj', employee=employee, group='nf', is_complete=True)

```

Figure: Sample code from the personality scoring in the quiz view. This user is being given the elephant type.

The notification system was implemented using a function which would create a notification and issue it to the receiver containing a link either to the new job or application made on Connect2.

```

✓ def notifications(request):
    goto = request.GET.get('goto', '')
    notification_id = request.GET.get('notification', 0)
    extra_id = request.GET.get('extra_id', 0)

    ✓ if goto != '':
        notification = Notification.objects.get(pk=notification_id)
        job = notification.job
        notification.is_read = True
        notification.save()

    ✓ if notification.notification_type == Notification.MESSAGE:
        return redirect('view_application', id=notification.extra_id, pk=job.pk)
    ✓ elif notification.notification_type == Notification.APPLICATION:
        return redirect('jobprofile', pk=job.pk)

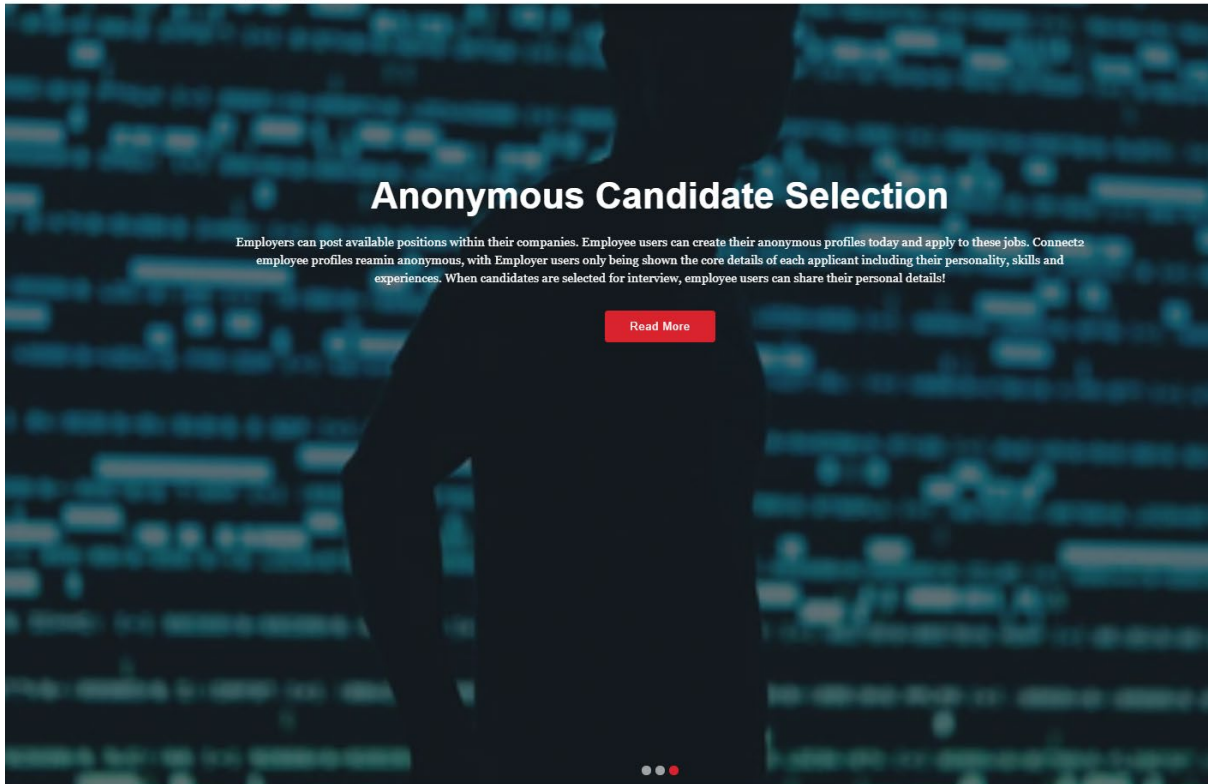
    return render(request, 'employer/noti.html')

```

Figure: Code issuing notification system

10. Graphical User Interface

Images showing the GUI for Connect2:



Anonymous Candidate Selection

Employers can post available positions within their companies. Employee users can create their anonymous profiles today and apply to these jobs. Connect2 employee profiles remain anonymous, with Employer users only being shown the core details of each applicant including their personality, skills and experiences. When candidates are selected for interview, employee users can share their personal details!

[Read More](#)

A NEW APPROACH TO HIRING!

Try Connect2 for your job searching or hiring needs! We have helped hundreds of individuals and organisations find their match.

Complete our professional personality test to understand your behavioural tendencies within the professional environment!

Create job listings and allow Connect2 determine exactly what type of employee you are searching for!

Contact employees and employers directly using Connect2. Schedule interviews and more!

Connect2 will rank job applicants in against one another, allowing the employer to see which individuals meet the skills and personality requirements for the position the most.

Figure: Screenshot of homepage



Robert Smith



Robert Smith

Dublin,dublin

rob@rob.com

98989

Edit your personal details



You are part of the Owl group

Your work animal represents who you are as a person and your behavioural tendencies in work. For more information please click [here](#)

Data Science

Management

Excel

Java

Microsoft Office

Change Management


Linux

Ux Design

Add Skill

Add Tag

Figure: Screenshot of employer user profile



Home Profile Applications Find Job Notifications (0) Logout About Us

FIND JOB

OUR TOP JOBS FOR YOU

Office Job at Skype
Full Time, Dublin, Office

Some quick example text to build on the card title and make up the bulk of the card's content.


Required skills you have:
Microsoft Office
Excel

[View](#)

Junior Developer at Skype
Full Time, Dublin, Remote

Some quick example text to build on the card title and make up the bulk of the card's content.

This is perfect match for your personality!




Required skills you have:
Data Science

[View](#)

Computer Job at Skype
Full Time, Dublin, Office

Some quick example text to build on the card title and make up the bulk of the card's content.

This is perfect match for your personality!



Required skills you have:

[View](#)

FIND JOB

THERE IS A DEMAND FOR YOUR TAG CHANGE MANAGEMENT IN DUBLIN

Computer Job
Full Time, Dublin, Office

[View](#)

Computer Job
Full Time, Dublin, Office

[View](#)


NOT TOO SURE ABOUT THOSE?

CHECK THESE OUT

Title contains: Location: Contract: Office type: [Search](#) [Clear](#)

Title	Company	Date Posted	Location	
Office Job	Skype	May 13, 2021, 2:22 p.m.	Dublin	View
Computer Job	Skype	May 13, 2021, 2:24 p.m.	Dublin	View


Figure: Screenshot of “Find Job” page for Employee user



Home Profile Your Jobs Browse Employees Notifications (4) Logout About Us


Job Dashboard


Office Job
Employer: Skype
Number of Applicants: 6
Status: Open



We have matched foxs with this listing!

#	Applied	Personality	Score	Common Skills	Cover Letter	Highlight	Round	
1	May 16, 2021, 9:45 p.m.	✓	6	Customer Service	Cover Letter	☆		Message Delete
2	May 16, 2021, 9:46 p.m.	✗	2	Excel Microsoft Office	Cover Letter	☆		Message Delete
3	May 14, 2021, 5:24 p.m.	✗	None		Cover Letter	★		Message Delete

 this symbol means that the applicant has a personality match with the job
 [Cover Letter](#)
Click this button to view the applicants personalised cover letter

 This symbol means that the applicant does not have a personality match with the job
 SCORE
The score represents how this applicant scored in the Connect2 ATS systems. The score is based on shared skills, experiences and personality.


 Click this button to favourite the applicants you are more interested in
 [Schedule](#)
Schedule interviews with your selected candidates once the job has reached its interview stage

Figure: Screenshot of Application Management System

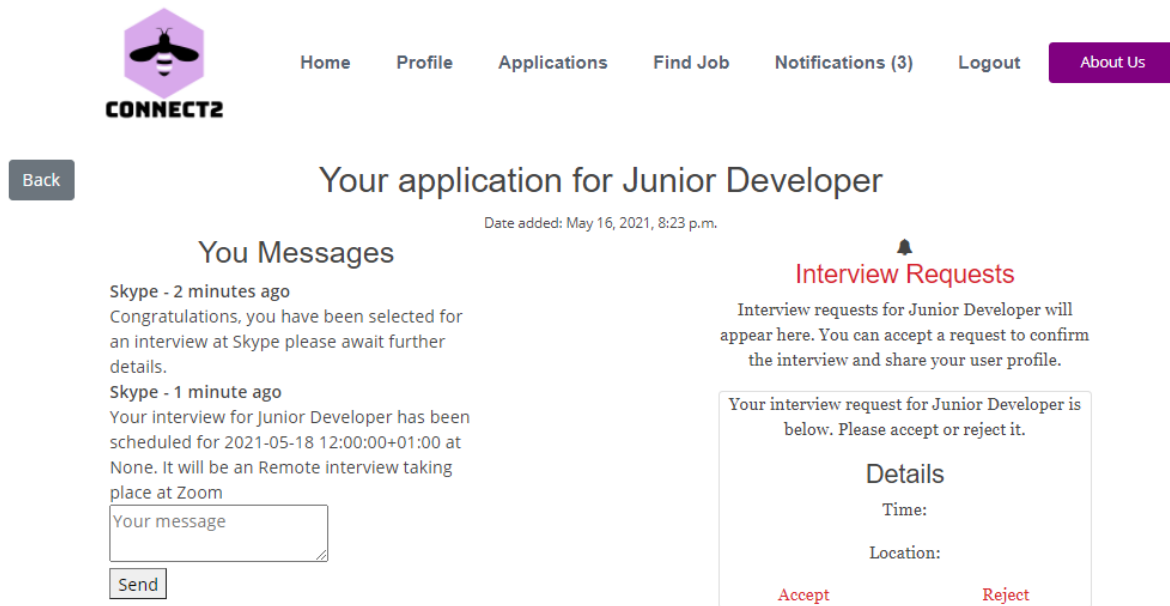


Figure: Screenshot of messaging system with interview request logged.

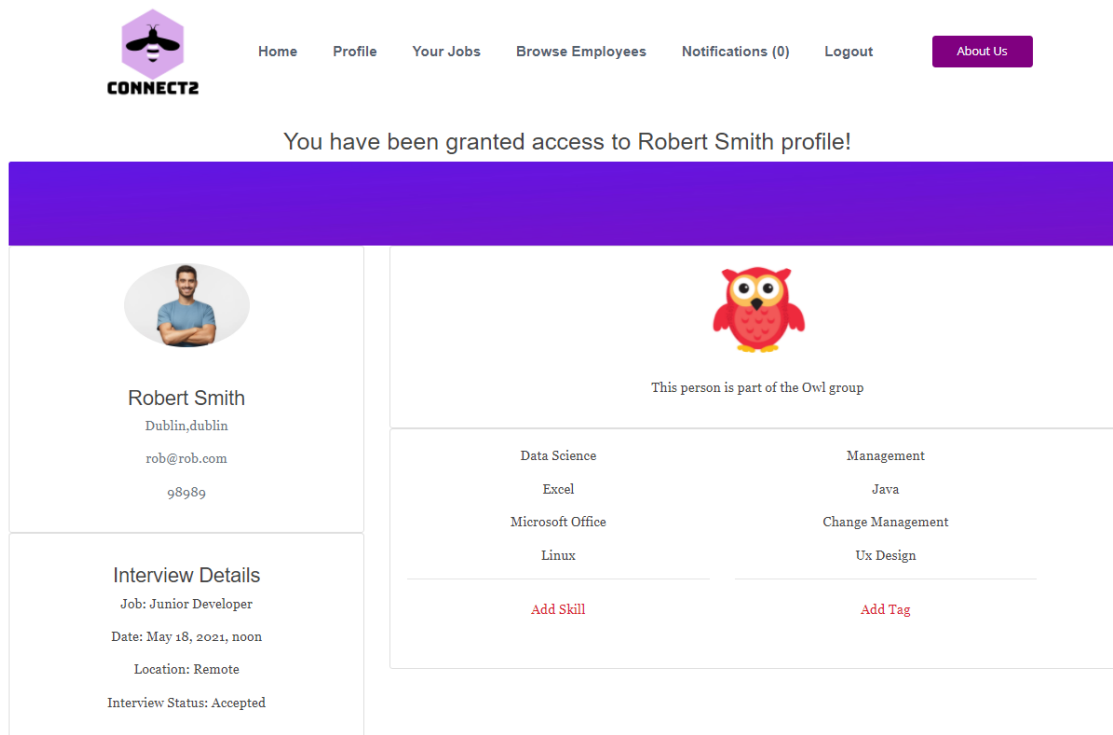



Figure: Screenshot of user profile access being granted to recruiter after an interview has been scheduled.

 [Home](#) [Profile](#) [Applications](#) [Find Job](#) [Notifications \(0\)](#) [Logout](#) [About Us](#)

Sometimes

Rarely

Never

You feel relieved when you get home after a day of social interactions.

Always

Often

Sometimes

Rarely

Never

You would describe yourself as being impulsive

Always

Often

Sometimes

Rarely

Never

You would consider yourself to be more imaginative than observant

Always

Often

Sometimes

Rarely

Never

You don't mind being at the centre of attention

Always

Often

Sometimes

Rarely

Never

Figure: Personality Quiz for Employee users to take

11. Testing

An AAA(Arrange-Act-Assert) approach was be followed for testing, this creates a pattern for the tests that makes them readable and useful.

11.1 Black-Box Testing

Black box testing was used to test specific functions or features of Connect2. Black-box testing was continuously conducted throughout the projects development to verify the introduction of new code and features. It was achieved through supplying new functions various inputs and observing the outputs.

11.1.2 Functional Testing

Smoke testing was the first type of testing conducted for Connect2. Smoke Testing is a software testing technique performed after development to verify functionality of core components. It was executed prior to detailed testing of the application and typically conducted after the introduction of new code to the system.

A log of the most important elements of Connect2 which make up the RS were tested using black box testing:

Smoke Testing for Connect2						
#	Title	Description of test	App	User	P/F	Comments
1	Create Account	Choose if you are employer or employee, enter details and register account. User profile is created accordingly.	Account	Employee/Employer	P	Working
2	Login / Logout	Allow a user to login with correct credentials Incorrect credentials warrant an error	Account	Employee/Employer	P	Working
3	Home Page	Access homepage after login Displays navbar and information	Employee /Employer	Employee/Employer	P	Working
4	User Profiles	View user profile Update and Edit personal details	Employee /Employer	Employee/Employer	P	Working
5	Complete personality test	Complete employee personality test. Results are saved and information page with the correct work animal is displayed	Employee	Employee	P	Personality generation working and animal information pages displaying correctly
6	Add Skill/Tag	Add a skill or tag to an employee/job listing	Employee /Employer	Employee/Employer	P	Working
7	View Jobs	View posted Jobs and see recommendations	Employee	Employee	P	Working
8	View Job Profile	View job profile with job information and required skills/tags displaying	Employee	Employee	P	Working
9	Filter Jobs	Enter interests and click filter to view relevant jobs	Employee	Employee	P	Working
10	Receive notifications	Send a message as one user and receive a notification as the receiver of the message	Employer /Employer	Employee/Employer	P	Working
11	Send Messages	Send messages to employee or employer related to a job application	Employer /Employer	Employee/Employer	P	Working
12	Create Job	Create job posting with job details specified including ideal candidate requirements	Employer	Employer	P	Job postings saving. Classification model correctly predicting animals for job and saving.
13	Receive notifications for job applications	Send an application to a job as an employee and receive a notification as an employer than the job listing has a new applicant.	Employer	Employer	P	Working
14	View applications	View all applications made to a particular job. Applications are ranked in descending order with personality match and common skills displaying correctly. Cover letter can be viewed.	Employer	Employer	P	Cover Letter not displaying correctly
15	Highlight application	Highlight application from job dashboard to favourite it	Employer	Employer	P	Working
16	Create interview round	Create interview round and all highlighted applications are selected to receive an interview	Employer	Employer	P	Working
17	Delete application	Delete applications from job listing	Employer	Employer	P	Working
18	Schedule interview	Schedule an interview for each highlighted application. Select date, time, sort of interview and location.	Employer	Employer	P	Working
19	Interview information is sent to employee	Automatic message is issued to employee user when an interview is scheduled containing interview information	Employee	Employee	P	Working
20	Interview request is issued to employee	Automatic interview request is sent to employees selected for interview. Accept or reject interview is possible.	Employee	Employee	P	Working
21	Interview is accepted and profile is shared.	Accept interview as employee and employer receives message containing a link to the employees personal user profile.	Employer	Employer	P	Working
22	Access limited version of employee user profile	View employee profile as an employer with interview details specified on the page.	Employee	Employer	P	Working
23	Reject interview	Reject interview request and automatic message to employer is issued notifying them of the rejection	Employer	Employer	P	Working

Figure: Table outlining elements tested for smoke testing to verify stability of Connect2.

Conclusions:

The overall functionality of Connect2 passed the smoke tests. Each requirement from the RS was achievable.

11.2 Performance Testing

11.2.1 Personality Classification Model Testing

Test Description:

The machine learning personality predicting modal was tested using a pre-defined testing personality data set and noting its performance. The classification model was trained using 80% of the personality dataset whilst the remaining 20% was saved for later testing purposes.

The model's performance was tested using cross-validation.

Why is this being tested?

The personality classification model is responsible for the classification of job listings into animal types. This is a core component of Connect2 and it is vital that the models performance is verified.

Arrange:

The personality dataset was divided into two sets: training and test.

```
train_size = int(len(personalitydata)*.8)
test_size = len(personalitydata)- train_size
VOCAB_SIZE= 1000

train_qs = data8['content'].values[:train_size]
test_qs = data8['content'].values[train_size:]

processor = TextPreprocessor(VOCAB_SIZE)
processor.create_tokenizer(train_qs)

body_train = processor.transform_text(train_qs)
body_test = processor.transform_text(test_qs)
```

Figure: Code setting the size of the testing dataset and splitting it. The tokenizer is used to divide the chunks of text into words. Body_test is 20% of the personality testing data set split into words.

Assemble:

Once the model had been fitted using the training dataset, the model was evaluated using the testing set.

```
model.evaluate(body_test, test_tags, batch_size=128)
```

Figure: Assembling the model for testing

Act:

The model was evaluated. From the training data, an accuracy of 90% was achieved. To pass the test, the model had to score similarly.

```
0s 11ms/step - loss: 0.0956 - accuracy: 0.9004 - val_loss: 0.0937 - val_accuracy: 0.9056
```

Figure: Results from training modal

```
model.evaluate(body_test, test_tags, batch_size=128)
```

```
33/33 [=====] - 0s 2ms/step - loss: 0.0938 - accuracy: 0.9011
[0.09383134543895721, 0.9010516405105591]
```

Figure: Results from training modal. 90% accuracy was achieved.

Conclusions:

The test passed.

11.3 Unit Testing

Connect2's key individual code components were selected for unit testing. A series of unit tests were carried out to measure the functionality of these individual components at a low level. These classes were tested using the PyTest framework of Python and underwent a series of dynamic white box tests.

Dynamic white-box testing was utilised to test the internal code, structure, and design of Connect2. It involved running the test code and seeing the results. The core advantages of white box testing are the maximum test coverage plus the opportunity to find hidden errors which impact the quality of the code.

11.3.1 Unit Test: Personality Categorising Code

(available in `personality/views.py`)

Why is this being tested?

This code component is responsible for categorising employee users into 1 of 16 groups of personality. Additionally, it assigns the user their Connect2 work animal. This code is essential to be tested because it gives employee users their personality, which is a key feature of Connect2 and the employee user profile.

Arrange:

The section of the `save_quiz_view` function which determines which category of personality the user will be placed in was simplified. Rather than creating personality objects, the system returns a set with the result personality and animal type.

```

def PersonalityGrouper(e_score, i_score, n_score, s_score, t_score, f_score, p_score, j_score):
    if e_score > i_score:
        if n_score > s_score:
            if f_score > t_score:
                if j_score > p_score:
                    return set('enfj - elephant')
                elif p_score > j_score:
                    return set('enfp - elephant')

            elif t_score > f_score:
                if j_score > p_score:
                    return set('entj - owl')

                elif p_score > j_score:
                    return set('entp - owl')

        elif s_score > n_score:
            if f_score > t_score:
                if j_score > p_score:
                    return set('esfj - beaver')

            elif p_score > j_score:
                return set('esfp - fox')

            elif t_score > f_score:
                if j_score > p_score:
                    return set('estj - beaver')

                elif p_score > j_score:
                    return set('estp - fox')

    elif i_score > e_score:
        if n_score > s_score:
            if f_score > t_score:
                if j_score > p_score:
                    return set('infj - elephant')

                elif p_score > j_score:
                    return set('infp - elephant')

            elif t_score > f_score:
                if j_score > p_score:
                    return set('intj - owl')

                elif p_score > j_score:
                    return "intp - owl"

        elif s_score > n_score:
            if f_score > t_score:
                if j_score > p_score:
                    return set('isfj - beaver')

            elif p_score > j_score:
                return set('isfp - fox')

            elif t_score > f_score:
                if j_score > p_score:
                    return set('istj - beaver')

                elif p_score > j_score:
                    return set('istp - fox')

```

Figure: Arranging the code to test the personality categoriser

Act:

The code was tested firstly using personality scores which add up to the result “ENFJ – ELEPHANT”. This test is expected to pass.

The code was tested again using a different expected result ,“ESTJ – BEAVER”, but with **the same personality scores**. This test is expected to fail.

```

class Test_Personality:

    def test_person(self):
        result = PersonalityGrouper(10, 2, 10, 2, 1, 4, 5, 8)
        expected = set('enfj - elephant')
        assert result == expected

class Test_Personality2:

    def test_person2(self):
        result = PersonalityGrouper(10, 2, 10, 2, 1, 4, 5, 8)
        expected = set('estj - beaver')
        assert result == expected

```

Figure: Tests for assessing personality code. First test with correct information, second test with incorrect information.

Assess

The tests performed in the expected way. Test 1 passed whilst test 2 failed.

```

..... 1 failed, 1 passed in 0.13s .....
(testenv) C:\Users\abbie\OneDrive\Desktop\connect2_master\Connect2-Software-Project-master\Testing\greeting> pytest
===== test session starts =====
platform win32 -- Python 3.8.5, pytest-6.2.4, py-1.10.0, pluggy-0.13.1
rootdir: C:\Users\abbie\OneDrive\Desktop\connect2_master\Connect2-Software-Project-master\Testing\greeting
collected 2 items

tests\test_greeting.py .F [100%]

===== FAILURES =====
Test_Personality2.test_person2

self = <test_greeting.Test_Personality2 object at 0x00000266E5757DF0>

    def test_person2(self):
        result = PersonalityGrouper(10, 2, 10, 2, 1, 4, 5, 8)
        expected = set('estj - beaver')
        assert result == expected
AssertionError: assert {' ', '-', 'a...'f', 'h', ...} == {' ', '-', 'a...'e', 'j', ...}
Extra items in the left set:
  'l'
  'f'
  'p'
  'n'
  'h'
Extra items in the right set:...
...Full output truncated (6 lines hidden), use '-vv' to show

tests\test_greeting.py:18: AssertionError
===== short test summary info =====
FAILED tests\test_greeting.py::Test_Personality2::test_person2 - AssertionError: assert {' ', '-', 'a...'f', 'h', ...} == {' ', '-', 'a...'e', 'j', ...}
===== 1 failed, 1 passed in 0.11s =====

```

Figure: test_person test passed. Test_person2 failed.

Conclusion:

The test passed.

11.3.2 Unit Test: Assigning Personality to Job

def createJob (available in employer/views.py)

Test Description:

Testing the code which is responsible for:

1. Taking in the employer job description data from the user
2. Running the personality prediction model
3. Predicting the jobs ideal personality type
4. Assigning the job its type

Why is this being tested?


```

class Test_Job:

    def test_job(self):
        result = AnimaltoJob(
            "logical, rational, able to think of ideas, loves systems")
        expected = set('owl')
        assert result == expected

    def test_job2(self):
        result = AnimaltoJob(
            "skilled at turning creative ideas into reality, focus small details and find solutions to crises")
        expected = set('fox')
        assert result == expected

    def test_job3(self):
        result = AnimaltoJob(
            "hardworking, dedicated, detailed planner, works well within a team, dependable and reliable")
        expected = set('beaver')
        assert result == expected

```

Figure: The tests for the personality prediction function. The result variable resembles the user input data collected from job descriptions.

Test 1 contains user input. This description best fits the owl personality and therefore the model is expected to predict owl from this input data. Additionally, Test 2 contains information which best describes the fox personality whilst test 3 best describes the beaver personality. These tests are subsequently expected to predict these personalities.

The result variable represents the text-input data which is collected from the employer in the original code.

Act:

The 3 tests were ran using PyTest. All three tests passed indicating the modal is capable of accepting user information and making an accurate prediction.

```

=====
===== 3 passed, 102 warnings in 4.16s =====
=====

```

Figure: Results from the 3 Job Tests

Conclusions:

The test passed.

11.3.3 Unit Test: Ranking Applications (available in employee/views.py)

Test Description:

Testing the code responsible for ranking applications to employers in descending order of suitability. The code compares the skill set of the applicant and their personality with the requirements of the job and scores them. When the employer renders the job dashboard, the applications appear in descending order.

Why was this tested?

This code was tested because it is a main feature and attraction of Connect2. This function acts as an applicant ranking software and ranks applications based on their suitability to the job.

Arrange:

The code was arranged to output values rather than create objects. Sample data for the job was created. The job in this test case was specified to be looking for an owl personality and skills in Java, Python and Testing.

```
def RankingApplications(person_type, employeeskills):
    job_type = "owl"
    jobskills = {'Java', 'Python', 'Testing'}
    match = False
    score = 0
    intersection = employeeskills.intersection(jobskills)
    intersection_as_list = list(intersection)

    if job_type == person_type:
        match = True
        score += 1

    else:
        match = False

    for x in intersection_as_list:
        score += 1

    return score
```

Figure: Arrange function to test applicant ranking system.

Assembling:

Three tests were carried out to assess if the system could sufficiently accept applications and rank them accordingly. These tests consisted of three applications being made to the sample job in the above figure. The expected value is the expected score these applications should have based on their matchiness with the job listing.

```

class Test_Ranking:

    def test_ranking(self):
        result = RankingApplications("owl", {'Java', 'Python', 'Testing'})
        expected = 4
        assert result == expected

    def test_rankin2(self):
        result = RankingApplications("owl", {'Cleaning'})
        expected = 1
        assert result == expected

    def test_rankin3(self):
        result = RankingApplications("fox", {'Java', 'Python'})
        expected = 2
        assert result == expected

```

Figure: Tests containing applications with various personality types and skill sets.

Act:

The three tests passed. This indicates the code can respectively rank applications based on the candidates' skills and personality match with the job.

```

===== 3 passed, 2 warnings in 1.25s =====

```

Figure: Tests for applicant ranking system passing

Conclusion:

The test passed.

12. Evaluation and Conclusions

The Connect2 project successfully managed to implement all the functional requirements specified in the original project proposal submitted in December 2019. From the testing results, we can see that each function created for the website is working properly and according to the RS. The project oversaw a recruitment website being created from the ground up. Furthermore, it saw the introduction of a new approach to applying to job with its 'blind' application approach. The custom applicant ranking system was implemented successfully also, assisting employer users with their recruitment needs. Finally, the messaging and notification system allows recruiters and candidates to interact with one another and schedule interview.

From the machine learning model evaluation, it can be agreed that the performance of the personality classifier was high with an accuracy of 90% on with the training data set and passing 3 separate unit tests. The model was implemented to Connect2 to offer a unique way for recruiters to discover which type of candidates they are looking for.

In conclusion, the Connect2 project offers employer users a complete web application that supports all processes involved from posting a job to scheduling interviews with select candidates. For employee users, it offers a place to find jobs, apply anonymously and receive interviews. It can be

stated that the Connect2 project successfully achieved its vision and goals to tackle the issues of recruitment discrimination. The idea of introducing machine learning and blind applications to the recruiting processes was executed resulting in a unique software solution.

13. Further Development or Research

An interesting area of research the Connect2 project could consider for further development would be the idea of introducing pre-employment assessment tests. Such tests can be grouped into different categories depending on what the content of the questions are:

- Hard skills tests
- Work sample tests
- The interview assessment
- Cognitive ability tests
- Personality tests

Adequate pre-recruitment tests are an effective method of uncovering which candidates are most qualified for a particular job. Scientifically based tests (such as cognitive assessments) can increase the likelihood of selecting ideal candidates for jobs. Using the results from these tests to make decisions can often be more accurate than human judgment, due to the cognitive biases humans have as mentioned previously.

Connect2 could be developed by introducing standardised pre-employment assessment tests. These tests would be optional for the employee user to take and would be attached with their application made to jobs. The ranking systems could be updated to incorporate the result from these tests in the applicant ranking system and similarly the job recommendation system could begin to suggest jobs to employees based on their stronger points in the assessment tests.

14. References

Agarwal, P., 2020. Gender Bias In STEM: Women In Tech Still Facing Discrimination. [online] Forbes. Available at: <<https://www.forbes.com/sites/pragyaagarwaleurope/2020/03/04/gender-bias-in-stem-women-in-tech-report-facing-discrimination/?sh=5c145a3470fb>> [Accessed 15 May 2021].

Central Statistics Office Ireland, 2019. Equality and Discrimination. Statistical Yearbook of Ireland 2019. Dublin: Central Statistics Office Ireland.

Colossus Media Group, 2019. Benefits of MBTI In the Workplace - Colossus Media Group. [online] Colossus Media Group. Available at: <<https://cmg-agency.com/benefits-of-mbti-in-the-workplace/#:~:text=Statistics%20show%20that%20more%20than,as%20a%20team%2Dbuilding%20exercise.>> [Accessed 15 May 2021].

Kiersey, 2021. 4 Types. [online] Kiersey.com. Available at: <<https://keirse.com/temperament-overview/>> [Accessed 15 May 2021].

Kroeger, O., Thuesen, J. and Rutledge, H., 2002. Type talk at work. New York: Dell.

McGinnity, F., Lunn, P. and Quinn, E., 2009. Discrimination in recruitment. Roscrea, Co. Tipperary: Equality Authority.

Pozniak, H., 2020. The bias battle: how software can outsmart recruitment prejudices. [online] the Guardian. Available at: <<https://www.theguardian.com/careers/2020/dec/23/the-bias-battle-how-software-can-outsmart-recruitment-prejudices>> [Accessed 15 May 2021].

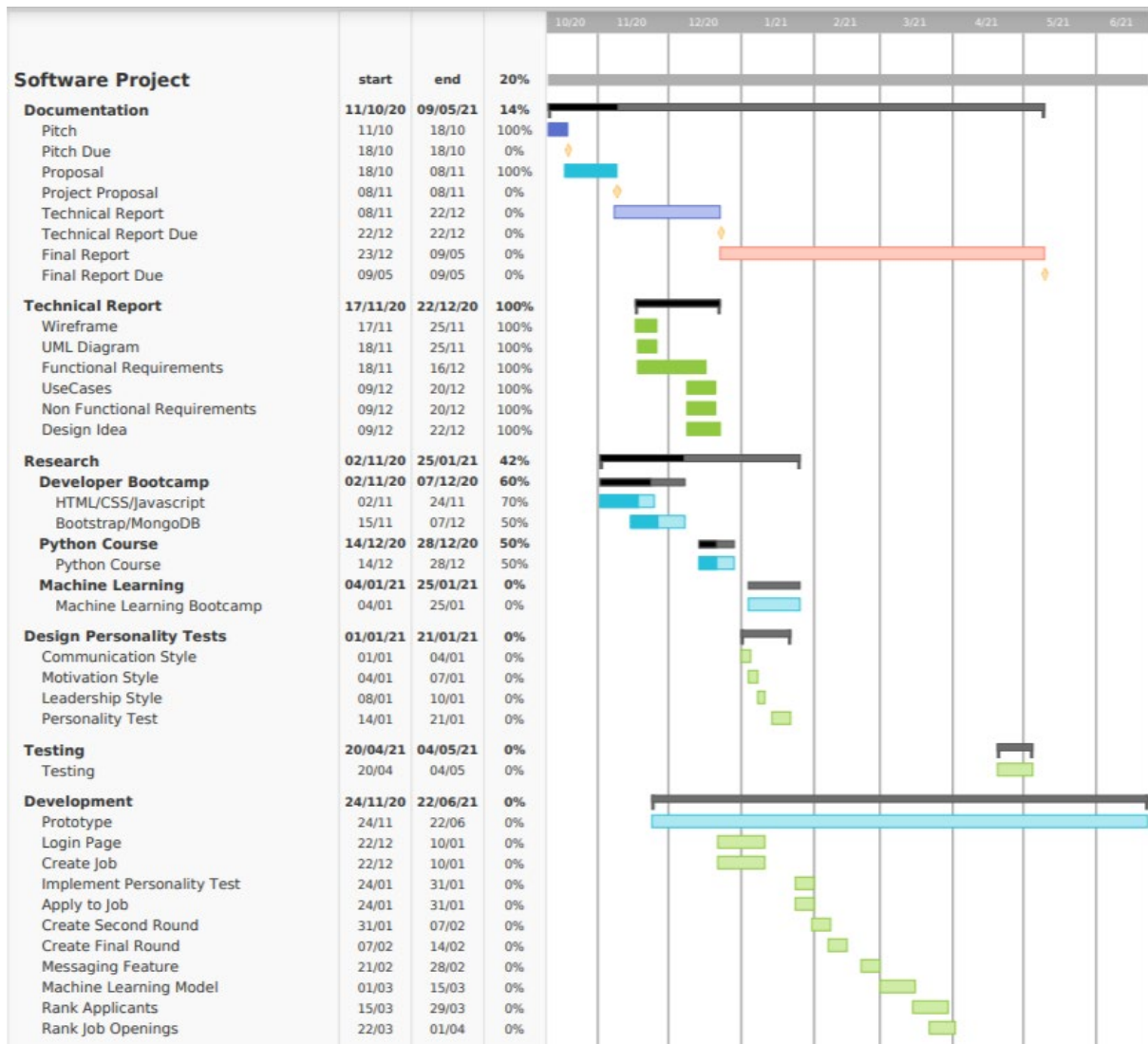
Zalcastin, 2013. MBTI frameworks compared with Keirsey. [image] Available at: <<https://zcalstin.files.wordpress.com/2016/03/keirsey.jpg>> [Accessed 15 May 2021].

Zhou, V., 2019. A Simple Explanation of the Bag-of-Words Model. [online] Medium. Available at: <<https://towardsdatascience.com/a-simple-explanation-of-the-bag-of-words-model-b88fc4f4971>> [Accessed 16 May 2021].

15. Project Plan

Below is the refined project plan

Figure 7: Project Gantt Chart outlining the activities over the course of the next few months



16. Project Proposal



National College of Ireland

Project Proposal

Business Information Systems

2020/2021

Abigail Boyle

X17323943

X17323943@student.ncirl.ie

Contents

1.0 Objectives.....	55
2.0 Background	57
3.0 Technical Approach	59
4.0 Special Resources Required	60
5.0 Project Plan	60
6.0 Technical Details.....	Error! Bookmark not defined.

Objectives

As part of the 4th year software project, a web application, accessible through a browser will be designed and produced to assist two user groups, individuals and organisations, with their hiring processes. The web application will act as a means for professionals to create accounts to be posted for employers to view and short list. The website will take the users

through a profile generation process that will be created with tools that can take in user input, analyse it and output a professional profile to illustrate the persons relevant strengths and characteristics to employers.

The web development will be targeted at graduate-level positions or higher.

When individuals begin their signing up, they will be requested to supply their personal **information**, knowledge areas and skills. They will have the option to include a professional picture. Once complete, the application will bring the user through a series of **psychometric tests**, of which will be centred around existing testing used today by various applicant tracking software and companies. Examples of these tests include:

- The Myers Briggs Type Indicator [1]
- DiSC [2]
- Hogan Personality Inventory [3]

The personality test which the system will implement, will combine factors of these three personality tests into one single test. It will be designed to measure how a person meets a variety of organisational priorities such as:

1. Coaching
2. Communication
3. Conflict management
4. Culture change
5. Customer service training
6. Onboarding
7. Personal development
8. Professional development
9. Teamwork

The system will take the answers and results from the testing in as data and analyse it for patterns and indicators to the persons personality and skills. A machine learning model will be trained with a dataset containing 8600 rows of information regarding the different personality types in the MBTI framework. The model will characterise the user and predict their temperament and needs. The profile will provide interested employers with information on the individual such as:

- Which employees will be motivated by financial incentives, social recognition, or true stories of positive impact on clients?
- Which manager candidates will drive toward stability vs. innovation?
- Will the candidate blend successfully into the existing team or could the new hire be detrimental to the existing high-performing project?

On completion of the standardised testing, the profile will be outputted listing personal information combined with their test results that is viewable by both the individual and interested organisations. The purpose of the tests will be to illustrate the **positive aspect** of their results to employers rather than possible weaknesses. Candidates will have the option to indicate whether they are actively pursuing employment or rather open to offers, how far

they are willing to travel for work, level of comfort with working remotely, desired working hours etc.

Employee users will be able to view the available jobs on the website. The system will display highly matched jobs at the top of the page and allow users to apply directly with the profile that has been generated.

Employer users will have the opportunity to post available positions in their company and include the ideal candidate description with the job posting. Employers can browse through the applicants to their job opening. The system will display highly matched candidates at the top of the page and less matched candidates at the lower part. The system will give a percentage which will resemble how matched a profile is to a job description.

The web application will **allow employers** to browse professional accounts who have applied to positions and offer the ability to advance appropriate candidates. A **chat box** will be provided as a method of contact between the two parties. Additionally, employers will be able to apply **filters** when browsing. Employers will be able to create job listings viewable to employee users and have applicants apply directly with their profile through the web app. The ML model will predict which employees are highly matched to the organisations open positions and vice versa.

When an employer user is happy they have found an appropriate candidate/candidates to interview, a request for such will be issued to the employee user. The employee user has the choice to accept the interview offer and share their personal details with the employer or decline the interview. It is only when the employee user accepts the interview invitation that the employer will be able to view their personal details.

The objective of the application is to provide the means for employers and employees to find suitable fits for interview. Candidates will agree to provide honest information on themselves so that employers receive accurate representations which have not been altered to suit the needs of specific positions or organisations.

Effectively, the project hopes to reduce the percentage of bad hires and unsatisfied employees through characterising users with machine learning and offer a space for individuals to be honest about their skills and experience through the usage of standardised personality and ability tests.

Background

As of modern times, employers and candidates have numerous resources to find each other including job posting websites, LinkedIn and company websites. Since the early 2000s, companies have been using the internet as the initial part of their hiring process and in turn candidates have created digital CVs. Nowadays, it can be agreed a huge percentage companies use the internet for hiring purposes and applying with a paper CV to a position is rather out-dated.

A corporate job posting receives, on average, 250 applications [4] Of these, typically only four to six can be interviewed due to both time and cost restraints. Additionally, large

companies can receive 50-70k resumes per week [4]. With those figures, employers are overwhelmed and need support to work through each application and select suitable candidates.

Applicant tracking software's were designed to tackle the difficulties of working through the high volume of applications. They are used to scan through CVs and applications for key information and automatically forward or erase those which either do or do not fit the job description.

Although these applicant tracking systems certainly make life easier for hiring managers, applicants are aware of their existence. In a study conducted in 2017, as many as 85% of job seekers admit to lying on their resume [6] to either beat the applicant tracking system or progress further in the interviewing process. The lies include, embellishing the work experience or education [6] they have or even directly copying the skills required from the job description.

In effect, many candidates are supplying inaccurate representations of themselves to employers and regularly changing their CV to suit the applicant tracking system used for a certain job. Furthermore, in a study conducted by career builder, 43% of managers admit to having insufficient background checks or assume nets to obtain qualitative data about the applicant beyond their CV[5].

This means hiring managers are often making hiring decisions based off documents which have been embellished by the applicant to suit the job description.

The idea for the project sparked from this loophole in today's professional world. Although in specific scenarios, there is not anything lawfully or ethically wrong about tweaking applications to stand out, I believe it is an issue that companies are relying on applicant tracking software to automatically forward and disqualify applicants based off documents which are misleading or embellished.

The website will reverse the standard process involved when applying for a job. Rather than having employers post an opening position with the experience/skills required, **employers will seek out individuals based off their personality and skills.**

The profiles will be generated from the users answers and results from the standardised tests. There will be between 3 – 5 types of tests which will assess the persons personality. The results will be formatted on the individuals account to demonstrate to potential employers their way of thinking, dealing with problems and communicating.

The **profiles will be centred around the individual's strengths, skills and characterises.** The website will also allow individuals to highlight their personal achievements, biggest dreams, hobbies, and goals.

The website will be targeted at all professional individuals and companies. It will use the "honest" profiles to drive hires based off an individual's culture, attitude, and ethics rather than experience and educational certificates. It will not include the person's name, birthday or nationality. Users will create a tagline for their profile which will describe them briefly to

browsing employers. This will attempt to eliminate employment discrimination around these three factors.

Unlike LinkedIn, there will be no interaction between users on the website or indication of the persons professional network or who they know. The only interaction users will have is direct messaging from potential employers.

Companies using the website can browse and filter through the candidates available by line of work, skills, personality, location etc. They can create lists on the website to save potential candidates for review at a later point in time.

Organisations with a job opening can post a position available with the relevant employment details and the ideal candidate description. The job will be posted for the employee users to view and browse. Employee users can apply with their generated profile and employer users will receive such applications for review.

Technical Approach

The application is being designed for individuals seeking employment and organisations hiring employees. Due to companies typically comprising of office workers and desktops, the application will be designed as a web application accessible through a browser.

The website will be designed and produced using HTML, CSS and JavaScript. In addition to those technologies, Bootstrap, Django and PostgreSQL will be used as web development technologies. An interactive organised interface will be designed to allow both user groups sign up, complete assessment tests, and generate their profiles. The assessment tests will be built with Python.

Python(scikit-learn) and machine learning will be used for the back end to extract answers and analyse data from the test results. The system will be able to detect user input as quantitative data and output the user profiles containing the individual's traits and characteristics. The system will begin to identify patterns in human answers and link such to human personality traits.

User Functional Requirements

There will be two groups of users:

1. Employee users
2. Employer users

Functional Requirements

- Users can create an account with login details.
- New employee users create profile with details.
- New employer users create profile with details.
- New employee users complete profile generation process.
- Employee user profile is generated from user input and testing results.
- Employee **user has control over their profile** and can edit their personal information but cannot edit their test results.

- Employer user profile is created from user input only.
- Employee user profile is posted to portal.
- Employer user profile is posted to portal.
- Employer users can filter candidates with various filters.
- Employee users' profiles will be displayed to employers with a level of anonymity.
- Employer users can advance candidates suitable for open positions.
- Employer users can interact directly with employee users through the message box provided.
- Employer users can post available positions in their company.
- Employee users can filter available positions with various filters.
- Employee users can view the available jobs.
- Employee users can apply directly to open positions with their generate profile.
- Employer users can arrange an interview with employee users they wish to advance.

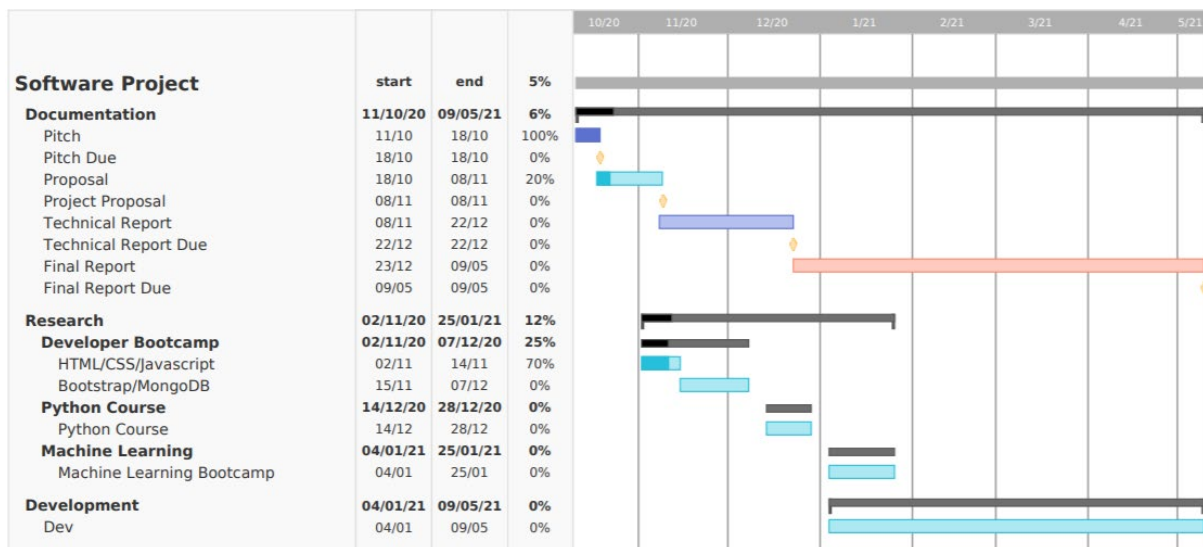
Special Resources Required

A collection of Udemy courses will be used as references to help with learning the technologies involved for the development of both the web application and machine learning model:

- The Web Developer Bootcamp 2020 – Colt Steele
- 2020 Complete Python Bootcamp: From Zero to Hero – Jose Portilla
- Python for Data Science and Machine Learning – Jose Portilla

Project Plan

Below is a Gantt chart with the rough project plan outline:



Potential Technologies to be used

The implementation languages which are being considered include:

- HTML
- CSS
- JavaScript
- Python (with ML libraries like scikit-learn)

- PostgreSQL
- Bootstrap
- Django
- Heroku

References

1. Myersbriggs.org. 2020. The Myers & Briggs Foundation - MBTI® Basics. [online] Available at: <<https://www.myersbriggs.org/my-mbti-personality-type/mbti-basics/>> [Accessed 8 November 2020].
2. Discprofile.com. 2020. What Is Disc®. [online] Available at: <<https://www.discprofile.com/what-is-disc>> [Accessed 8 November 2020].
3. Hogan Assessments. 2020. Hogan Personality Inventory | Hogan Assessments. [online] Available at: <<https://www.hoganassessments.com/assessment/hogan-personality-inventory/>> [Accessed 8 November 2020].
4. Sullivan, D., Sullivan, D. and Archive, A., 2020. Why You Can'T Get A Job ... Recruiting Explained By The Numbers. [online] ERE. Available at: <<https://www.ere.net/why-you-cant-get-a-job-recruiting-explained-by-the-numbers/>> [Accessed 8 November 2020].
5. Jaye, M., 2019. Pitfalls Of The Incomplete Background Check - HR Daily Advisor. [online] HR Daily Advisor. Available at: <<https://hrdailyadvisor.blr.com/2019/10/25/pitfalls-of-the-incomplete-background-check/>> [Accessed 8 November 2020].
6. O'Donnell, J., 2017. 85 Percent Of Job Applicants Lie On Resumes. Here's How To Spot A Dishonest Candidate. [online] Inc.com. Available at: <<https://www.inc.com/jt-odonnell/staggering-85-of-job-applicants-lying-on-resumes-.html#:~:text=According%20to%20HireRight's%202017%20employment,feel%20the%20need%20to%20lie.>>> [Accessed 8 November 2020].

17. Reflective Journals

October Reflective Journal

4th Year Software Project - Reflective Journal (October 2020)

Week 1

For the first week of the semester, I spent time brainstorming and researching possible ideas for my software project. I listed the technical skills I have gained over the pervious three years and attempted to envision how I could put such skills to use and design software. I concluded that my strongest technical skills were in the following areas:

1. Front-end web design
2. Android Development
3. Back-end web design

I decided that I would develop a web application for my project and if possible, attempt to create an Android application linking to the website. By doing so I would be both using the technical skills I have and deepening my knowledge on both topics.

I also considered which area of IT I would like to invest into and learn. After some evaluation, I settled on the topic of machine learning as I believe it would be highly beneficial for my own learning and career development after graduation. In line with the subject of machine learning, I decided to also invest into learning python.

Week 2

During the second week of the project I focused on choosing a topic for my application to be based around. Firstly, I considered other interests I have outside of my degree. I have always had an interest in human psychology and screening tests used by psychologists today to evaluate individuals. Additionally, I have always enjoyed investigating hiring processes and candidate requirements involved when companies are seeking to fill positions. I have become adapt at producing CVs for friends and family to apply for jobs.

I researched existing applications in use nowadays such as LinkedIn & applicant tracking software's. I had the idea initially of creating a website employer can post jobs to and have applicants apply. I considered then building my own applicant tracking software into the website to analyse CVs for employers. Upon assessment however, I did not believe this filled a gap in the market or was particularly exciting or interesting.

Next, I drafted the idea of creating a website which would reverse the standard job applying procedure which is the candidate sends their modified CV to the employer for a position posting. I brainstormed an application which would allow job seekers to create profiles for themselves listing their skills and determined strengths for employers to approach them.

Due to my interest in psychology, I dabbled with the idea of putting existing personality tests (such as MBTI) into the profile generation process. The web application would use machine learning to detect the candidate's skills and traits by their answers in such tests.

Week 3

Once I had a general idea for what my project would be based on and how I would create it, I began to create my project pitch on PowerPoint. To do so, I had to conduct primary market research into the existing similar software's today, such as LinkedIn.

The project pitch took several days to complete. The result was a six-minute-long video explaining the project idea, purpose and technical work involved.

Week 4

Once the pitch was submitted, I invested time into sourcing credible resources to assist my learning on the technology I intend to use to develop the explained application. After evaluation I decided to purchase 3 courses on Udemy.com. They are:

Web Application Development boot camp

Beginners to Advanced Python

Machine Learning through Python.

I started the Web Development bootcamp immediately and hope to finish the course in three to four weeks before moving onto the next one. I believe these courses will contribute considerably to the outcome and quality of my project.

November Reflective Journal

Week 1:

I had my meeting with my project supervisor, Thanos, during the first week of November on Ms Teams. During this meeting I walked Thanos through my idea and vision for my project and exactly what it was I wanted to achieve. Thanos recommended some machine learning libraries, such as Scikit-Learn, to look into for building the matchmaking model required for my project. Meeting with supervisor helped a lot as it was the first time I received feedback on my project idea. We agreed that we would contact each other through email and have Teams meetings every second week.

The next item we discussed in our meeting was the project proposal. I sent my original draft to Thanos after our teams call and he sent back feedback and recommended changes to make. I submitted my completed proposal on November 8th.

Week 2:

For the second week of November, I began to map out the requirements for Connect2. I started by thinking about what each user would want to achieve on the website and go from there. Initially I had over 20 requirements but I refined them to about 15 by the end of the week.

Outside of that I continued with the web development bootcamp on Udemy and began to learn about the Bootstrap framework and refresh my memory on JavaScript.

Week 3:

During week 3 of November, I began the Python and Django course and downloaded Visual Studio Code to begin coding along with the course. The Django course walked me through how to set up a Django application on your computer and how the models in Django work. I also completed a quick 2 hour course on Github to refresh my memory on it.

I had my second teams meeting with Thanos where we discussed the requirements I had drafted. My supervisor made some recommendations for the environmental and usability requirements.

At the end of the week I designed several basic wireframes to visualise what the website pages might look like.

Week 4:

During week 4 I began creating prototype for Connect2. This involved watching the Django course on Udemy as well as several Youtube videos with explanations and suggestions on how to code specific features. I set up the Admin dashboard for the application and created classes for the user profiles.

I began formatted my website with HTML mark-up and Bootstrap. I continued the Udemy courses on Python, Django and web development.

I also began to work on my Technical Report, starting with the functional requirements. I drafted a Use Case for each requirement as well as a Use Case Diagram.

December Reflective Journal

Week 1:

I continued to develop the prototype for Connect2. This involved building HTML/CSS templates for each screen and beginning to display information from the back-end on the website. I did not meet with my supervisor this week because I decided that I was going to continue work on the prototype and demo this work the following week during our call.

I continued to develop my technical report draft by finalising the list of use cases to 14.

Week 2

During the second week of December, I had a call with Thanos, my project supervisor. During this call I demoed the prototype I had developed and we also discussed how my technical report was going. Thanos asked that I continue to work on the technical report and email him the draft the following week.

I worked on the technical report for the remainder of the week, building upon the functional and non-functional requirements. When I emailed the draft to Thanos, the document had grown to be 7,000 words in size.

Week 3:

During week 3 I received feedback from my supervisor on my technical report draft during an MS Teams call. Thanos made a lot of suggestions and recommendations for change such document formatting and structure. He suggested I begin to work on reducing repetitions in the document, especially in the use cases, and create base use cases to refer to at later points. Another suggestion was to expand more on the personality aspect of the project and perform more research on this topic.

For the remainder of the week I edited the document until all of the suggestions to change were made and the structure of the document was improved. In addition to such, I reviewed my project proposal and added it into the report. The report was submitted on December 18th.

Outside of the document, I created PowerPoint slides to present during my mid point video presentation. The slides contained a brief overview of the content in my project report whilst also explaining the plan.

Whilst recording the video presentation I also included a brief demo of the prototype created in November and week 1 of December, outlining which elements of the web app had been created. This video presentation was also submitted on December 18th.

Week 4:

During week 4 I continued to develop the prototype, creating a login system for users to register and create user profiles. In addition to such I continued the Udemy courses to further my knowledge in web application development and machine learning.

January Reflective Journal

Week 1:

During this week I had exams and TABA's due for semester 1 to prioritise.

I spent approximately 3 hours on my Python/Django and machine learning courses.

Week 2:

Exams/TABA for semester 1.

Week 3:

To get back into my project work I decided to restart the project on visual studio code and reuse parts of the existing prototype because I had become slightly unfamiliar with the existing code and had to refresh my memory. I continued with my courses mentioned above.

Week 4:

During this week I developed the HTML pages with their layout for each page required in the project. I emailed my supervisor to arrange for a meeting next week to review the midpoint grade and the existing work.

February Reflective Journal

Week 1:

During this week I created models for the two different user types that are going to exist within my application, employers and employees. I designed the models so that each user within my application would be assigned to either an employee or employer user profile. This was achieved through the use of foreign keys.

I met with my supervisor via Teams. We discussed my mid point submission and I received a break down of my marks. My supervisor additionally offered some suggestions and advise.

Week 2:

During week 2 I implemented register/login/logout functions which allow a user to register to the Django app, authenticate and login the user, redirect the user to the correct application and logout. I created two different apps to exist within the project, one for the employer and one for the employee, each with their own pages/nav bar etc.

Week 3:

During this week I focused on adding the skill, experience, and job models to the system. By adding such models, employee users are now able to add relevant skills and employment history to their Connect2 user profiles. Additionally, employer users are able to post available job positions from their company.

Week 4:

During week 4 I had another meeting with my supervisor to demonstrate my application so far. I began to work on the personality application, which requires setting up models to store the personality test with the questions. I also finalised my ethics form submission.

March Reflective Journal

Week 1:

During this week I designed the questions which are going to be used to understand what personality type the employee user is. I began to create the HTML pages which will be used to ask the questions.

Week 2:

During this week I added the ability to create a job to the website. Employers could create a job object which had the relevant information for the job. Additionally, I created job skills and job tag models to store the requirements of jobs.

Week 3:

During this week I had a meeting with my supervisor where we discussed how the machine learning element of Connect2 was going to be introduced. I began researching into the best approach.

Week 4:

During this week I began to collect data for my personality model classifier. This model needs a lot of data so more than 4 days were spent collecting the data.

Additionally, I began to work on the ability for employees to apply to job and create applications.

April Reflective Journal

Week 1:

This week I focused on building the machine learning model with the collected personality data. I had a meeting with my supervisor where I demonstrate my progress on the machine learning aspect.

Week 2:

During this week I introduced the machine learning aspect to the Django app and created information pages to explain to the employer which animal work group they were being assigned to their job listing.

Week 3:

During this week I created the applicant ranking system which ranked applications based on their matchiness with the job. I also created the ability to highlight, delete, interview candidates.

The notification system was also created.

Week 4:

During this week I finished the full process of apply to a job and receiving an interview request. The ability to share user profile details with recruiters once an interview was scheduled also was complete.

May Reflective Journal

Week 1:

During this week I fixed the web applications design, improving the CSS and overall look of the website.

Week 2:

During this week, the finishing touches were added to the web application.

The application was also tested and the final technical report was completed.

