



# DEVELOPMENT OF A CAREER RECOMMENDATION WEB APPLICATION UTILISING THE MBTI INSTRUMENT

This dissertation was submitted in partial fulfilment  
of requirements for the degree of MSc Software  
Development

Department of Computer and Information Sciences University of Strathclyde

Thomas Cotton  
Registration Number: 202082932  
August 2021

## DECLARATION

This dissertation is submitted in part fulfilment of the requirements for the degree of MSc of the University of Strathclyde.

I declare that this dissertation embodies the results of my own work and that it has been composed by myself.

Following normal academic conventions, I have made due acknowledgement to the work of others.

I declare that I have sought, and received, ethics approval via the Departmental Ethics Committee as appropriate to my research.

I give permission to the University of Strathclyde, Department of Computer and Information Sciences, to provide copies of the dissertation, at cost, to those who may in the future request a copy of the dissertation for private study or research.

I give permission to the University of Strathclyde, Department of Computer and Information Sciences, to place a copy of the dissertation in a publicly available archive.

(please tick) Yes ☒ No ☐

I declare that the word count for this dissertation (excluding title page, declaration, abstract, acknowledgements, table of contents, list of illustrations, references and appendices is: 8716 words.

I confirm that I wish this to be assessed as a Type 1 2 3 4 5 Dissertation (please circle)

Signature:



Date: 15/08/2021

## Abstract

*Deciding on a career can be a major source of stress and anxiety for those who are unsure about what their career aspirations are, often causing them to settle for an available option that may cause them further distress and unhappiness. The purpose of this project was to build a recommender system that recommends careers to users, making suggestions based on specific aspects of their personality type and assisting them in their career search.*

*This report documents the entire software development lifecycle from requirement gathering to user validation. The research conducted around personality type theories and recommender systems is discussed in detail to outline the foundation on which the system is built and how psychology theory was incorporated into the software. The decisions made during the development life cycle including requirements, design, methodology and construction are detailed to outline how the final minimum viable product was produced. Testing and analysis on each aspect of the development life cycle is then discussed in detail, displaying how the results from this analysis helped to inform the rest of the project and improve the overall quality of the application produced.*

*This project concluded in the development of a web application that can successfully recommend careers to potential users by means of a personality type questionnaire. While the product produced is only a minimum viable product to demonstrate the functionality of the recommender system, with the further workings outlined in the final chapter implemented into the system, the application could be successfully deployed on the web.*

*I would like to thank Dr William Bell for his support and guidance throughout the duration of this project.*

## Contents

Abstract.....	1
Introduction .....	5
Project Motivations.....	5
Careers and Psychology .....	6
Myers Briggs Type Indication.....	6
MBTI and the Workplace .....	6
Additional Model Considerations .....	6
MBTI Drawbacks .....	7
Recommender Systems .....	8
Methodology.....	9
Requirements.....	10
Requirements Gathering.....	10
User Stories.....	11
Prioritisation.....	12
Non-Functional Requirements.....	12
Design.....	13
System Architecture.....	13
User Interface Design.....	14
Data Design .....	19
Low Level Design.....	20
Backend Application .....	20
Application Programming Interface Design .....	20
Implementation .....	21
Introduction .....	21
Technologies and Frameworks .....	22
React Application .....	22
Django Web Service .....	22
Development.....	24
React Development.....	24
Django Development .....	27
Project Management .....	29
Other Tools .....	32
Version Control .....	32
Evaluation .....	33
Requirement Analysis .....	33

INVEST Principles .....	33
Prioritisation Analysis.....	33
Public Requirements Survey .....	34
Design Analysis.....	34
Heuristics UI Evaluation .....	34
Database Design Evaluation.....	35
Prototyping .....	35
Verification Testing .....	36
Unit Testing.....	36
Integration Testing.....	37
User Acceptance Testing.....	39
Conclusion.....	40
Further Work.....	40
Bibliography .....	41
Appendix .....	42
Appendix A – Prioritised User Story Table .....	42
Appendix B – User Interface Wireframe.....	45
B.1 – Home Screen and Navigation Bar .....	45
B.2 – Quiz Screen .....	46
B.3 – Result Screen.....	47
B.4 – Login Screen.....	48
B.5 – Register Screen .....	48
Appendix C – Entity Relationship Model .....	49
C.1 – Entities and Attributes .....	49
C.2 Entity Relationship Table .....	49
Appendix D – REST API Documentation.....	50
Appendix E – Requirements Survey.....	65
Appendix F – Heuristic Design Analysis.....	73
Appendix G – User Acceptance Tests .....	76
Appendix G.1 – Session One .....	76
Appendix G.2 – Session Two .....	77
Appendix G.3 – Session Three.....	78
Appendix H – Acceptance Test and Story Mapping Table .....	80

## Introduction

Choosing a career is one of most important decisions that a person will have to make in their lifetime. A study was performed in 2016, using average working hours and projected years spent working, and found that on average people in Britain will spend 35% of their time during their working life at their workplace.<sup>1</sup> Given that this is quite a significant portion of a person's life, coupled with the fact a person's career has a substantial impact on their mental wellbeing,<sup>2</sup> making the correct choice early is significant. The other issue faced is that of career change. Sometimes it is necessary to change career later in life, where the change might be non-obvious to the person. To address these issues the brief for the project was as follows:

*“At school and during the working life, it may be necessary to choose a career path that is non-obvious. Those at school and those thinking of changing career may be unaware of career paths and trends that are available. The project goal is to build a web application that provides career path suggestions based on a personality test, followed by a set of additional questions. These additional questions may include asking the user if they would be willing to learn a new subject area. The output of the search could be a graphical display that allows the user to filter the results and look at careers that are further from their skillset. Realtime filtering and postprocessing could be implemented.*

”

## Project Motivations

The rise of the information age has led to dramatic changes in the way people handle making career decisions. In the past your career path was often decided by what area or community you lived in, now countless opportunities are available anywhere in the world at the click of an “Apply” button. While this is a fantastic thing and can inspire people to live out their dreams, this sheer volume of availability can add extra stress to those who are unsure of their own career aspirations. The purpose of this project is to build a tool that would take away some of the ambiguity involved in searching for a career and give those individuals some direction in their search, by building a recommendation system for potential careers based on personality type theory. The users will be able to take a quick personality assessment and have careers recommended to them based on their answers. As these are just recommendations, it is important that the user is left with enough information to come to their own conclusions. This presents them not only with an opportunity to further their career aspirations but also help them to learn more about themselves, and as a by-product help them in other aspects of their life.

## Careers and Psychology

### Myers Briggs Type Indication

The Myers Briggs Type Indicator (MBTI) was developed by mother and daughter team, Katherine Briggs and Isobel Myers.<sup>3</sup> This was created in an attempt make the post WWII era a happier and more peaceful place, by helping people better understand themselves and each other by utilising Carl Jung's personality type theory.<sup>4,5</sup> They converted Jung's theory into a tool that could be utilised to assess an individual's personality and assign them a type, the MBTI. Myers and Briggs declared personality to have 4 separate axes following Jung's theory:

- Extrovert or Introvert (E or I): how you interact with the world around you.
- Sensing or Intuitive (S or N): how you take in the information from that interaction.
- Thinking or Feeling (T or F): how you process and act on the information that you have gathered.
- Perceiving or Judging (P or J): how you respond to your current environment.<sup>6</sup>

Each person can fall on either side of each of the four axes, leading to a personality code of 4 letters being developed.

### MBTI and the Workplace

Research has shown that there is a direct correlation between personality type and career choice, with analysis being conducted on various careers, showing a large % of employees share the same personality type within a chosen job.<sup>7</sup> The MBTI tool has been recorded to be used by at least 70% of companies listed in the fortune 500, to help improve performance, personal relationships and overall happiness in the work place.<sup>8</sup> This highlights the reliance these corporations place on personality assessment tools to bring the right individuals in, especially when a large number of employees gives a less personal relationship with management.

### Additional Model Considerations

When conducting initial research on personality typing to be used in this project it was important to consider all possible avenues that could be used to provide accurate recommendations to the user. One of the early standouts, alongside MBTI, was the five-factor personality model (FFM). A paper written 1992 by McRae and John,<sup>9</sup> consolidated many different theories surrounding personality into a singular model.<sup>9</sup>



The FFM follows a very similar approach to MBTI, however it breaks a person's personality down into the following five axes:

- Extraversion.
- Openness to experience.
- Agreeableness.
- Conscientiousness.
- Neuroticism.<sup>9</sup>

Unlike MBTI, subjects are not assigned a particular personality type depending on where they fall on each end of the axes, instead a measurement of where the subject lies on each axis is taken and given an empirical value, leaving the participant with 1 score for each factor.<sup>9</sup>

FFM has grown widely in popularity among academics in psychology, passing MBTI as the favoured indicator for personality assessment.<sup>10</sup> This has led to many professionals in the field calling for organisations to switch from the traditional MBTI to the more favoured FFM. However, research has shown,<sup>10</sup> that direct correlations can be drawn between each of the MBTI and FFM factors, with neuroticism being the only uncorrelated difference between the two.<sup>10</sup> Given the similarities between the two approaches and the possible monetary implications organisations could face moving to a different approach, this has led the majority to continue utilising MBTI as their main personality assessment tool. It is for that reason the chosen model for this framework would remain using MBTI.

### MBTI Drawbacks

One of the main drawbacks of the type indicator is that it does not account for how an individual's personality may be subject to change over time. If the individual takes the test once, they may believe that their personality type is not subject to change throughout the course of their life, negatively affecting decisions about their career they make in the future. This is in fact not the case, as studies have shown,<sup>11</sup> that personality can change over time just through the aging process itself.<sup>11</sup> It is important if an individual chooses to use the MBTI as a tool to help them better understand themselves, it is taken at extended intervals to allow enough time for their personality to have changed and maintain the accuracy of the data provided by the indicator.

It is also important to note that although there are 16 personality types, each component of the type has its own axis that the individual can fall into within the boundary. Comparing the finite details of two individuals given the same type, both individuals will likely fit differently on the 4 axes. This needs to be accounted for when making career recommendations, for example someone who possess the correct personality type for a job in sales but has scored the bare minimum to be classed as an extrovert will likely suffer in this role in comparison to someone with a much higher extraversion score.

## Recommender Systems

A recommender system is a software system that aims to predict a user's preferences and likes based on a certain data set. There are two forms of recommender system: content filtering or collaborative filtering. Content filtering focuses on building up a user profile based on their interaction with the system and makes recommendations of items that are similar in nature, whereas collaborative filtering makes its suggestions based on two users with shared preferences.

This application allows a detailed user profile to be built up just by completing the quiz alone, therefore content-based filtering would be most appropriate for this project. This application utilises content filtering with vector dot product matching. The dot product is given in Equation 1 and assigns a numerical value to represent the strength of a match between two items.

$$c \cdot p = \sum_{i=personalityComponent}^n c_i p_i$$

*Equation 1. Content Filtration Vector Model, where  $c$  = career personality and  $p$  = user personality vectors.*

Implementing MBTI into the application requires both the user and career data be broken up into a vector model consisting of 4 axes, one to represent each factor in the personality type. Each career and user will be assigned a value for each one of the 4 axes and the dot product is evaluated to provide an overall match score between the two. Although matches between two users with the same personality type are the same, each match is ranked differently depending on the user's score for each factor. This provides a more accurate and meaningful recommendation by accounting for both the user and careers position on each of the four personality axes.

## Methodology

It was decided that the project lifecycle would be managed using an Agile methodology. Although traditional agile methodologies are implemented within teams and involve a lot of communication with the customer, it was important this solo project be conducted this way as Agile methods are by far the most popular method of project management currently used in industry today, so becoming accustomed to this way of working is imperative for personal development.

This project was completed using aspects of the popular methodology, Scrum. The development phase was broken up into iterations known as “Sprints”, lasting two weeks each. Each user story is moved into the “Product Backlog” and at the beginning of each sprint, stories are selected to be moved into the “Sprint backlog”, i.e. all the functionality that will be implemented during that iteration. After completion of a sprint, the first stage of the next sprint would be to analyse and assess the backlog based off the conducted analysis in the previous sprint discussed in Section 7.0.0. Some of the scrum aspects, such as daily stand ups and sprint meetings, could not be used in the project as these are only applicable to teams rather than a sole developer.

To produce a cohesive and well-structured report, the results of each phase are discussed in detail in a progressive manner. In practice, aside from the initial design, each phase occurred concurrently to maintain the agile approach and respond to any necessary changes. Each sprint followed the same pattern:

1. Modify Requirements.
2. Update Design.
3. Implement code.
4. Asses each phase.

## Requirements

Requirements gathering is the process of turning an idea or concept into chunks of information that the developers can use to create the final software product. Due to the fact this project was completed as a solo development project, with a singular stakeholder, it was primarily down to the developer to gather requirements from the various resources available.

### Requirements Gathering

To gather the initial functional requirements for the system the three main sources of information were: the project brief, current systems and the developers own experience. Outlined in Section 2.0.0, the project brief was first consulted in detail to gather all the requirements that would be necessary to match the description and marking criterion. As mentioned in Section 2.2.4, some of the drawbacks of MBTI needed to be amended, therefore preventing immediate quiz retake and calculating match score were added to the requirements.

Primary research was then conducted on similar applications to build out further extended features that could be offered by the system. As the primary function of the application was covered by the project brief, this offered the opportunity to explore other pieces of functionality that would be useful to an end user. The design layouts of each of the systems were also recorded to ensure the applications interface was a familiar experience for users. A final brain storming session was conducted to gather requirements from the developers own job search experience, and all the high-level requirements were recorded on a whiteboard shown in Figure 1.

Potential users were also involved in the requirements gathering process through a requirements analysis survey, discussed in section 7.1.3.

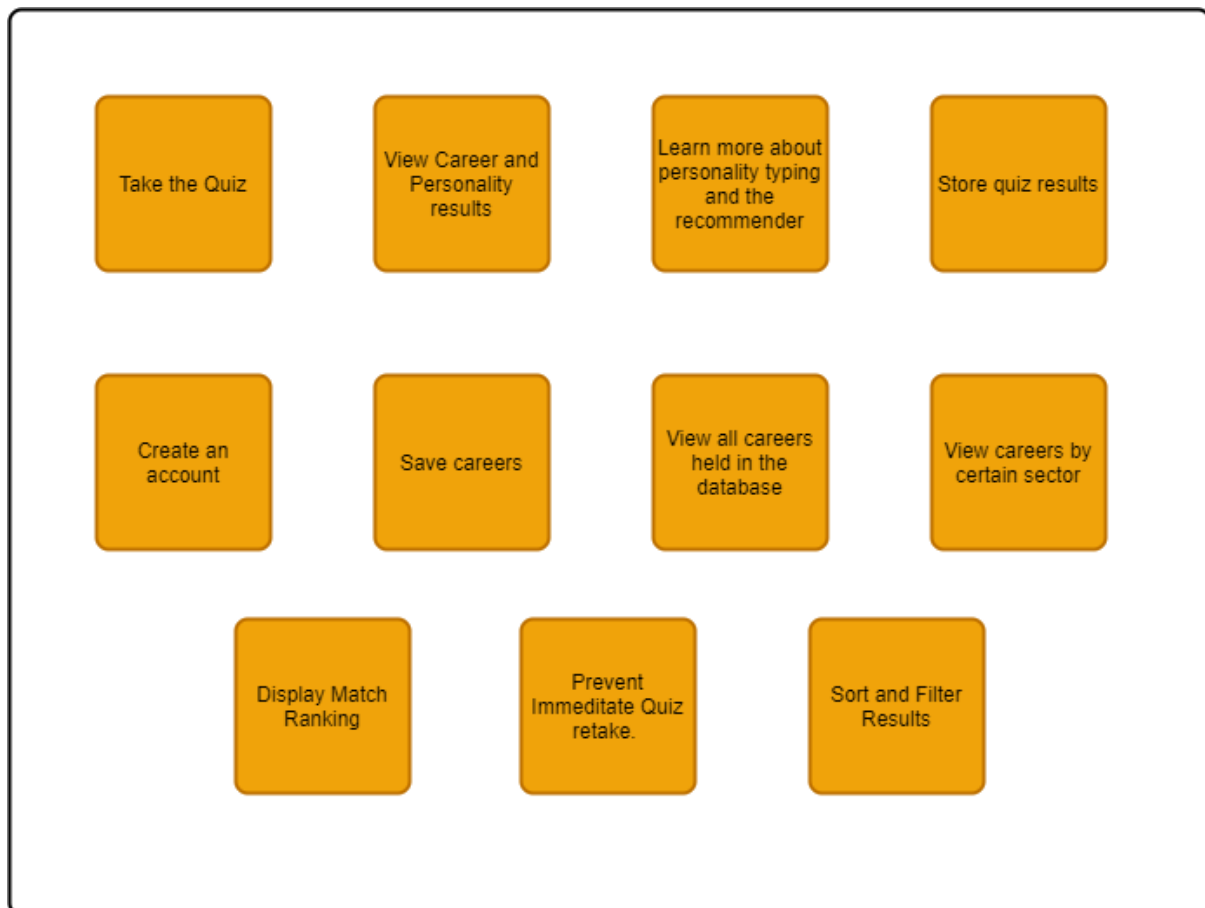


Figure 1.Initial Requirement Story Board

## User Stories

The initial requirements were then converted into user stories that could be used by the developer to produce the final piece of software. This is the process of taking a requirement and breaking it down into a piece of functionality that the user wants to be able to use to achieve some desired result from the system. The user stories for this project all followed the same structure:

*“As a <type of user>, I want <feature>, so that <benefit>”*

User stories are an excellent way of transcribing exactly what you want the user to be able to do on your system, and what value completing this action will create for them. This also helps for an initial analysis of some of the requirements, as if it is difficult to come up with the “why” for a user story, it may not be worth including in the system. User stories are a very popular method of requirement documentation as not only are they easily trackable in acceptance testing, discussed in Section 7.4.0,

but they are also written in plain English, leading to a direct understanding from the development team all the way up to the final customer in one document.

### Prioritisation

The prioritisation process begins first with story pointing which is the process of assigning points to each individual story based on how important it is to the stakeholders, or in this case the project brief. Points should also be increased based on how difficult / how much time the story will take to implement. The next stage was to prioritise the stories using the MoSCoW method, to gauge in which order the stories would be completed. The MoSCoW method splits the stories up into:

- Must Have.
- Should Have.
- Could Have.
- Won't have.

Must have requirements were prioritised based on being a direct match to the project brief and marking criteria. The should have requirements were all functionality that extended the core recommendation system. The could have and wont requirements were extended functionality not relevant to the core of the project, but were included for an extended challenge after the conclusion of the marked project. The final stories table, including prioritisation, can be found at Appendix A.

### Non-Functional Requirements

Non-functional requirements must also be accounted for when drafting the requirements of a system as these are of equal importance. Non-functional requirements are more difficult to gather, as an understanding needs to be developed of what the system is going to do before you can assess what conditions it is going to operate under. The following requirements were gathered:

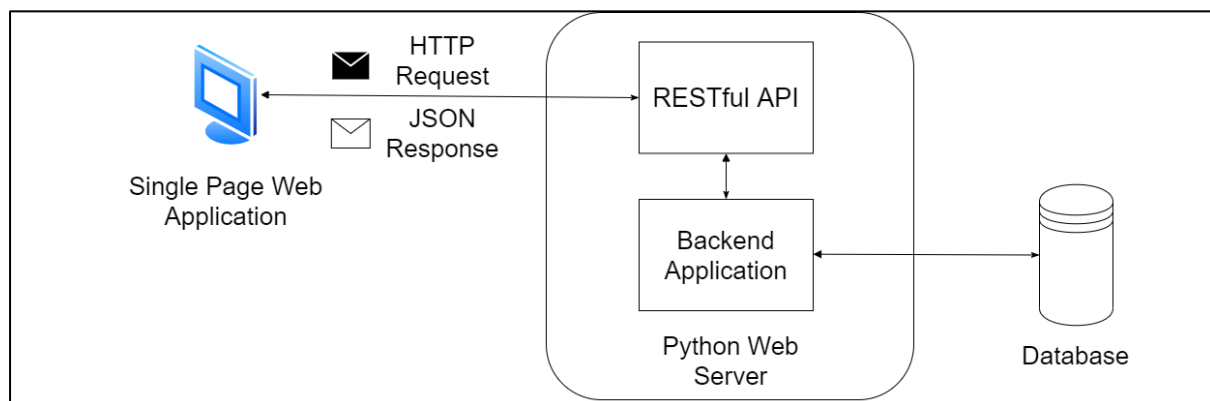
1. Security measures need to be implemented in order protect user account data.
2. Website needs to be compatible on all web browsers including on both mobile and desktop hardware

## Design

Once requirements have been gathered and analysed, an overall design for the system was then produced. To fulfil the needs of a range of age groups, from students just planning their career journey to those later in life looking for a switch, it was decided that a web application would be the preferred approach to implement this system. Most of the population are now familiar with utilising a browser-based system, whereas some may still struggle using mobile app technology. Web applications are also more suited to the classroom environment, with most schools and universities having access to them, allowing for the application to be used during lessons or on the students own time in school.

### System Architecture

With the decision to implement a web application, the complete system architecture was then developed. The application was implemented using a 3 layered system, comprising a single-page web application, restful web service and database, illustrated in Figure 2. The decision to implement a Python environment on the server side was down to the developer's familiarity with the language and technologies, allowing for the best implementation of the code possible.



*Figure 2. Three Layered Web Application*

The main decision for opting to use a single-page over a multi-page application was down to application performance. The use of a single page application will simply re render the page based on the users' actions and only require a connection to the backend service when updates to the stored data are required. As the data loaded in from the web service will be modified often on the site, it was more optimal to use an application that can handle these changes in the web browser, and then only utilise the web server connection when necessary.

Operating using a separate application to handle the business logic was done to follow best practice guidelines for developing web applications. Keeping the business logic side of your application separate from the client on your own server helps to ensure the security and integrity of your entire application, preventing bad actors from applying modification to your app on their own device and gaining unauthorised access to restricted data.

The application programming interface (API) acts as the connector in the system, sending data between the two applications through HTTP requests. The API was chosen to be implemented using the Representational State Transfer (REST) architecture. An API defines the set of protocols in which the client, the application requesting information, must follow to access information from the web service, while also defining the information returned based on that request.<sup>12</sup> REST is an architectural framework for API development, creating guidelines the API must follow to be classed as RESTful.<sup>12</sup> REST was chosen as it provides solid guidelines that the developer must follow in order to develop an effective API, but is still flexible in its implementation and does not enforce technology choices on the developer.

### User Interface Design

Working closely with the constructed use case diagram, shown in Figure 3, and requirements document in Appendix A, a medium fidelity wireframe was created using Axure RP. Axure was chosen as the wireframing tool as it offers the ability to add basic click functionality to demonstrate the full flow of the application, such as drop downs and button navigation. It is important to note that although a single-page application was implemented for this project, the top-level UI components will be referred to as pages in both the design and development for ease of understanding.



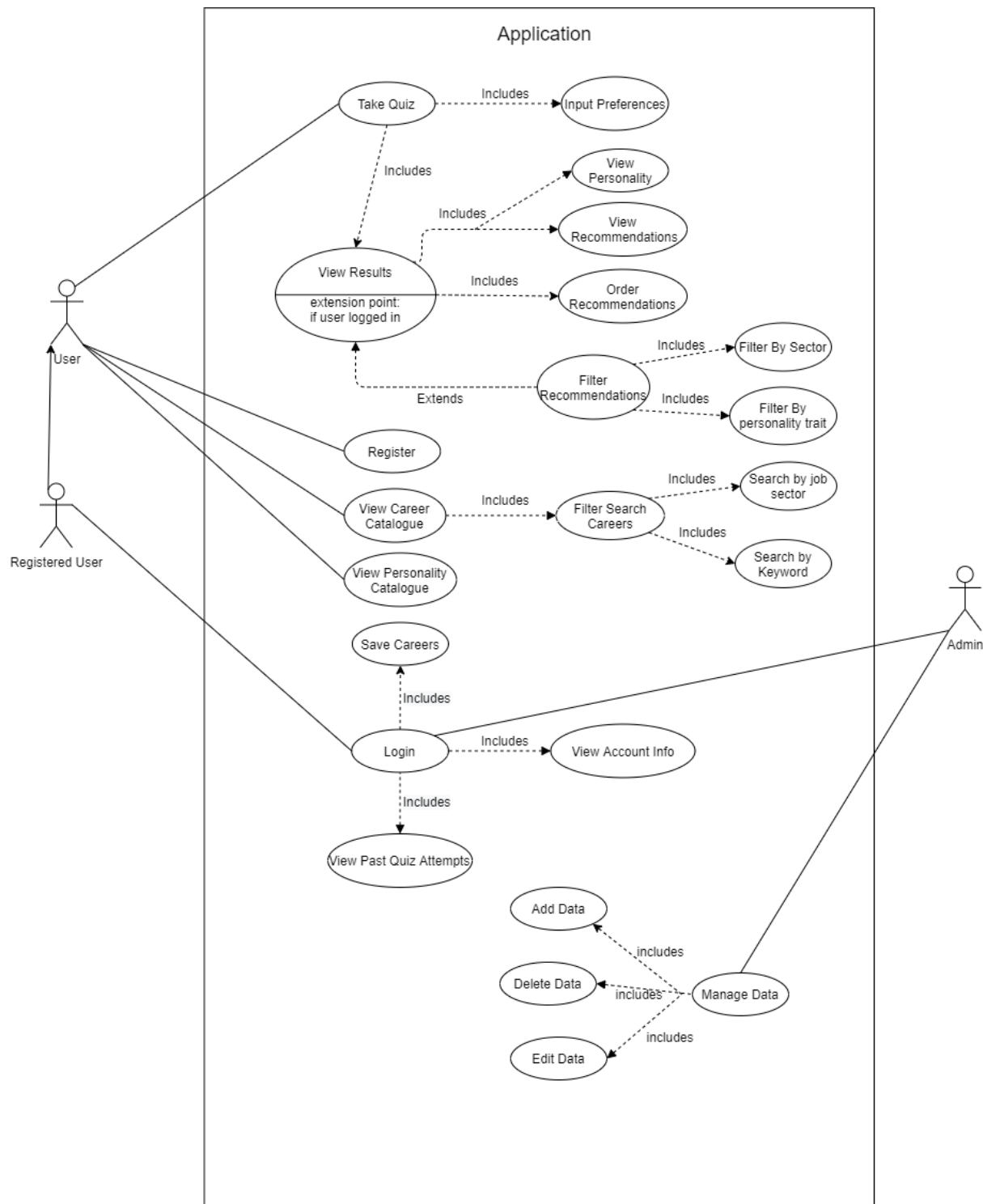


Figure 3. Use Case Diagram

Allowing the user to apply a grading to a question is essential in building up the correct user profile to be used by the recommender. As such, during the design of the quiz screen, story 6 was added to the requirements table to allow it to be validated separately from the quiz itself. Radio buttons were then added to the quiz screen design, shown in Figure 4.

The wireframe shows a quiz interface for 'Career Finder'. At the top, a green header bar contains the text 'Career Finder' on the left, 'View Results' and 'Careers' in the center, and 'Login / Username' on the right. Below the header, on the left side, is a blue link labeled 'Quit Quiz'. The main content area is white and contains the instruction 'Please answer all of the following questions' followed by a green progress bar. Below this, there are two identical example questions, each preceded by the text 'Here is an example question'. Each question has five radio button options: 'Strongly Agree', 'Agree', 'Neutral', 'Disagree', and 'Strongly Disagree'. At the bottom center of the main content area is a blue button labeled 'Next'.

Figure 4. Quiz Screen Wireframe

The major design change when implementing the wireframe came in the results screen. The initial impression would be that the result screen was split into 3 separate columns, shown in Figure 5. This layout would lead to poor downscaling for use on mobile browsers, with the filter section being stacked on top of the results section and pushing it off the screen, breaking requirement 10. Instead, the design was moved into a single column layout, with filters and sorting functions available in drop down menus shown in Figure 6.

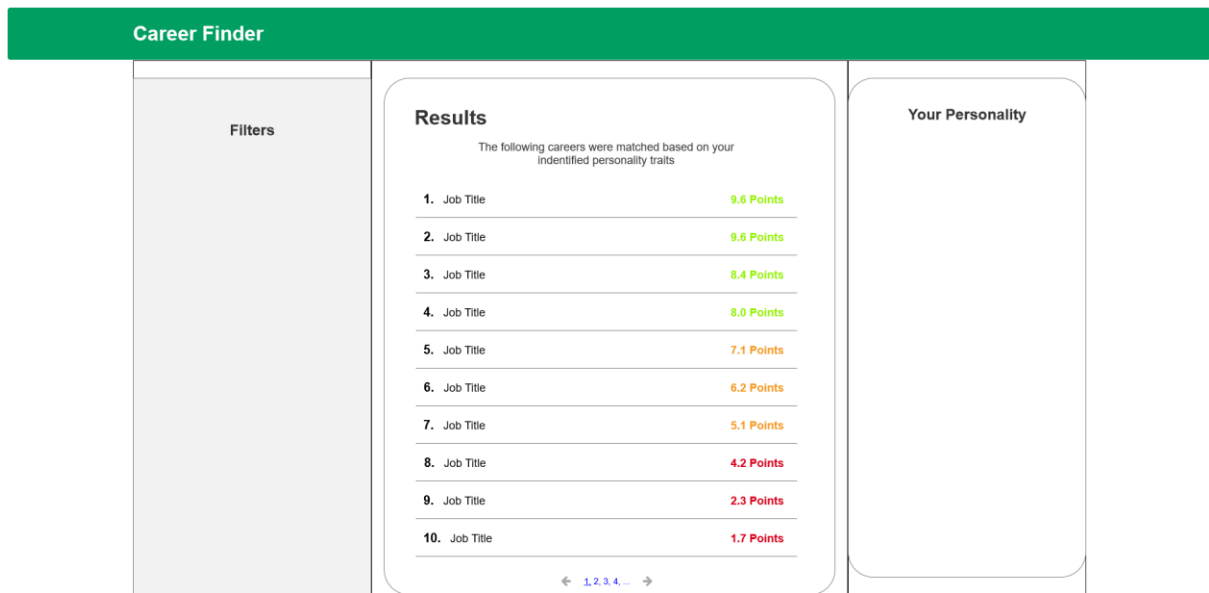


Figure 5. Initial Design for Result Screen

### Your Recommendations

The following careers were matched based on your  
identified personality type  
(please click panels to expand)

Sort By	Filter By
1. Job Title	9.6 Points
2. Job Title	9.6 Points
3. Job Title	8.4 Points
4. Job Title	8.0 Points
5. Job Title	7.1 Points
6. Job Title	6.2 Points
7. Job Title	5.1 Points
8. Job Title	4.2 Points
9. Job Title	2.3 Points
10. Job Title	1.7 Points

1 2 3 4 ...

### Your Personality

The Example  
(ESTP)

Example Description of the personality type.

#### E - Extrovert

Example Extrovert Description

#### S - Sensing

Example Sensing Description

#### T - Thinking

Example Thinking Description

#### P - Perceptive

Example Perceiving Description

Figure 6. Final Result Screen Design

The wireframe was designed to fulfil all the must have and should have requirements that would be implemented during the project lifecycle, the link between each requirement and component of the wireframe can be found in Appendix B.

## Data Design

When deciding on the data design it was important to consider how the data would support the backend recommendation algorithm while also considering how it would be displayed on the front-end UI. For the recommendation algorithm to work correctly, the data relating to users and careers must hold a personality type and the 4 personality component vector values. For the UI to fulfil all the user stories around personality typing and recommendations, the relevant data on personality type and careers must be present to be shown for the user. This adds quite a considerable amount of structure to all the data that would be held in the database for this application and as such, a relational database management system (RDMS) would be more appropriate. With the data in the site also likely to change often as a result of the user taking the quiz, the ACID compliance of relational systems mean that the integrity and accuracy of this data will remain intact as these changes are consistently made. Once the system was chosen, the next stage was to produce an enhanced entity relationship diagram (EER) shown in Figure 7, to illustrate the data and relationships held in the database.

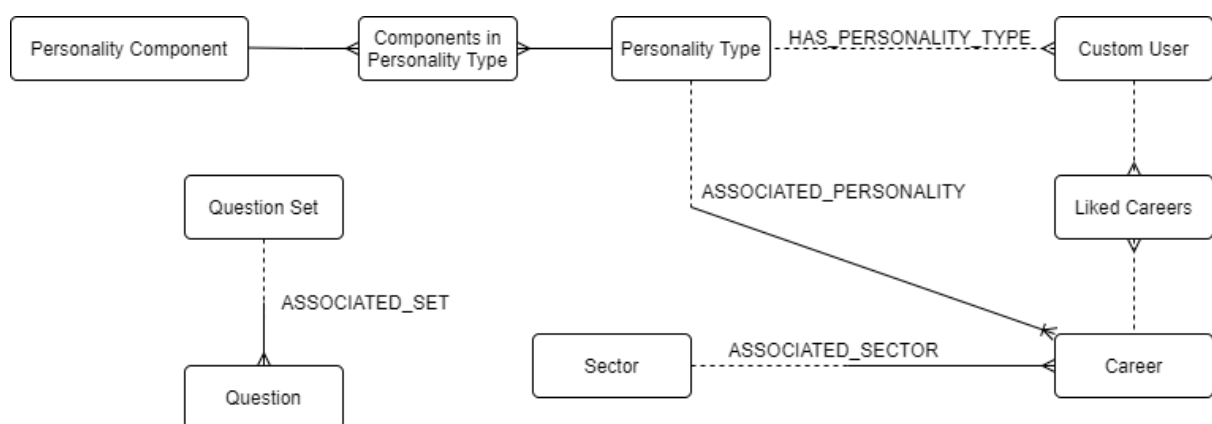


Figure 7. EER Diagram

Questions were included in a set using the “associated\_set” relationship, this meant that the frontend application could load a set of questions once and then filter the array based on the personality factor that was currently in question as opposed to making 4 separate calls for questions on each factor. The “associated\_personality” relationship aided in fulfilling requirement 2, by linking the careers to a particular personality type that could be recommended to users of that same type. The “has\_personality\_type” relationship allowed for the fulfilment of requirement 19, allowing for the saving of a user’s quiz results to the database. Although design principles dictate this relationship should be represented as a new entity, a user is unlikely to create an account without first taking the

test, thus negating the null value for personality type that would be present and preventing the need for a new relationship. “Liked\_careers” addressed requirement 37, allowing for users to have certain careers associated with them. “Associated\_sector” was included for future planning and meant that requirement 15 could be fulfilled later.

The entity relationship model used to produce the EER diagram can be seen in full in Appendix C.

## Low Level Design

### Backend Application

An object relational mapper (ORM) was used to interface with the database. This implies that the data model was constructed in software and then used to build the associated database table structure. ORM allows the developer to interface with the database utilising the chosen programming language instead of standard SQL.<sup>13</sup> ORM avoids the need to change the application in two places, reducing the likelihood of errors during database operations. As the application only involves basic querying, ORM can handle this efficiently. The data are pulled from database and placed into objects that can have create, read, update and delete (CRUD) operations performed on them.

### Application Programming Interface Design

To complete the low-level backend design, API documentation was published using Swagger.io to map out the relationship between the server-side framework and client application, shown in Appendix D. This acted as the design for the Restful API, detailing what requests and corresponding responses would be sent between the two applications. All endpoints relating to stored user data were designed to be protected to fulfil the security non-functional requirements.

The design of the API was done closely following the collected user stories in Appendix A. The “Get All Careers” endpoint was designed to fulfil requirement 23. The “Get Careers by Personality Code” and “Get Careers with Match Score” were both designed to fulfil requirement 2. These were separated into two endpoints as a logged in user can view their associated match score between personality and career whereas a normal user cannot. To enforce security, utilising a user id meant the use of a protected view is necessary and as such a separate end point was created. The need to search careers by personality type is also included in the requirements and this negates the need to create a separate

endpoint to implement that in a future iteration. The “Get Careers by Component Letter Code” endpoint fulfils requirement 25.

Both personality endpoints fulfil requirement 12. The choice to separate these into two endpoints meant that third party applications can be created for this API, and a user can access their data through their own application so long as they have relevant authentication information using the “Get Users Personality Type” endpoint.

The “Get Quiz Questions by Question Set” endpoint fulfilled requirement 1, returning the question dataset to populate the personality quiz. Both the “Post Unlogged User Response” and “Post Logged in User Response” endpoints also address requirement 1, by returning the calculated personality type. The logged in user endpoint posts the users results to the relevant user instance in the database, fulfilling requirement 19. Again, these needed to be separated to allow for logged out users to have their results calculated without the need for authentication credentials.

The login and register endpoints were designed to fulfil requirements 35 and 34 respectively. The creation of these endpoints also meant that all other account-based stories could be implemented into the system in the future.

## Implementation

### Introduction

With the overall architecture and initial designs complete, it was now time to move into the development phase of the project. The first stage of the development process was to make a final decision on the technologies and frameworks that would be used to meet the requirements and match the system architecture laid out in Section 5.1.0.

## Technologies and Frameworks

### React Application

React was chosen as the obvious stand out for the front-end application, splitting the UI into “components” as opposed to individual pages, allowing for the creation of a single-page application. A survey done in 2020, analysing web frameworks used by JavaScript developers, shows that 80% of developers had some experience working with React in their career.<sup>14</sup> Previous development work using flutter had been completed, which follows a very similar development pattern to react, allowing for easy transferal of acquired knowledge into the project.

React Bootstrap, an extension package for React, was used as it predefines common UI components such as buttons and forms that are commonly used in almost all web applications, negating the need to create excess code for these essential components. Bootstrap also operates using a grid system for its layouts, so the developer can manage how much space each component takes up on a screen depending on the size, allowing for effective downscaling for mobile browsers, fulfilling the non-functional requirement.

### Django Web Service

Django and Flask are the two most popular Python web frameworks, with both being considered for this application. While Flask may have been more suitable for the size of the project, Django includes many of the features needed to complete this application out of the box whereas Flask relies heavily on using external packages that extend onto its core framework. Given the relative inexperience of the developer, having this functionality already included by Django, and in one set of documentation, makes it the better choice for this project.

Django automatically creates an administration site that can be used to manage the backend data, allowing for developers and admins to manually change the data stored in the database without the need to create administrative functionality for the application. As the possibility to include this functionality on the front-end would not be achievable, the inclusion of the admin panel allowed for these requirements to be temporally fulfilled until such a time the administrative functionality can be developed on the front-end. Django ORM capabilities also come built in with framework in the form of “Models”. By inheriting from the base model’s class, custom ORM classes were created following the class diagram in Figure 8, to fit the data needs of the application.



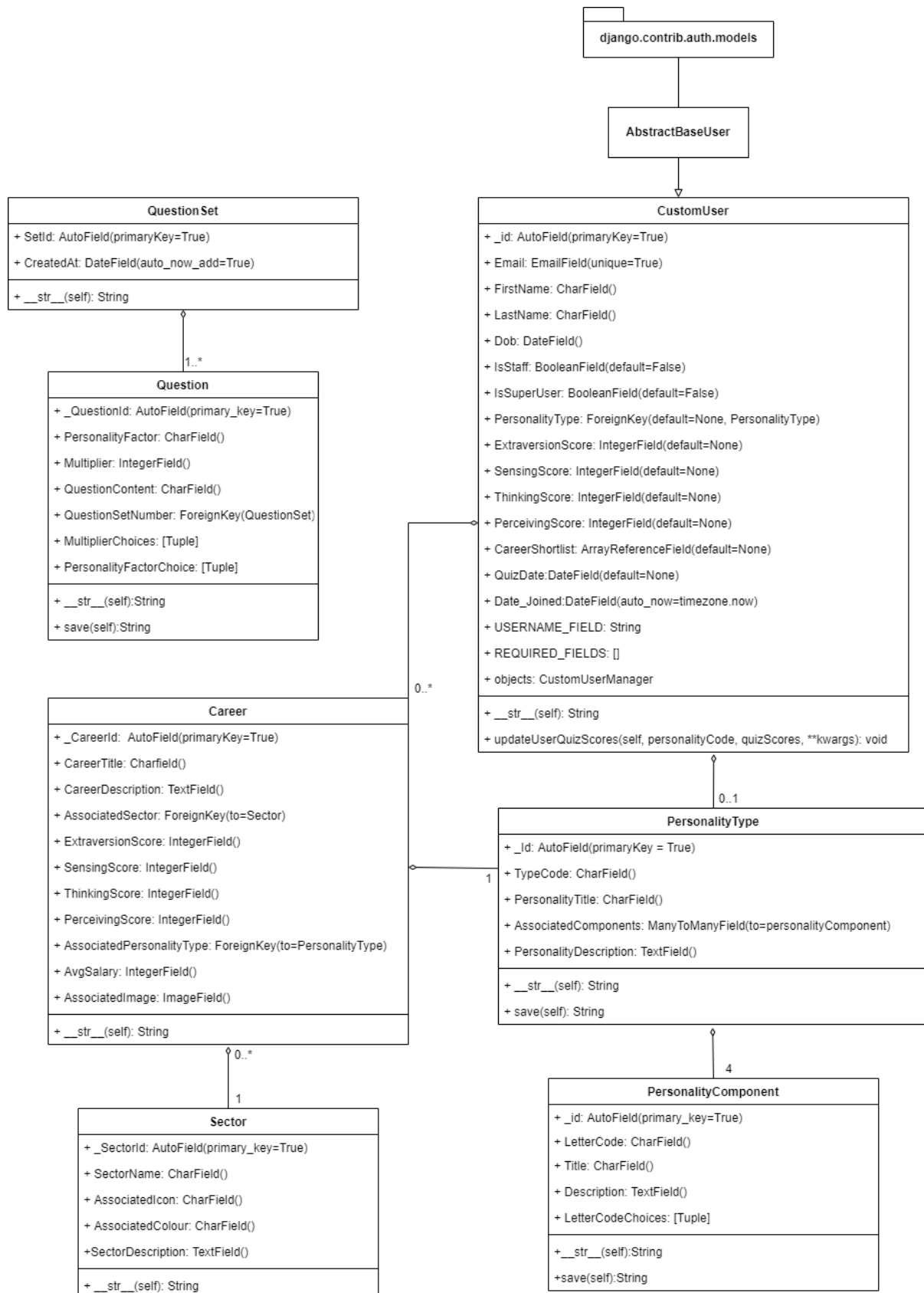


Figure 8. Django UML Class Diagram

To implement the API, the Django Rest Framework was used to streamline the development process and ensure strict adherence to the REST principles. The Rest Framework is an extension of the Django framework and carries all the utilities necessary to create functions that handle and respond to HTTP requests. An extension of the rest framework, Simple JWT, was used to manage user authentication and allow access to protected views and data with the correct credentials, allowing for the fulfilment of the security related non-functional requirement.

## Development

### React Development

To manage the global state of the application, the React-Redux package was used which allowed for the creation of a global data store shown in Figure 9. Each individual dataset in the data is then assigned a reducer. This reducer is responsible for making each dataset accessible to the application components. The reducers utilise switch statements that perform update operations on the dataset depending on which action is dispatched to the global store, then returning the new state to the components. An example of the login reducer can be seen in Figure 10.

```
16  const reducer = combineReducers({
17    questionSet: questionSetReducer,
18    userAnswers: quizAnswerScoreReducer,
19    personalityResults: personalityTypeReducer,
20    user: userPersonalityReducer,
21    registration: userRegisterReducer,
22    recommendedCareers: recommendationReducer,
23    loggedUser: loggedInUserReducer,
24  });
25
```

Figure 9. Global Data Store

```

61 export const loggedInUserReducer = (state = {}, action) => {
62   switch (action.type) {
63     case USER_LOGIN_REQUEST:
64       return { loading: true };
65     case USER_LOGIN_SUCCESS:
66       return { loading: false, userInfo: action.payload };
67     case USER_LOGIN_FAIL:
68       return { loading: false, error: action.payload };
69     case UPDATE_USER_PERSONALITY_TYPE:
70       const updatedUserInfo = {
71         userInfo: {
72           ...state.userInfo,
73           personalityType: action.payload.personalityCode,
74           quizDate: action.payload.quizDate,
75         },
76       };
77       localStorage.setItem(
78         'userInfo',
79         JSON.stringify(updatedUserInfo.userInfo)
80       );
81       return { userInfo: updatedUserInfo.userInfo };
82     case USER_LOGOUT:
83       return {};
84     default:
85       return state;
86   }
87 };

```

Figure 10. Login Reducer

Axios was used to create the actions that are dispatched to the reducers. Axios handles making HTTP requests to the web service, acquiring the necessary dataset needed to be sent to the reducer to perform updates on the store. The actions dispatch constants so that the store knows which reducer to utilise and ensure the correct dataset is updated. An example of the login action can be seen in Figure 11. The “logUserIn” function is a utility function used by both the login and register action to dispatch “USER\_LOGIN\_SUCCESS” and the user information to the store.

```

export const login = (email, password) => async (dispatch) => {
  try {
    dispatch({
      type: USER_LOGIN_REQUEST,
    });

    const config = {
      headers: {
        'Content-type': 'application/json',
      },
    };

    const { data } = await axios.post(
      `/api/user/login/`,
      { email: email, password: password },
      config
    );

    logUserin(data, dispatch);
  } catch (error) {
    dispatch({
      type: USER_LOGIN_FAIL,
      payload:
        error.response && error.response.data.detail
          ? error.response.data.detail
          : error.message,
    });
  }
};

```

Figure 11. Login Action

All components were implemented using functions, utilising hooks to gain access to the data held in the global state. The use of functional components is now the most optimal way to create React applications and follows best practice guidelines. The main consideration when implementing the components was to ensure that the number of components accessing the global store was minimised. Having too many components hooked into the store creates error prone code and could create desync between datasets. To combat this, page components were created to manage the global state that is then passed to the child component for rendering. An example of this can be shown for the quiz screen and quiz component in Figure 12 and Figure 13, whereby the quiz screen passes the question data to the quiz component as a parameter for rendering, while also passing the state update function “setQuestionAnswer” so the parent state can be updated by the child.

```

11     <QuizComponent
12       factorQuestions={factorQuestions}
13       questionAnswers={questionAnswers}
14       setQuestionAnswer={setQuestionAnswer}
15     />

```

Figure 12. Code Snippet of Quiz Screen, passing global data 'factor questions' to child component

```

4
5  function QuizComponent({
6    factorQuestions,
7    questionAnswers,
8    setQuestionAnswer,
9  }) {
10    return (
11      <div>
12        {factorQuestions.map((question, questionNumber) => (
13          <QuestionItem
14            key={question._questionId}
15            question={question}
16            questionNumber={questionNumber}
17            onChange={(e) =>
18              userAnswerChangeHandler(e, questionAnswers, setQuestionAnswer)
19            }
20          ></QuestionItem>
21        ))}
22      </div>
23    );
24  }
25
26  export default QuizComponent;

```

Figure 13. Quiz Component

## Django Development

As mentioned in Section 6.2.2, models were used to implement the ORM functionality. All models inherited from the base model's class except for the 'Custom User' class. A user class needs to be present inside the Django application to implement all the authentication features needed. As the user in this application required more fields than provided by the base user class, for data such as personality type, this model was implemented by inheriting the 'Abstract Base User' class as shown in Figure 14. This negated the need to create a secondary model to manage user data not related to the base user class while still implementing all Django's authentication features. The common function

'updateUserQuizScores' was also created inside the model to follow the don't repeat yourself (DRY) coding principles, as a user's personality type can be updated from multiple points in the application.

```
127 class CustomUser(AbstractBaseUser, PermissionsMixin):
128     _id = models.AutoField(primary_key=True, editable=False)
129     email = models.EmailField(('email address'), unique=True, null=False, blank=False)
130     firstName = models.CharField(max_length=100, null=True, blank=True)
131     lastName = models.CharField(max_length=100, null=True, blank=True)
132     dob = models.DateField(null=True)
133     personalityType = models.ForeignKey(PersonalityType, default=None, on_delete=models.SET_NULL, null=True, blank=True)
134     extraversionScore = models.IntegerField(default=0, blank=True, null=True)
135     sensingScore = models.IntegerField(default=0, blank=True, null=True)
136     thinkingScore = models.IntegerField(default=0, blank=True, null=True)
137     perceivingScore = models.IntegerField(default=0, blank=True, null=True)
138     quizDate = models.DateTimeField(null=True, blank=True, default=None)
139     is_staff = models.BooleanField(default=False, blank=False, null=False)
140     is_superuser = models.BooleanField(default=False, blank=False, null=False)
141     date_joined = models.DateTimeField(auto_now=timezone.now)
142
143
144     USERNAME_FIELD = 'email'
145     REQUIRED_FIELDS = []
146
147     objects = CustomUserManager()
148
149     #Method to update the users quiz information
150     def updateUserQuizScores(self, personalityCode, userAnswers, **kwargs) :
151         personalityType = PersonalityType.objects.get(typeCode=personalityCode)
152         self.personalityType = personalityType
153         self.extraversionScore = int(userAnswers['extraversion'])
154         self.sensingScore = int(userAnswers['sensing'])
155         self.thinkingScore = int(userAnswers['thinking'])
156         self.perceivingScore = int(userAnswers['perceiving'])
157
158         #Add the passed in quiz date to the custom user model if it is datetime object and not greater than the current date.
159         if 'quizDate' in kwargs.keys() :
160             quizDate = kwargs['quizDate']
161
162             quizDate = quizDate.astimezone(timezone.utc)
163
164             if quizDate > datetime.now(tz=timezone.utc) :
165                 raise Exception('Cannot Have quiz date greater than the current date')
166
167             self.quizDate = quizDate
168
169         self.save()
170
171     def __str__(self):
172         return self.email
```

Figure 14. Custom User Model

Functional views were used to implement the API endpoints and handle the fetch and return operations on the data. While it is considered best practice to implement class based views in Django applications, given the small number of views needed for this application and the fact they contain minimal logic, it was unnecessary to add further complexity to the application by utilising class based views. Permission class decorators were utilised to protect the views that implemented access to any custom user classes, requiring a JWT token to be present to access the view. An example of the protected view to post a logged in user's quiz response can be seen in Figure 15. The implementation of the dot product recommendation algorithm can be found in Figure 16.

```

49 @api_view(['Post'])
50 @permission_classes([IsAuthenticated])
51 def registerLoggedUserResponse(request) :
52     data = request.data
53     user=request.user
54
55     try :
56         quizScore = util.unpackQuizDetails(data)
57
58         personalityCode = util.calculatePersonalityCode(quizScore)
59         user.updateUserQuizScores(personalityCode, quizScore, quizDate=datetime.now())
60
61         return Response({'personalityCode': str(user.personalityType), 'quizDate' : user.quizDate})
62
63     except KeyError as ke :
64         message = {'message': 'No key found for ' + str(ke)}
65         return Response(message, status=status.HTTP_400_BAD_REQUEST)
66     except ValueError as ve :
67         message = {'message': 'Incorrect value passed to factor data, error= ' + str(ve)}
68         return Response(message, status=status.HTTP_400_BAD_REQUEST)
69     except Exception as e:
70         message = {'detail' : str(e)}
71         return Response(message, status=status.HTTP_400_BAD_REQUEST)

```

Figure 15. Protected Post User Quiz Response View

```

3 def calculateMatchScore(career, personality) :
4     try :
5
6         matchScore = 0
7
8         matchScore += career['extraversionScore'] * personality['extraversion']
9         matchScore += career['sensingScore'] * personality['sensing']
10        matchScore += career['thinkingScore'] * personality['thinking']
11        matchScore += career['perceivingScore'] * personality['perceiving']
12
13        return matchScore
14
15    except Exception as e:
16        return 'Failed to access ' + str(e)

```

Figure 16. Recommender Algorithm Code

## Project Management

Youtrack, a project management tool created by JetBrains, was used to manage the requirements and track project progression during development. The main feature used in this project was the “Agile Boards”, allowing for full tracking of user story development across each sprint. Once each story was placed in the product backlog, it was then tagged by its priority in the requirement document and was

referenced using story ID numbers. Bugs were also tracked in a similar way, however these were tagged as “Dev bugs” and prioritisation ranged from “Minor” to “Critical”, shown in Figure 17.

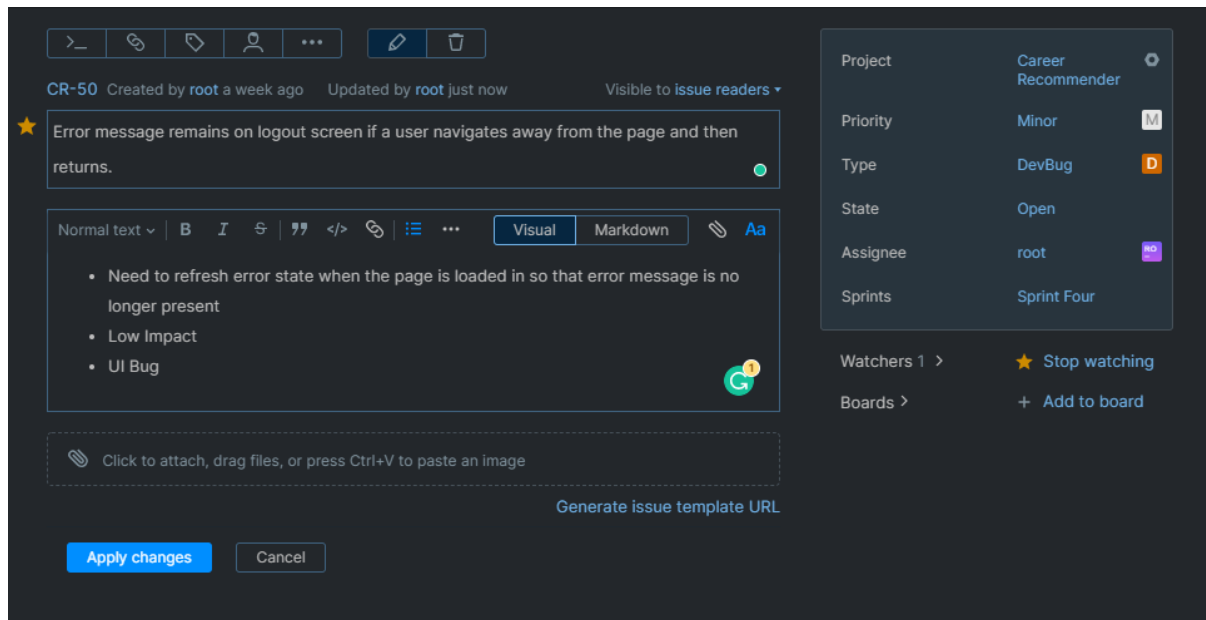


Figure 17. Example Bug Report Youtrack.

From here at the beginning of each sprint, a new board was opened known as the sprint board. The sprint board had four separate sections: Open, In Progress, To Verify, Done. Each story that would be worked on in that sprint was first moved into the open category from the product backlog, from there each individual story was moved into the in-progress section while in development. Once completed, the story was moved into the to verify section. This process was then repeated for each story in the open section until all stories were moved across to be verified. At the end of the sprint, once user acceptance testing had concluded, all stories successfully validated were moved into the done section. The full tracking for sprint one can be seen in Figure 18, Figure 19 and Figure 20. In the case of a failed verification test the story was placed back in the backlog and the associated bug was marked for fixing in the next sprint.



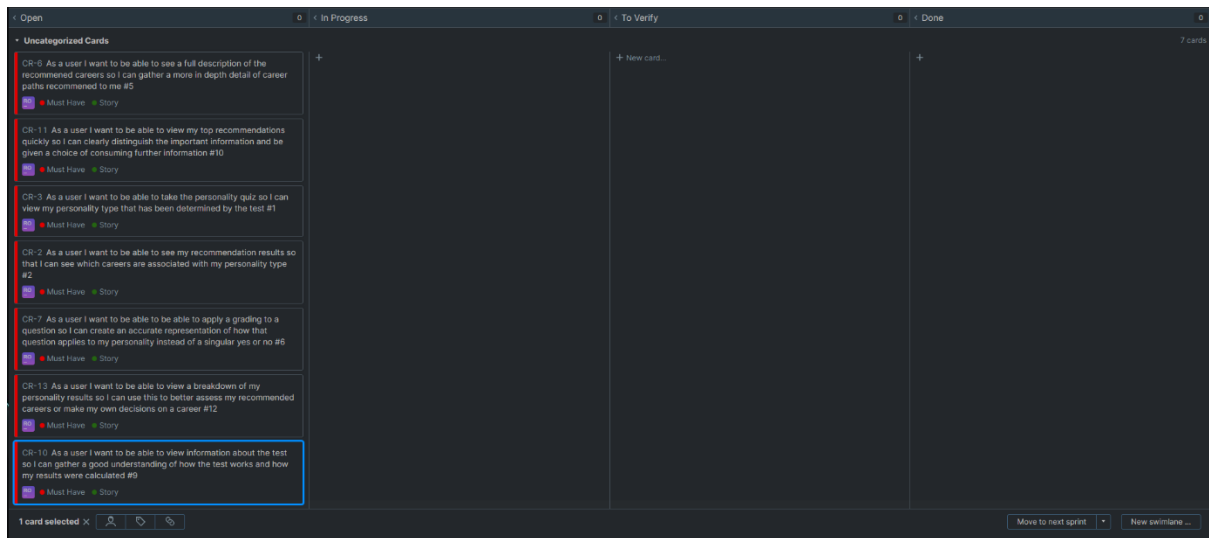


Figure 18. Example Youtrack board sprint begin.

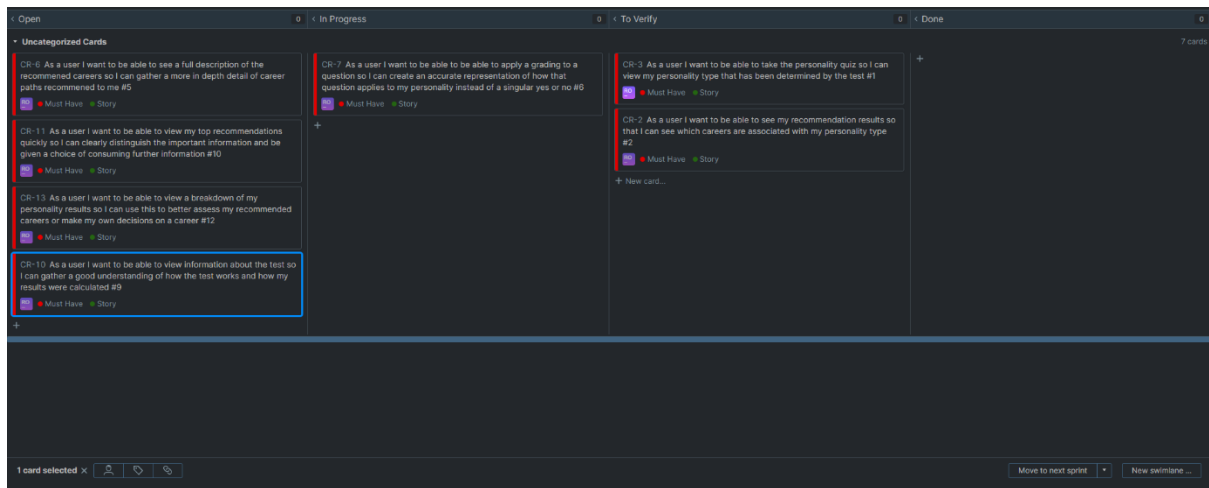


Figure 19. Example Youtrack board sprint in progress.

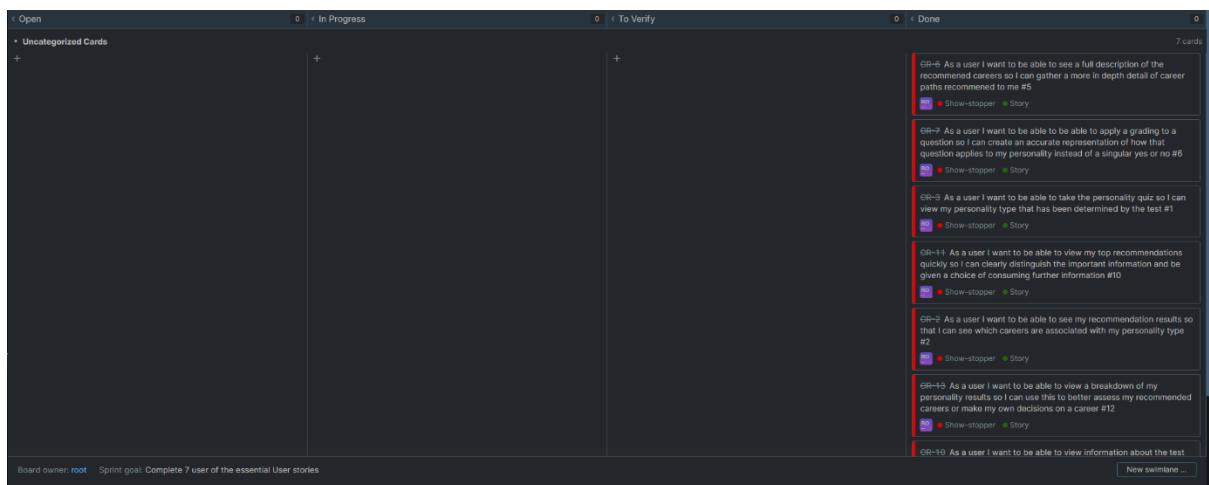


Figure 20. Example Youtrack board sprint end.

## Other Tools

### Version Control

Although usually targeted toward teams working in tandem on a particular codebase, version control is still an important tool that every developer should be familiar with as it is used extensively in industry. Git was the chosen tool to implement version control in this project, as it is the most popular tool used by software teams and can integrate seamlessly with the cloud repository GitHub.

The standard git workflow was reviewed before development began,<sup>15</sup> and implemented throughout the entire process. When functionality was adapted or added, a new branch of the master code was created. This allowed for modifications to be made without affecting the main working codebase and ensuring no system breaking bugs were introduced to the functioning code. Once all the changes were implemented and passed the relevant tests, the code was then committed back to the repository to be merged with the master branch. Once this master branch was then tested again extensively and confirmed to be working, the main codebase was pushed up to the GitHub repository to have the new working version stored off the local machine in case any local data was lost or destroyed. Branching was used extensively in this project, so that a working version of the code was always ready for submission while other aspects were in development.

## Evaluation

The methods used to evaluate each aspect of the project lifecycle are detailed and discussed in this section. The steps outlined were repeated multiple times over each iteration to ensure the quality of the final project output.

### Requirement Analysis

#### INVEST Principles

The first stage of the analysis process was to conduct an initial analysis on the user stories utilising the INVEST principles, to ensure that each story was of satisfactory quality before being included in the story table. The INVEST principles are as follows:

1. Independent.
2. Negotiable.
3. Valuable.
4. Estimable.
5. Small.
6. Testable.<sup>16</sup>

In the initial requirement document, the following story was included:

*“As a user I want to be able to take the test in a concise manner so that I do not lose interest in the quiz and stop using the application.”*

Upon analysis using INVEST, it was discovered that this story did not fit the criteria of being testable. It is difficult to measure a user’s attention span without gathering a large user test pool, as it will be different across multiple people. As this was beyond the scope of this project it was removed from the requirements document.

#### Prioritisation Analysis

The prioritisation of the requirements was reviewed regularly at the end of each sprint based on how the project was progressing. It was apparent at the end of sprint two that the inclusion of the careers catalogue functionality was not going to be possible alongside the recommendation filtering. Building a solid recommendation system was the core focus of the project and any user stories related to that

should be prioritised first, as such filtering remained within the should have priority and the catalogue functionality was moved into the could have requirements.

### Public Requirements Survey

The final stage of requirement analysis was to gain feedback from potential users through the form of a requirements survey, shown in Appendix E. This survey was a technique employed to gain tangible feedback on the stories produced during the requirements gathering phase while also offering the opportunity for potential users to include their own features that they would value in the application. The results of the survey mainly confirmed the prioritisations set by the developer, with most could have requirements returning a negative response and all should have requirements returning a positive. There were some notable exceptions, with a unanimous positive response to requirement 37, this was moved into the should have requirements. Requirements 26 and 38 were also moved into the should have requirements after a positive response rate of 70% from the survey. The inclusion of job vacancies for each career was also suggested by a participant, although beyond the scope of this project it was included in the could have requirements for future consideration.

### Design Analysis

#### Heuristics UI Evaluation

A heuristic evaluation is the process of comparing your user interface design with the 10 core design principles set out by Jakob Nielsen.<sup>17</sup> The design was also assessed using the Web accessibility principles laid out by the World Wide Web Consortium,<sup>18</sup> to ensure that the design is inclusive for those with disabilities. Analysis using both sets of principles was conducted in Appendix F on the final wireframe and any changes were implemented before moving into the development phase. The most notable violation came from issue 4, in which the representation of match strength between user and career was done through colour grading instead of numerically. This meant that those with severe vision impairment utilising a screen reader could not see the full representation of the strength of their match. As such the design was changed and the colour gradient removed to now be represented by a numerical value.

## Database Design Evaluation

The design of the data structure was subject to the process of normalisation. Normalisation evaluates your proposed data structure and attempts to remove the possibility of any redundancy present in the final tables and ensure the integrity of the data. It was discovered that each personality code was repeated multiple times in the original personality type entity, creating repeated redundant data. To correct this, a personality component table was created to hold each of the 8 possible values that can be contained within a personality type. As a personality type can be made of 4 possible values, and each personality component can be linked to many personalities, a new many to many table was constructed in order to handle the relationship between personality type and component, moving the structure into second normal form.

## Prototyping

It was identified during design that certain aspects of a particular career, such as required licenses or level of education, may be relevant to some jobs and not to others. To preserve the structure of the data and include these new columns, null values would have to be included in a relational table which can damage the data integrity. As such a document approach may be more appropriate when storing career data in the application and prototyping was carried out to compare the two possible approaches before beginning development. Two prototypes were created using both a relational and non-relational database structure. Both were implemented using the chosen technologies, with MongoDB as the database for the non-relational and SQLite for the relational.

The production of the SQL prototype was nearly twice as fast the document prototype due to the developer's familiarity with relational systems. The main concern however came due to recent compatibility issues between Django, the Django MongoDB system, and Django's administration system that were introduced with the latest Django version. This would therefore prevent access to the Django admin panel, a key factor in the decision to make use of a Django server-side framework. For the reasons outlined, it was decided that an SQL system would be adopted. Although the compatibility issue with MongoDB could have been easily solved by implementing another document database or server-side framework, this however would have drastically hindered development time. By this point the developer was already familiar with Django and MongoDB, taking the time to learn a new system would impact what could be included in the final project. As SQL was still a completely valid solution and only career data needed to be unstructured, the benefits of a polyglot approach would not outweigh the possible negative impact on the development of the minimum viable product.

## Verification Testing

Verification testing is the process of evaluating the low-level code that is written to ensure that it functions in the way the developer intended. It focuses on uncovering any bugs that are present in the system to increase the overall quality of the code produced and prevent errors. The main forms of verification testing used in this project were Unit and Integration.

## Unit Testing

Unit tests were developed for both the front and backend applications using the recommended methods by the framework developers. This meant for the React application both Jest and the React Testing Library were used, with UnitTest being used for the Python backend. This allowed for the complete automation of unit testing for the application, building a large and extensive test suite for the entire system.

The main purpose of the backend application was to fetch and return, so many functions and classes had set outputs with minimal logic that could be easily tested in isolation, while also being included in the regressions tests for the entire system. Regression testing is the process of running all unit tests present in a related suite, or for smaller projects like this one the entire system, to ensure no bugs are present in other units after the introduction of new code. This was especially important in the backend application, as the “View” units that make up the API largely contain many different units from across the application.

Unit testing the React application followed the guidelines of the creator of the React Testing Library, that state the overall render of the component should be tested as opposed to the individual implementations.<sup>19</sup> This meant focusing on what was produced by the component on render rather than the internal workings. The unit tests also focused on ensuring that the correct hooks were called, this was achieved using mocking in Jest. In the case where complex logic was implemented in the component that required its own tests, this functionality was abstracted out into a utility package so that it could be tested in isolation.

## Integration Testing

Given the short timescale of the project and the developers limited initial knowledge of the testing libraries used, creating an automated integration test suit would have taken a significant amount of time and impacted the overall result of development. Given the small amount of integration tests needed for the system, this was done manually by the developer using various tools.

As returning data through the API incorporates all the individual units contained within the backend application, testing these endpoints returned the desired result was the primary method of integration testing for the web service. To achieve this, the popular development tool Postman was used. Postman allows for the user to make HTTP requests to API endpoints and then displays the JSON response on the UI. Upon completion of the unit tests for a piece of functionality, the related endpoints that utilise this code were then tested and the results from a request in Postman were then compared with the API documentation that held the expected JSON returned by that endpoint. Tests using incorrect parameters were also carried out to ensure that each endpoint correctly handled a bad request without crashing the system. An example of a valid and invalid API call using Postman can be seen in Figure 21 and Figure 22

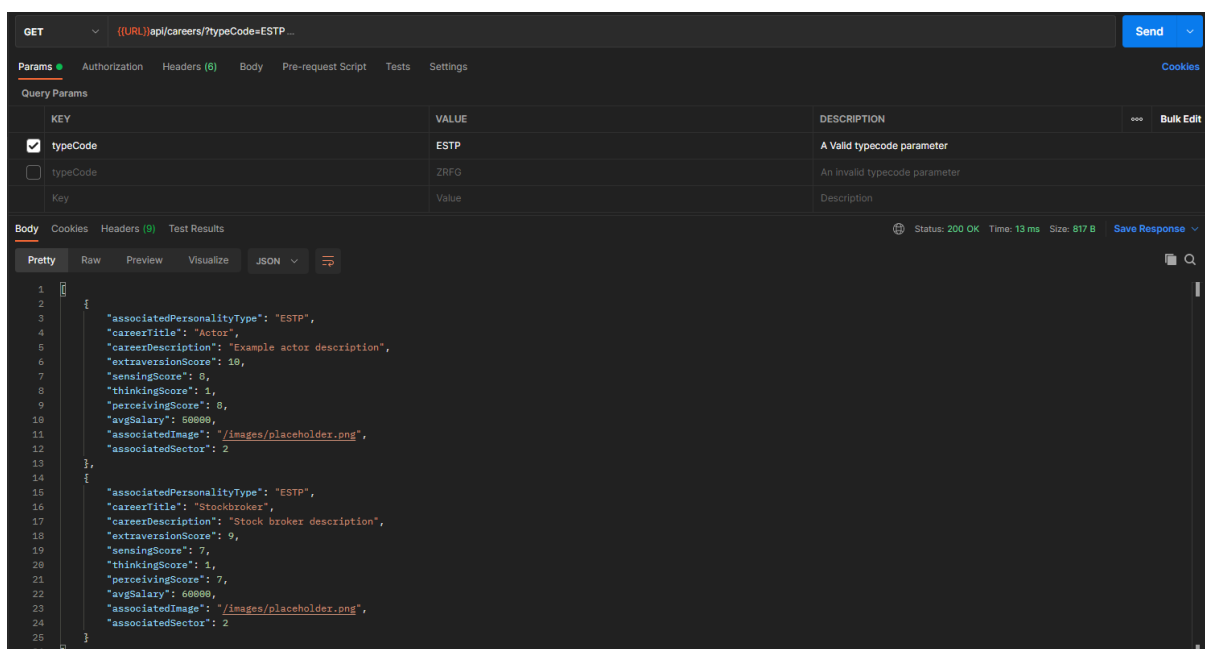


Figure 21. Valid API Response Captured in Postman

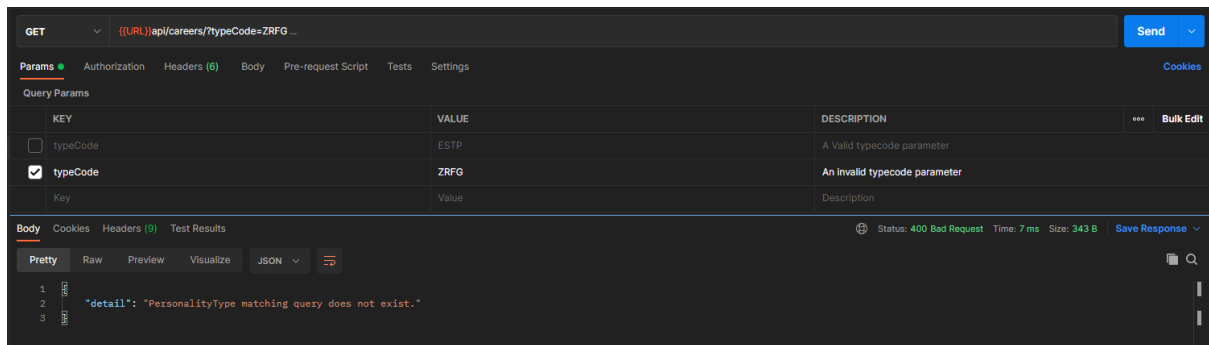


Figure 22. Invalid API Response Captured in Postman

Integration testing on the frontend was performed using Redux Dev Tools, a chrome browser extension that allows the user to monitor the global state present in an application. Unit testing covered the rendering of components using mock data, so the focus of the integration testing was on the applications state management. Actions were completed on the UI by the developer following the user manual and the dev tools were used to monitor the state updates fired by the application as well as the current global state at each point. Utilising this information, it was easy to monitor the state transfer across the system and ensure the relevant data was present at component render. Common functions such as redirection, navigation and refresh were also completed to measure how the state responded when these actions were applied. An example of the dev tool testing can be found in Figure 23.

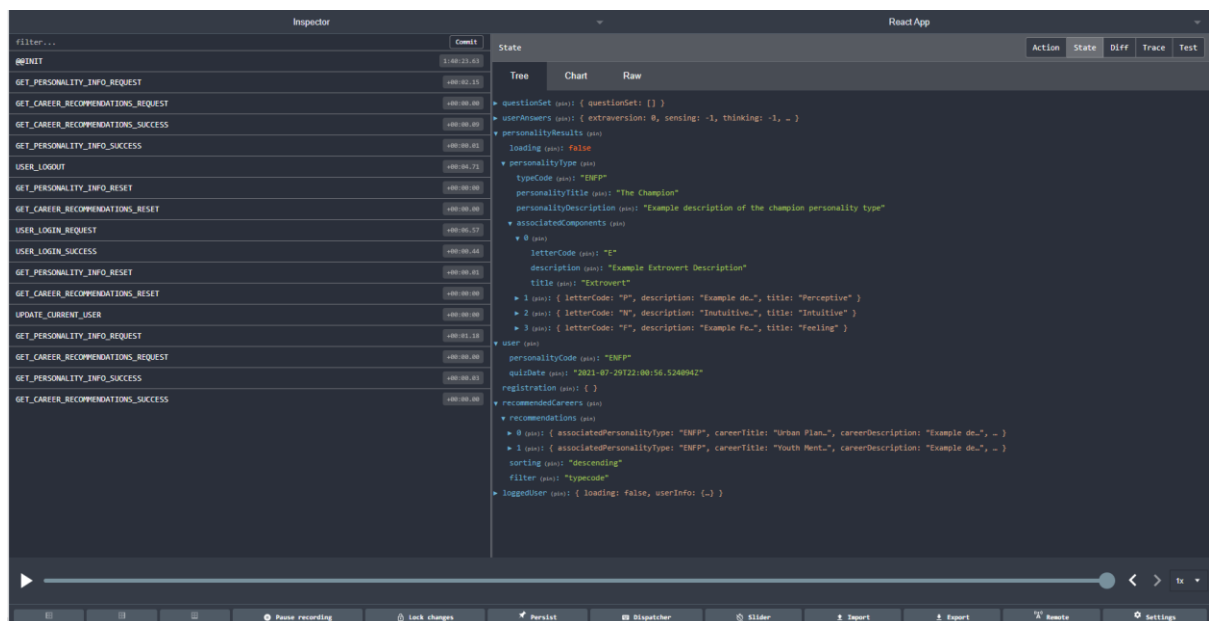


Figure 23. Redux Dev Tool State Monitoring (right) and Action Monitoring (left)



## User Acceptance Testing

User Acceptance testing is the process of ensuring that each piece of functionality is validated against expected requirements by external stakeholders, proving that the software has been built for the purpose it was intended. As this project does not have any stakeholders other than the developer, a tester with no technical coding knowledge was brought in for this task.

Acceptance testing sessions took place at the end of each sprint, so that the functionality produced in that sprint could be validated and defined as done or so any errors can be recorded and amended in the next sprint. The developer's assessment sheet captured other metrics such as time taken to complete the task and number of user mistakes. As the user interface is supposed to be a minimalist design and simple to use, any drastic differences in time or mistakes being made indicate an overall problem with the design. In Session one the tester pointed out they only knew the branding on the navigation bar was clickable as they had recently done a lot of work that involved using their companies web site. As such a backward navigation button was created on the result screen to take the user back to the home page.

The second section in the acceptance testing offered the tester the opportunity to provide feedback on the overall visual design of the system, based on the heuristic principles and their own opinion. In session two the test user points out that more colours should be added to the personality type information to create an association between colour and personality factor for the user, as such each factor was colour coded on the UI.

A full list of all the tests completed can be found in Appendix G. A mapping table between story and acceptance tests was created to monitor which stories had been implemented and validated in the system and can be found in Appendix H. Test 3 of session 2 was the only acceptance test not to pass on the first attempt. This bug was subsequently fixed and the test was conducted again in session 3 test 6, to validate the stories inclusion in the system.

## Conclusion

The overall aim of this project was to build a recommender system that matched a person's personality to careers that would be suitable for them. This was achieved through creating a three-layered web application consisting of a single-page web application, RESTful web service and SQL database. The project lifecycle was managed using an Agile methodology to effectively respond to any changes, especially as developer skill level increased, and to become familiar with working under standard web development best practices. All aspects of the lifecycle, from requirements to design, were tested and analysed extensively using known techniques and principles to increase the overall quality of the final code produced, while also ensuring that the final application matched both the project description and the developer's initial vision for the system. To implement MBTI successfully this requires administration from a trained practitioner with the instrument, however using the personality models as a guide allowed for the successful creation of a recommender system that utilises content filtering to recommend careers to users based on their answers to the personality test.

## Further Work

The MVP submitted for this project contained all the functionality relevant to the project brief detailed in Section 2.0.0. The first step of further work would be to include all the remaining should have requirements, along with fixing known minor errors in the application. The next stage would be to migrate the database from the SQLite instance inside the Django application and move it to a deployment ready Postgres instance on a separate server, leaving the application ready for deployment on the web. The discussion around polyglot data structures was considered in Section 7.2.3. With the introduction of preference tags in the should have requirements, and careers keyword search functionality coming in later releases, moving to a document structure for the career data would be the most optimal solution. This could be introduced in a post release version of the application and be rigorously tested to ensure it keeps the same level of performance as the SQL structure before beginning to work on the careers catalogue functionality and the rest of the could have requirements.

## Bibliography

---

- <sup>1</sup> K. Thompson, "What Percentage of Your Life Will You Spend at Work?", *ReviseSociology*, 2016. Available: <https://revisesociology.com/2016/08/16/percentage-life-work/>.
- <sup>2</sup> E. Herr, "Career Development and Mental Health", *Journal of Career Development*, vol. 16, no. 1, pp. 5-18, 1989. Available: 10.1177/089484538901600102.
- <sup>3</sup> I. Briggs- Mayers, M. McCaulley, N. Quenk and A. Hammer, *A guide to the Development and Use of the Myers-Briggs Type Indicator*, 3rd ed. USA: Consulting Psychologists Press, 1998.
- <sup>4</sup> "The history of the MBTI® assessment", *Eu.themyersbriggs.com*. Available: <https://eu.themyersbriggs.com/en/tools/MBTI/Myers-Briggs-history>.
- <sup>5</sup> C. Jung, *Psychological Types*, 1st ed. London: Taylor and Francis, 2016.
- <sup>6</sup> "MBTI® personality types", *Eu.themyersbriggs.com*. Available: <https://eu.themyersbriggs.com/en/tools/MBTI/MBTI-personality-Types>.
- <sup>7</sup> D. Pittenger, "The Utility of the Myers-Briggs Type Indicator", *Review of Educational Research*, vol. 63, no. 4, pp. 467-488, 1993. Available: 10.3102/00346543063004467
- <sup>8</sup> R. Wagner and R. Weigand, *Measuring Results of MBTI® Type Training*, 1st ed. 2003, pp. 2-4.
- <sup>9</sup> R. McCrae and O. John, "An Introduction to the Five-Factor Model and Its Applications", *Journal of Personality*, vol. 60, no. 2, pp. 175-215, 1992. Available: 10.1111/j.1467-6494.1992.tb00970.x.
- <sup>10</sup> A. Furnham, "The big five versus the big four: the relationship between the Myers-Briggs Type Indicator (MBTI) and NEO-PI five factor model of personality", *Personality and Individual Differences*, vol. 21, no. 2, pp. 303-307.
- <sup>11</sup> R. Damian, M. Spengler, A. Sutu and B. Roberts, "Sixteen Going on Sixty-Six: A Longitudinal Study of Personality Stability and Change across 50 Years", 2018.
- <sup>12</sup> "rest-apis", *ibm.com*, 2021. Available: <https://www.ibm.com/cloud/learn/rest-apis>.
- <sup>13</sup> M. Hoyos, "What is an ORM and Why You Should Use it", *Medium*, 2018. Available: <https://blog.bitsrc.io/what-is-an-orm-and-why-you-should-use-it-b2b6f75f5e2a>.
- <sup>14</sup> "State of JS 2020: Front-end Frameworks", *The State of Javascript*, 2021. Available: <https://2020.stateofjs.com/en-US/technologies/front-end-frameworks/>.
- <sup>15</sup> S. Chacon and B. Straub, *Pro Git*, 2nd ed. New York: Apress, 2014.
- <sup>16</sup> "What does INVEST Stand For?", *Agile Alliance* |, 2021. Available: <https://www.agilealliance.org/glossary/invest/>.
- <sup>17</sup> Nielsen, "10 Usability Heuristics for User Interface Design", *Nielsen Norman Group*, 1994.
- <sup>18</sup> "Accessibility Principles", *Web Accessibility Initiative (WAI)*, 2019. Available: <https://www.w3.org/WAI/fundamentals/accessibility-principles/>.
- <sup>19</sup> K. Dodds, "Testing Implementation Details", *Kentcdodds.com*, 2021. Available: <https://kentcdodds.com/blog/testing-implementation-details>.

## Appendix

### Appendix A – Prioritised User Story Table

ID_NO	Story	Tag	Points	Sprint Number
1	As a user I want to be able to take the personality quiz so I can view my personality type that has been determined by the test	Must Have, UI, Backend	70	1
2	As a user I want to be able to see my recommendation results so that I can see which careers are associated with my personality type	Must Have, UI	10	1
3	As a user I want to be able to see a representation of the strength of each of my matches between user personality and job so I can assess which ones are best	Must Have, UI,	40	2
5	As a user I want to be able to see a full description of the recommended careers so I can gather a more in-depth detail of career paths recommended to me	Must Have, Ui, Backend	10	1
6	As a user I want to be able to be able to apply a grading to a question so I can create an accurate representation of how that question applies to my personality instead of a singular yes or no	Must Have, UI,	20	1
9	As a user I want to be able to view information about the test so I can gather a good understanding of how the test works and how my results were calculated	Must Have, UI	10	1
10	As a user I want to be able to view my top recommendations quickly so I can clearly distinguish the important information and be given a choice of consuming further information	Must Have, UI Design	10	1
12	As a user I want to be able to view a breakdown of my personality results so I can use this to better assess my recommended careers or make my own decisions on a career	Must Have, UI, Backend	30	1
16	As a user I want to be able to order my matches that are closer/further away from my personality so that I can see the strongest or the weakest matches	Should Have, UI, Backend	50	3
17	As a user I want to be able to retake the test after some time has passed so I can keep my results in line with how my personality has changed and matured over time	Should Have, UI	20	3
19	As a user I want to be able to store my results to the personality quiz and close my browser so I can return at any time and view the results and use them in further analysis of choosing a career	Should Have, Backend	60	2
20	As a user I want to be informed on the risk of taking the test again too soon so I can make an informed decision about whether I wish to risk harming the results	Should Have, UI, Backend	20	3
34	As a user I want to be able to create an account so that my own data is associated with me	Should Have, UI, Backend	30	2
35	As a user I want to be able to log into my account so that I can access the data that is held for me on the website and make use of it in my career search	Should Have, UI, Backend	30	2
37	As a user I want to be able to save careers I have an interest in so I can create a shortlist of my favourite potential careers and do further research on them	Should Have, UI, Backend	30	

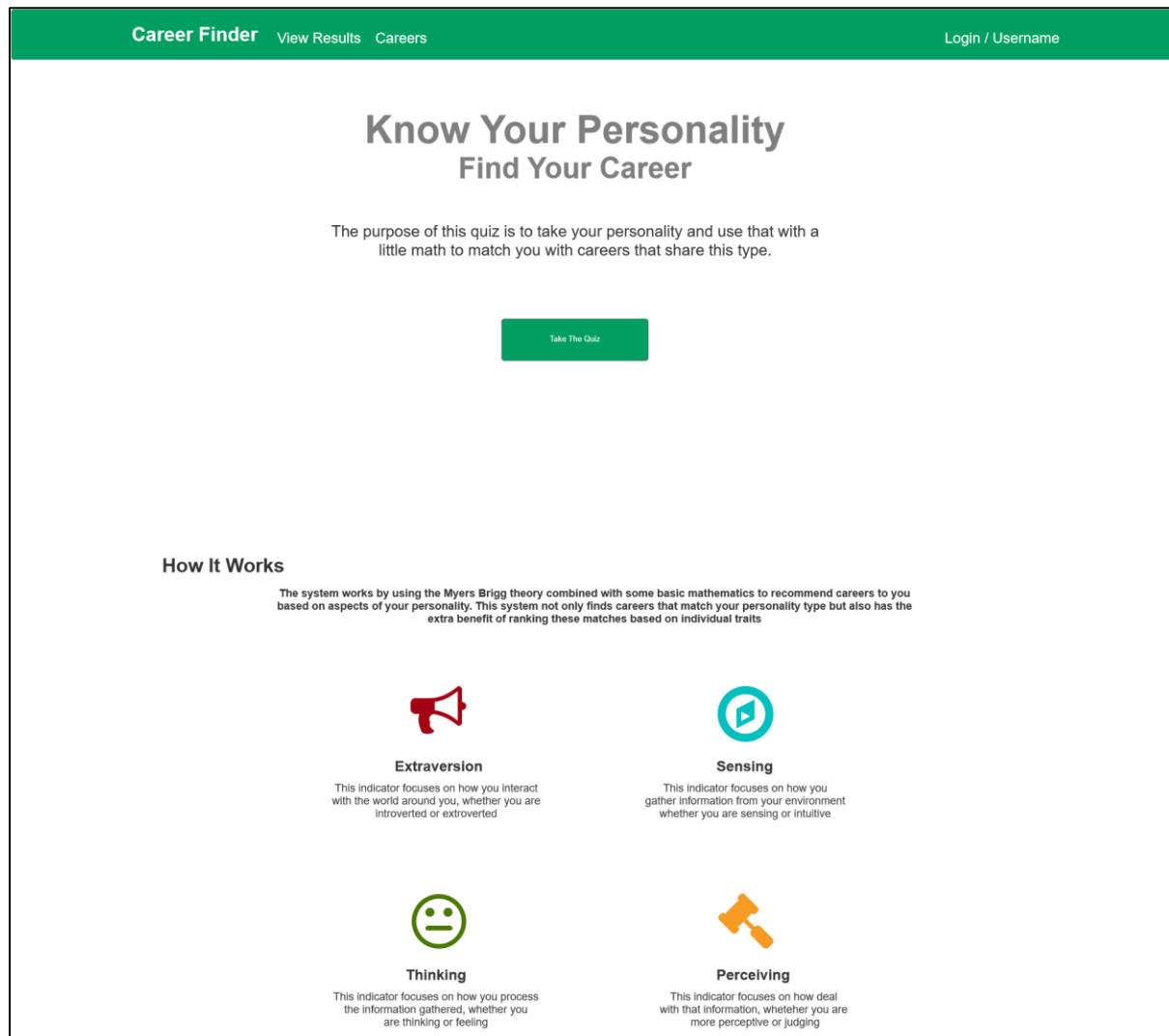
40	As a user I want to be able to view my account information so that I can access saved data about myself and modify it if need be	Should Have, UI Backend	40	
39	As a user I want to be able to add my quiz results to my account without taking the test again If I have previously taken the test before registering	Should Have, UI, Backend	20	3
11	As a user I want to be able to navigate away from the page and come back later to view my results without creating an account so I can use the information to carry out my own searches on the web	Should Have, UI	30	2
13	As a user I want to be prevented from immediately changing my answers after the test has been completed so I can be prevented from changing my true results to something I would prefer	Should Have, Backend, I	10	3
25	As a user I want to be able to filter results based on certain matching personality traits so I can see careers based around the traits I feel are my strongest	Should Have UI, Backend	80	3
38	As a user I want to be able to list my preferences I have surrounding issues like working environment, conditions etc so that I can filter careers that match any of these preferences	Should Have, UI, Backend	70	
26	As a user I want to be able to view my progress throughout the quiz so I remain engaged with the task at hand	Should Have, UI	10	
32	As an admin I want to log in so that I can access my credentials and take actions on the web service	Could Have, UI, Backend	20	
33	As an admin I want to be prompted when entering personality values into a career so that I do not enter incorrect values for the personality traits and cause problems with the recommendations	Could Have, UI	10	
29	As a user I want to be able to see all the personality types available on the website so I can learn more about the different personalities that aren't my own	Could Have, UI, Backend	40	
15	As a user I want to be able to clearly distinguish between different Job sectors so I can explore opportunities that are relevant to each other	Could Have, UI Design	10	
23	As a user I want to be able to view the catalogue of careers held in the database so I can inspire my own ideas on a potential career	Could Have, UI, Backend	40	
24	As a user I want to be able to search the job catalogue by keyword so I can find jobs I am directly interested in	Could Have, UI, backend	50	
27	As a user I want to be able to categorise jobs by industry so that I can view career matches I have a strong initial interest in	Could Have, UI, Backend	30	
36	As a user I want to be able to view my results from previous quizzes so that I can use the information to see how my personality has progressed over time and how I have changed as an individual	Could Have, UI, Backend	30	
30	As an admin I want to be able to add necessary data to the website database from the UI so I can quickly and easily add data to the web service with limited technical knowledge	Could Have, UI, Backend	40	
31	As an admin I want to be able to manage the database from the UI so I can quickly make any changes to the database without the need to change any code	Could Have, UI, Backend	60	

<b>41</b>	As a user I want to be able to navigate to a relevant job vacancy platform so I can search for vacancies in positions related to my recommendations.	Could Have, UI, Backend	80	
<b>22</b>	As a user I want to be able to view a graph displaying all my matches and their strengths so I can make a mathematical comparison between recommendations	Won't Have UI, Backend	80	
<b>18</b>	As a user I want to be able to view my answers before submission so that I can correct any miss clicks or change my answer if need be	Won't Have, UI	10	

*Appendix A. User Story Table.*

## Appendix B – User Interface Wireframe

### B.1 – Home Screen and Navigation Bar

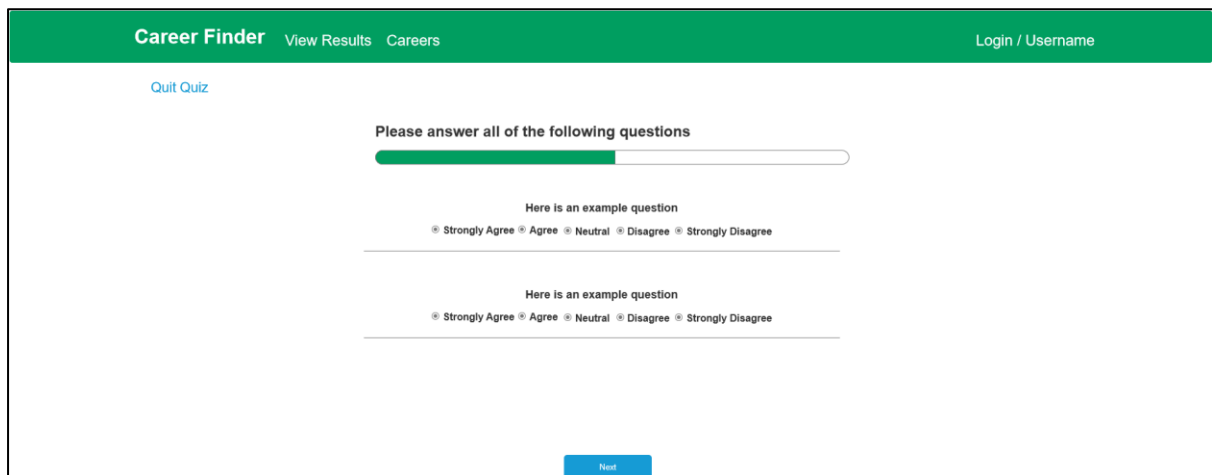


Take the test through the 'Take The Quiz' button, addressing requirement 1.

Extra information is displayed through the 'How It Works' Section, addressing requirement 9.

User can navigate straight to the results screen if they have stored results without needing to take the test again by clicking 'View Results', addressing requirements 11 and 19.

## B.2 – Quiz Screen



The screenshot shows a web interface for a 'Career Finder' quiz. At the top, a green navigation bar contains the text 'Career Finder', 'View Results', 'Careers', and 'Login / Username'. Below the navigation bar, on the left, is a link 'Quit Quiz'. The main content area is white and contains the instruction 'Please answer all of the following questions' followed by a progress bar that is approximately one-third full. Below the progress bar, there are two example questions, each consisting of the text 'Here is an example question' and a horizontal line of five radio buttons labeled 'Strongly Agree', 'Agree', 'Neutral', 'Disagree', and 'Strongly Disagree'. At the bottom center of the main content area is a blue button labeled 'Next'.

Grading from 'Strongly Agree' to 'Strongly Disagree' can be applied through the five radio buttons, addressing requirement 6.

Completing the quiz finalises requirement 1.



## B.3 – Result Screen

Career FinderView ResultsCareersLogin / Username

Your Recommendations

The following careers were matched based on your identified personality type  
(please click panels to expand)

Sort By

Filter By

1. Job Title	9.6 Points
2. Job Title	9.6 Points
3. Job Title	8.4 Points
4. Job Title	8.0 Points
5. Job Title	7.1 Points
6. Job Title	6.2 Points
7. Job Title	5.1 Points
8. Job Title	4.2 Points
9. Job Title	2.3 Points
10. Job Title	1.7 Points

← 1 2 3 4 ... →

Your Personality

The Example  
(ESTP)

Example Description of the personality type.

E - Extrovert

Example Extrovert Description

S - Sensing

Example Sensing Description

T - Thinking

Example Thinking Description

P - Perceptive

Example Perceiving Description

User can view personality type results and career recommendations, addressing requirements 1 and 2.

User can learn more about their recommendation by expanding the list item, addressing requirement 5.

User can quickly view their top recommendations first as they are at the head of the screen, addressing requirement 10.

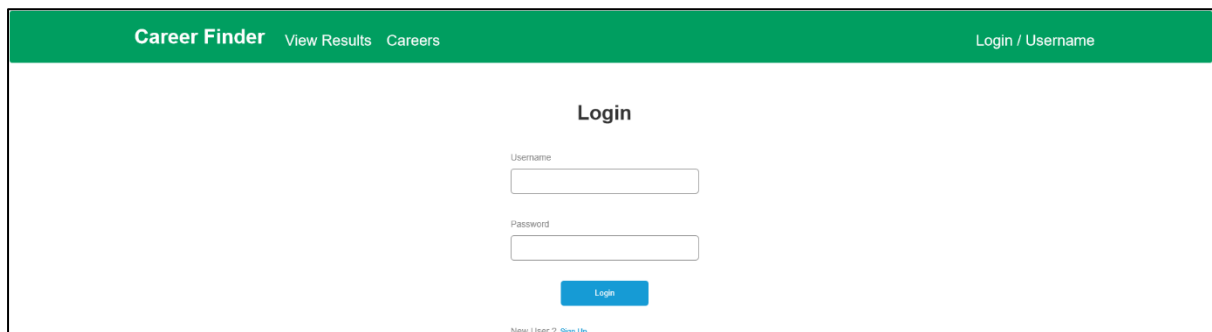
User can view more information on their personality type underneath the recommendations to use in a further study, addressing requirement 12.

The strength of the match is represented as a numerical match next to the career title, addressing requirement 3.

Recommendations can be sorted using the 'Sort By' dropdown menu, addressing requirement 16.

Recommendations can be filtered using the 'Filter By' menu, addressing requirement 25.

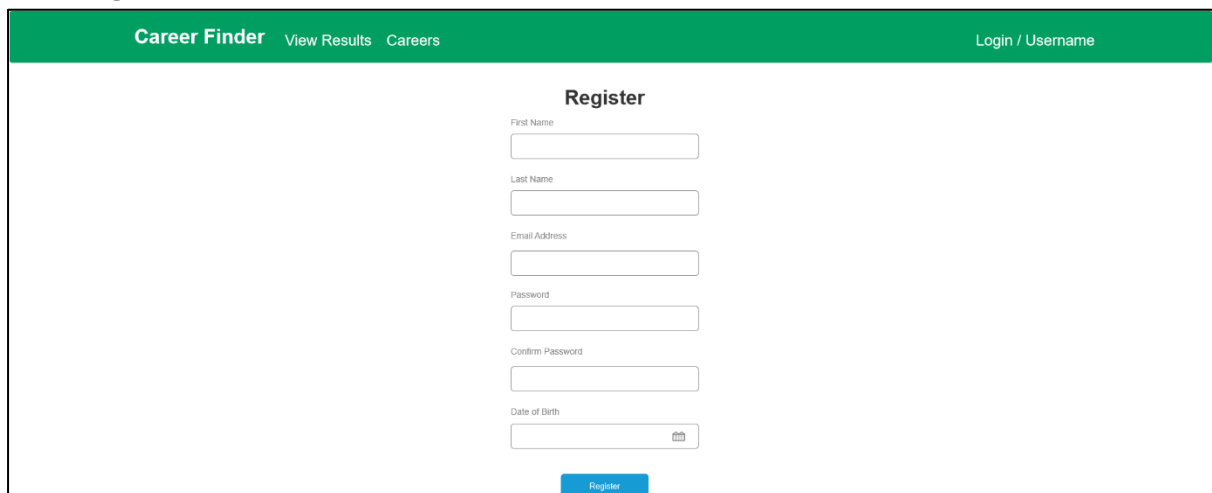
## B.4 – Login Screen



The Login screen features a green header bar with the text "Career Finder" on the left, "View Results" and "Careers" in the center, and "Login / Username" on the right. The main content area is white and contains the title "Login" centered at the top. Below the title are two input fields: "Username" and "Password". A blue "Login" button is positioned below the password field. At the bottom left, there is a link that reads "New User ? Sign Up".

User can login by filling in their details on this screen, addressing requirement 35.

## B.5 – Register Screen



The Register screen features a green header bar with the text "Career Finder" on the left, "View Results" and "Careers" in the center, and "Login / Username" on the right. The main content area is white and contains the title "Register" centered at the top. Below the title are six input fields: "First Name", "Last Name", "Email Address", "Password", "Confirm Password", and "Date of Birth". A blue "Register" button is positioned below the "Date of Birth" field. The "Date of Birth" field includes a calendar icon on the right side.

User can register by filling out the registration details on this screen, addressing requirement 34.

*Appendix B. User Interface Wireframe*

## Appendix C – Entity Relationship Model

### C.1 – Entities and Attributes

- QuestionSet                SetId, CreatedAt
- Question                 QuestionId, PersonalityFactor, Multiplier, QuestionContent, QuestionSetNumber
- PersonalityComponent        ComponentId, LetterCode, Description, Title
- PersonalityType            TypeId, TypeCode, PersonalityTitle, PersonalityDescription, AssociatedComponents
- Sector                    SectorId, SectorName, SectorDescription, AssociatedIcon, AssociatedColour
- Career                    CareerId, CareerTitle, CareerDescription, AssociatedSector, ExtraversionScore, SensingScore, ThinkingScore, PerceivingScore, AssociatedPersonalityType, AvgSalary, AssociatedImage
- CustomUser            UserId, Email, FirstName, LastName, Dob, PersonalityType, ExtraversionScore, SensingScore, ThinkingScore, PerceivingScore, QuizDate, IsStaff, IsSuperuser, DateJoined

### C.2 Entity Relationship Table

Relationship Name	Entities	Degree	Optionality
<b>ASSOCIATED_SET</b>	Question, Question Set	1:M	Obligatory on Many
<b>COMPONENTS_IN_PERSONALITY_TYPE</b>	Personality Component, Personality Type	N:M	Obligatory on Both
<b>ASSOCIATED_SECTOR</b>	Sector, Career	1:M	Obligatory on Many
<b>HAS_PERSONALITY_TYPE</b>	Custom User, Personality Type	1:M	Obligatory on Neither
<b>LIKED_CAREERS</b>	CustomUser, Career	N:M	Obligatory on Neither
<b>ASSOCIATED_PERSONALITY</b>	Career, Personality Type	1:M	Obligatory on Many

# Career Recommender API

Version: 1.0.0.

The backend API for the career recommender system

## Career

### Get Careers By Component Letter Code

Returns all the matching careers that have a personality type containing the passed in personality component letter.

GET

```
api/careers/component/{letterCode}
```

#### Usage and SDK Samples

- Curl
- Java
- Android
- Obj-C
- JavaScript
- C#
- PHP
- Perl
- Python

```
curl -X GET \
-H "Accept: application/json" \
"/api/careers/component/{letterCode}"
```

#### Parameters

Path parameters

Name	Description
letterCode*	String <i>The personality component to retrieve recommendations for.</i> Required

## Responses

**Status: 200 - Successfully returned matching careers for that personality component.**

### Schema

```
▼ [ {}
  ▼ { {}
    associatedPersonalityType: string
    careerTitle:                string
    careerDescription:           string
    extraversionScore:           integer
    sensingScore:                integer
    thinkingScore:               integer
    perceivingScore:             integer
    avgSalary:                   integer
    associatedImage:              string
    associatedSector:             integer
  }
]
```

**Status: 400 - Invalid request**

### Schema

```
▼ { {}
  message: string
}
```

## Get All Careers

Get all the available careers held in the database.

GET

api/careers/

### Usage and SDK Samples

Curl   Java   Android   Obj-C   JavaScript   C#   PHP   Perl   Python

```
curl -X GET\
-H "Accept: application/json"\
"/api/careers/"
```

Responses

Status: 200 - Successfully returned careers.

Schema

▼ [ {} ]

▼ { {} }

associatedPersonalityType: string

careerTitle: string

careerDescription: string

extraversionScore: integer

sensingScore: integer

thinkingScore: integer

perceivingScore: integer

avgSalary: integer

associatedImage: string

associatedSector: integer

}

]

Status: 400 - Invalid request

Schema

▼ { {} }

message: string

}

Get Careers By Personality Code

Returns all the matching career objects for the personality code.

GET

api/careers/personalityCode=?

Usage and SDK Samples

CurlJavaAndroidObj-CJavaScriptC#PHPPerlPython

curl -X GET\  
-H "Accept: application/json"\  
"/api/careers/personalityCode=??personalityCode="

Parameters

Query parameters

Name	Description
personalityCode	String The personality code to get recommendations for.

Responses

Status: 200 - Successfully returned matching careers for that personality code.

Schema

▼ [ {} ]

▼ { {} }

associatedPersonalityType: string

careerTitle: string

careerDescription: string

extraversionScore: integer

sensingScore: integer

thinkingScore: integer

perceivingScore: integer

avgSalary: integer

associatedImage: string

associatedSector: integer

Status: 400 - Invalid request

Schema

▼ { {} }

message: string

Get Careers with Match Score

Returns all the matching careers for the users personality type, along with the strength of the match represented by an integer.

GET

api/careers/points/

Usage and SDK Samples

CurlJavaAndroidObj-CJavaScriptC#PHPPerlPython

curl -X GET\  
-H "Authorization: Bearer [[accessToken]]"\  
-H "Accept: application/json"\  
"/api/careers/points/"

Parameters

Header parameters

Name	Description
Authorisation	String (JWT) The users access token.

Responses

Status: 200 - Successfully returned matching careers for that user.

Schema

▼ [ {}

▼ { {}

associatedPersonalityType: string

careerTitle: string

careerDescription: string

extraversionScore: integer

sensingScore: integer

thinkingScore: integer

perceivingScore: integer

avgSalary: integer

associatedImage: string

associatedSector: integer

matchScore: integer

}

]

Status: 400 - Invalid request

Schema

▼ { {}

message: string

}

Status: 401 - Failed Authorisation

Schema

▼ { {}

message: string

}

---

---



# Personality

## Get Personality by Personality Code

Gets the personality type information for the given code.

GET

```
api/personalities/{personalityCode}
```

### Usage and SDK Samples

- Curl
- Java
- Android
- Obj-C
- JavaScript
- C#
- PHP
- Perl
- Python

```
curl -X GET\
-H "Accept: application/json"\
"/api/personalities/{personalityCode}"
```

### Parameters

Path parameters

Name	Description
personalityCode*	String The personality code of the personality type to be retrieved. Required

### Responses

Status: 200 - Successful return of the personality.

Schema

▼ { }

typeCode: string

personalityTitle: string

personalityDescription: string

associatedComponents: ▼ [ {}  
    letterCode: string  
    description: string  
    title: string  
  ]

}

Status: 400 - Invalid request

Schema

▼ { {} message:  
  string  
}

## Get Users Personality Type

Gets the personality type of the user whose authentication credentials are passed in.

GET

```
api/personalities/user/
```

### Usage and SDK Samples

Curl   Java   Android   Obj-C   JavaScript   C#   PHP   Perl   Python

```
curl -X GET\  
-H "Authorization: Bearer [[accessToken]]"\  
-H "Accept: application/json"\  
"/api/personalities/user/"
```

### Parameters

Header parameters

Name	Description
Authorisation	String (JWT) <i>The users access token.</i>

### Responses

**Status: 200 - Successful return of the personality**

Schema

```
▼ {   
  typeCode:      string  
  personalityTitle:  string  
  personalityDescription: string  
  associatedComponents: ▼ [   
    ▼ {   
      letterCode: string  
      description: string  
      title:      string  
    }  
  ]  
}
```

**Status: 400 - Invalid request**

Schema

```
▼ {   
  message: string  
}
```

Schema

▼ { }

message: string

}

# Quiz

## Get Quiz Questions by Question Set

Gets the quiz questions belonging to a certain question set.

GET

api/quiz/questions/{setNo}

### Usage and SDK Samples

CurlJavaAndroidObj-CJavaScriptC#PHPPerlPython

curl -X GET\  
-H "Accept: application/json"\  
"/api/quiz/questions/{setNo}"

### Parameters

Path parameters

Name	Description
setNo*	Integer <i>The question set to be returned.</i> Required

## Responses

**Status: 200 - Successful return of the quiz question set.**

### Schema

```
▼ [ ]
  ▼ { }
    _questionId: integer
    personalityFactor: string
    multiplier: integer
    questionContent: string
    questionSetNumber: integer
  }
]
```

**Status: 400 - Invalid request**

### Schema

```
▼ { }
  message: string
}
```

## Post Unlogged User Response

Post the users response to the personality quiz, returns the personality typecode and the time the quiz was taken.

### POST

api/quiz/responses/

### Usage and SDK Samples

Curl   Java   Android   Obj-C   JavaScript   C#   PHP   Perl   Python

```
curl -X POST\
-H "Accept: application/json"\
-H "Content-Type: application/json"\
"/api/quiz/responses/"
```

## Parameters

### Body parameters

Name	Description
body *	<div>▼ { }</div> <div>extraversion:integer</div> <div>sensing: integer</div> <div>thinking: integer</div> <div>perceiving: integer</div> <div>}</div>

### Responses

**Status: 200 - Successfully calculated results, return personality code and quiz date.**

#### Schema

▼ { }

personalityCode: string

quizDate: string

}

**Status: 400 - Invalid request**

#### Schema

▼ { }

message: string

}

## Post Logged In User Response

Post the users response to the personality quiz, stores the 4 personality components and personality type in the database along with the time the test was taken for that user. Returns the personality typecode and the time the quiz was taken.

### POST

api/quiz/responses/user

### Usage and SDK Samples

Curl   Java   Android   Obj-C   JavaScript   C#   PHP   Perl   Python

```
curl -X POST\
-H "Authorization: Bearer [[accessToken]]"\
-H "Accept: application/json"\
-H "Content-Type: application/json"\
"/api/quiz/responses/user"
```

## Parameters

### Header parameters

Name	Description
Authorisation	String (JWT) <i>The users access token.</i>

### Body parameters

Name	Description
body *	<div>▼ { }</div> <div>extraversion: integer</div> <div>sensing: integer</div> <div>thinking: integer</div> <div>perceiving: integer</div> <div>userId: integer</div> <div>}</div>

## Responses

**Status: 200 - Successfully calculated results, return personality code and quiz date.**

### Schema

▼ { }

personalityCode: string

quizDate: string

}

**Status: 400 - Invalid request**

### Schema

▼ { }

message: string

}

# User

## Post User Login Details

User login endpoint. Returns user information and access tokens.

POST

```
api/user/login/
```

### Usage and SDK Samples

Curl   Java   Android   Obj-C   JavaScript   C#   PHP   Perl   Python

```
curl -X POST\
-H "Accept: application/json"\
-H "Content-Type: application/json"\
"/api/user/login/"
```

### Parameters

Body parameters

Name	Description
body	<div>▼ { }</div> <div>email: string</div> <div>password: string</div> <div>}</div>

## Responses

**Status: 200 - Successfully login and return of user info. Personality type and quiz date will be null if user has not completed the quiz.**

### Schema

```
▼ { }
  refresh:      string
  access:       string
  _id:         integer
  email:       string
  firstName:   string
  lastName:    string
  dob:         string
  personalityType: string
  quizDate:    string
  token:       string
}
```

**Status: 400 - Invalid request or failed login**

### Schema

```
▼ { }
  message: string
}
```

## Post User Register Details

User Register endpoint. Returns user information and access tokens. User answers are optional, only if the user has completed the quiz and has valid answer data.

### POST

```
api/user/register/
```

## Usage and SDK Samples

Curl   Java   Android   Obj-C   JavaScript   C#   PHP   Perl   Python

```
curl -X POST\
-H "Accept: application/json"\
-H "Content-Type: application/json"\
"/api/user/register/"
```



Parameters

Body parameters

Name	Description
body	<div><div>▼ { }</div><div><div>email:</div><div>string</div></div><div><div>firstName:</div><div>string</div></div><div><div>lastName:</div><div>string</div></div><div><div>dob:</div><div><div>▼ { }</div><div>string</div><div>Must be a valid date string</div></div></div><div><div>password:</div><div>string</div></div><div><div>personalityCode:</div><div>string</div></div><div><div>userAnswers:</div><div><div>▼ { }</div><div><div>extraversion:integer</div><div>sensing: integer</div><div>thinking: integer</div><div>perceiving: integer</div></div><div>}</div></div></div></div>

## Responses

**Status: 200 - Successful registration and user info and token returned. Quiz Date and Personality type will only be returned if user registers with valid quiz results.**

### Schema

```
▼ {  
  _id: integer  
  email: string  
  firstName: string  
  lastName: string  
  dob: string  
  personalityType: string  
  quizDate: string  
  token: string  
}
```

**Status: 400 - Invalid request or failed registration.**

### Schema

```
▼ {  
  message: string  
}
```

# Career Recommender Engine

---

## Start of Block: Default Question Block

The purpose of this project is to build a career recommendation engine and website that allows users to complete a questionnaire and have careers recommended to them based on their determined personality type and traits. The purpose of this research is to validate the requirements gathered during the design of the system by gaining an outside opinion on what features potential users would value in a career recommendation system. The results of the research will then be used to influence the design and implementation of the website. The content of the survey consists of mainly multiple-choice questions with the opportunity for the participant to include their own thoughts and ideas about the system at the end. The survey is open to anyone over 18 regardless of career status so long as they have consideration for their future career choices.

This Study has been created and organised by Thomas Cotton as part of the MSc Software Development program summer project and is supervised by William Bell. Any queries regarding the study can be forwarded to (thomas.cotton.2020@uni.strath.ac.uk). Should you have any further questions about the study and wish to consult an independent party out with the research organiser, you can contact the CIS departments ethics committee at (ethics@cis.strath.ac.uk). This study has been approved by the University of Strathclyde ethics committee.

*Data will be collected and captured through Qualtrics and stored with Strathclyde University. This study does not require the participant to disclose any personal or identifiable information and will ensure anonymity. Should the participant choose or accidentally disclose any identifiable data this will be erased from the data set. All data collected will be removed from the study on the date of October 31st, 2021 and will be solely processed for the purpose of this project only. Upon completion of the survey all anonymous data cannot be removed once collected before the given date.* By choosing to continue I confirm that I have read and understood the above information and my initial queries regarding the survey have been answered in a satisfactory manner and I consent to participation in this project. I understand my participation in this survey is completely voluntary and I am free to withdraw at any time before completion without reason or consequence. I confirm my agreement to the data policies, and I consent to my data being gathered for the purpose of this project only.

☐ I Consent (1)

☐ I do Not Consent (2)

## Recommendation and Quiz Section

---

When completing recommendation quizzes online like this one, how important is it that the quiz can be completed in a timely manner?

- ☐ Crucial (1)
  - ☐ Important (2)
  - ☐ Neutral (3)
  - ☐ Unimportant (4)
  - ☐ Insignificant (5)
- 

If you selected Crucial or Important for the previous question, for topics as significant as career choice, would a lengthier more in-depth quiz still deter you from completing the quiz?

- ☐ Yes (1)
  - ☐ No (2)
-

How important is it to you to be able to see a numerical representation of the strength of a match between your personality type and career?

- ☐ Crucial (1)
  - ☐ Important (2)
  - ☐ Neutral (3)
  - ☐ Unimportant (4)
  - ☐ Insignificant (5)
- 

As a user how important is it for the system to present roadblocks to you to prevent you from retaking the quiz to soon and preserve recommendation integrity if you are not satisfied with your initial results?

- ☐ Crucial (1)
  - ☐ Important (2)
  - ☐ Neutral (3)
  - ☐ Unimportant (4)
  - ☐ Insignificant (5)
- 

If you selected crucial or important for the previous question, should these roadblocks be enforced and prevent you accessing the quiz or come as reminders the test should not be taken again?

- ☐ Reminders (1)
  - ☐ Forced Blocks (2)
-

How important is it to you to be able to filter career recommendation results based on a particular part of your personality as opposed to your personality as a whole?

- ☐ Crucial (1)
  - ☐ Important (2)
  - ☐ Neutral (3)
  - ☐ Not Important (4)
  - ☐ Insignificant (5)
- 

As a user how important is it to be able to track how you are progressing with the quiz for you to remain engaged and answering to the best of your ability?

- ☐ Crucial (1)
  - ☐ Important (2)
  - ☐ Neutral (3)
  - ☐ Not Important (4)
  - ☐ Insignificant (5)
-

As a user, is having the option to filter matches that are further away from your personality important to you?

- ☐ Crucial (1)
- ☐ Important (2)
- ☐ Neutral (3)
- ☐ Not Important (4)
- ☐ Insignificant (5)

End of Block: Block 1

---

Start of Block: Question Block

Extended Feature Questions

---

As a user would you find it useful to be able to see information on other careers stored by the web service that are not related to your personality type?

- ☐ Strongly Agree (1)
  - ☐ Agree (2)
  - ☐ Neutral (3)
  - ☐ Disagree (4)
  - ☐ Strongly Disagree (5)
-

As a user would you find it useful to be able to see information on other personality types stored by the web service that are different from your own?

- ☐ Strongly Agree (1)
  - ☐ Agree (2)
  - ☐ Neutral (3)
  - ☐ Disagree (4)
  - ☐ Strongly Disagree (5)
- 

As a user how important is it to you to be able to save careers you have been recommended or searched for, making them more accessible than others you are not interested in?

- ☐ Crucial (1)
  - ☐ Important (2)
  - ☐ Neutral (3)
  - ☐ Not Important (4)
  - ☐ Insignificant (5)
-



As a user how important is it to you to be able to see a breakdown of your own personality type and the different factors involved?

- ☐ Very Important (1)
  - ☐ Important (2)
  - ☐ Neutral (3)
  - ☐ Not Important (4)
  - ☐ Insignificant (5)
- 

As a user how important is it to you that the system accounts for your known working preferences, such as office work or manual labour, when making recommendations to you?

- ☐ Crucial (1)
- ☐ Important (2)
- ☐ Neutral (3)
- ☐ Not Important (4)
- ☐ Insignificant (5)

End of Block: Question Block

---

Start of Block: Block 3

Final Questions

---

Would you use a system like this in your career search?

- ☐ Yes (2)
- ☐ Maybe (3)
- ☐ No (6)
- 

Would you find the website for useful if came with extended features such as searching careers and sectors by keyword or simplistic and built for the purpose of career recommendation only?

- ☐ Simple and Built for Purpose (1)
- ☐ Extended features (2)
- 

Q16 Any additional features you would find useful in a career recommendation website?

---

*Appendix E. Requirements Survey*

## Appendix F – Heuristic Design Analysis

### Heuristic and Accessibility Evaluation - Career Recommender System

Date: 08/06/2021

10 Heuristics:

1. Visibility of System Status
2. Match between system and the real world
3. User control and freedom
4. Consistency and Standards
5. Error prevention
6. Recognition over recall
7. Flexibility and efficiency of use
8. Aesthetics and minimalist design
9. Help users recognise, diagnose, and recover from errors
10. Help and Documentation

Credit: Jakob Nielsen (<https://www.nngroup.com/articles/ten-usability-heuristics/>)

Ratings:

- 0 – don't agree it's a problem
- 1 - cosmetic
- 2 - minorly affects usability
- 3 - major usability problem
- 4 - unusable with this problem

#### Violations

**Issue 1:** There are no loaders for the user while the recommender engine calculates the results

Screen: Results

Severity: 3

Heuristics violation: 1 (Visibility of system status)

Description: There are no loaders while the backend application processes the user results and creates their recommendations. If there are many matches or processing time increases in length while more jobs are added users may think the system has crashed. (Include Loaders)

**Issue 2:** Not clear that Job list items are clickable

Screen: Results

Severity: 3

Heuristics violation: 10 (Help and Documentation)

Description: It is not clear that a job list item is clickable and expandable in the results screen after the recommendations have been made. Experienced computer users will recognise this is a list item however unexperienced may not (include click to expand in the list item)

**Issue 3:** User must navigate back to the home screen to see information on how the results were calculated

Screen: Results Screen

Severity: 0

Accessibility violation: 6 (Recognition rather than recall)

Description: There's no context given for the results on the results page about how they were calculated, this was only detailed on the home page. Doesn't directly affect the usability of the programme and the information on how the recommender system works was only added extra detail on the homepage not integral to the user using the application.

**Issue 4:** Colour representation of match strength as opposed to numeric

Screen: Home Page

Severity: 4

Accessibility violation: Text Alternative for non-text content

Description: The career titles are coloured based on the strength of the match, this may have an adverse effect on users with colour blindness, as such should be changed to a numeric value.

**Issue 5:** Required fields in register form not marked with an Asterix

Screen: Register Page

Severity: 2

Heuristic Violation: Consistency and Standards

Description: It is common practice to have all required fields in a web form to be marked with an Asterix. Include Asterix on the title of each form element in the registration page to maintain this standard.

*Appendix F. Heuristic Evaluation*

## Appendix G – User Acceptance Tests

### Appendix G.1 – Session One

<b>Test name:</b> Complete the personality Test.	
<b>Test number:</b> 1	
Test step	Expected result
Press the take quiz button.	The quiz screen opens.
Answer the first set of questions present on the page by clicking the radio buttons.	The message is removed, and the next button is now shown and clickable.
Press the next button to move to the next set of questions.	The next set of unanswered questions appears.
Repeat steps 2 and 3 until the finish button appears.	The finish button appears and the quiz can be complete.
Press the finish button.	The results page is loaded.

<b>Test name:</b> View Recommendations.	
<b>Test number:</b> 2	
Test step	Expected result
Locate the career recommendations.	They should be present on the top of the page.
Click the recommended career option.	The option should expand to show all the information about the recommended career.

<b>Test name:</b> View Personality Type.	
<b>Test number:</b> 3	
Test step	Expected result
Scroll down to find the personality breakdown.	This should be present at the bottom of the page.
Read the personality type calculated.	A full breakdown of your personality and each component is given and understood.
Navigate Back to the home page	The home page loads.

<b>Test name:</b> View Quiz Information.	
<b>Test number:</b> 4	
Test step	Expected result
Scroll down to find the information on the test.	The Test information is at the bottom of the page.
Read the information on the test.	You now have a clearer idea on how the test works to calculate your recommendations.

## Appendix G.2 – Session Two

<b>Test name:</b> Registration and login.	
<b>Test number:</b> 1	
<b>Test step</b>	<b>Expected result</b>
Click the login button	The login page loads.
Click on the “sign up” link.	The registration page is shown.
Provide a username, email address, password and date of birth. Then click on the “Submit” button.	The application returns to the login window.
Enter your username and password.	The application returns to the home page and your provided first name is shown on the left of the navigation bar.

<b>Test name:</b> Match Recommendation Score.	
<b>Test number:</b> 2	
<b>Test step</b>	<b>Expected result</b>
Complete the quiz again as in session one	The quiz is complete and the result page is shown.
Locate the career recommendations.	Recommendations are shown and have a corresponding match score, recommendations are in descending order based on match score.

<b>Test name:</b> Quiz Results stored on database.	
<b>Test number:</b> 3	
<b>Test step</b>	<b>Expected result</b>
Close the browser.	The browser is closed.
Relaunch the browser and navigate back to the website	The website home page is loaded.
Click on the username in the navigation bar.	The dropdown is show
Click on the view results option	The user is navigated to the results page and the career recommendations and personality results are shown

<b>Test name:</b> Test results stored locally without an account	
<b>Test number:</b> 4	
<b>Test step</b>	<b>Expected result</b>
Click on the username in the navigation bar.	Dropdown appears.
Press logout	User is logged out, login option appears on the navigation bar.
Take the quiz again as in test 2.	The quiz result page is shown with the career recommendations and personality results
Close the current tab	Tab is closed
Relaunch the website	Home page loads with ‘View Results’ Present in the navigation bar
Click view results	Results page is now loaded with recommendations and personality result from second taken test, no match scores are shown.

### Appendix G.3 – Session Three

<b>Test name:</b> Auto Save Results to Account.	
<b>Test number:</b> 1	
<b>Test step</b>	<b>Expected result</b>
Please Complete the Quiz as in Session One and Two.	The results screen loads with personality type and recommendations.
Press the create account button on the results page.	The registration page loads.
Provide all the necessary details and then please check use my results and press register.	The home page loads and users name and drop down should be present at the top of the screen.
Click view results in the navigation bar.	Results page loads with same recommendations, user can now see match score alongside drop down filters for 'Sort By' and 'Filter By'. Matches are ordered in descending order.

<b>Test name:</b> Sort Results.	
<b>Test number:</b> 2	
<b>Test step</b>	<b>Expected result</b>
Click the 'Sort By' dropdown.	The dropdown appears with Ascending and Descending.
Click Ascending.	Matches are now ordered in ascending order based on match score.
Click Descending.	Matches are now ordered back in descending order based on match score.

<b>Test name:</b> Filter results based on personality component.	
<b>Test number:</b> 3	
<b>Test step</b>	<b>Expected result</b>
Click the 'Search By' dropdown.	The drop down shows with 'typecode' and the 4 personality factors based on your personality type.
Select a different personality component.	New results are loaded with no match score. The 'Sort By' menu has gone.
Click on a career.	The career expands. The personality code of the match should contain the chosen factor.
** Optional continue to expand careers.	Each career should contain the component you have selected in the dropdown.
Click the 'Search By' Dropdown and select another component beside typecode.	Matches are re rendered again with new results and identical format.
** Optional Expand careers and inspect results.	Factor is present in the typecode for that career.
Click 'Search By' dropdown and select typecode.	Original results are loaded ordered in descending order with match score.



<b>Test name:</b> Test Quiz Prevention	
<b>Test number:</b> 4	
<b>Test step</b>	<b>Expected result</b>
Click the 'Home' link to return to the home screen.	The home Screen loads.
Attempt to press take quiz.	The quiz prevents you from pressing the button, a message detailing the quiz can only be taken once in a 24hr period is shown.

<b>Test name:</b> Retake the Quiz With Warning	
<b>Test number:</b> 5	
<b>Test step</b>	<b>Expected result</b>
An Account that took the quiz yesterday has been provided to you. Please login using 'acceptance@email.com' and 'notapassword'.	The home Screen re loads.
Press the take quiz button.	A warning message appears, warning the user that taking the test again too soon will damage their results.
Press Okay.	The quiz loads.

<b>Test name:</b> Quiz Results stored on database.	
<b>Test number:</b> 6	
<b>Test step</b>	<b>Expected result</b>
Close the browser.	The browser is closed.
Relaunch the browser and navigate back to the website	The website home page is loaded.
Click on the username in the navigation bar.	The dropdown is show
Click on the view results option	The user is navigated to the results page and the career recommendations and personality results are shown

## Appendix H – Acceptance Test and Story Mapping Table

User Requirement	Requirement ID	Test Number	Validation Status
<b>Session One</b>			
As a user I want to be able to take the personality quiz so I can view my personality type that has been determined by the test	1	1	Passed
As a user I want to be able to be able to apply a grading to a question so I can create an accurate representation of how that question applies to my personality instead of a singular yes or no	6	1	Passed
As a user I want to be able to see my recommendation results so that I can see which careers are associated with my particular personality type	2	2	Passed
As a user I want to be able to see a full description of the recommended careers so I can gather a more in-depth detail of career paths recommended to me	5	2	Passed
As a user I want to be able to view my top recommendations quickly so I can clearly distinguish the important information and be given a choice of consuming further information	10	2	Passed
As a user I want to be able to view a breakdown of my personality results so I can use this to better assess my recommended careers or make my own decisions on a career	12	3	Passed
As a user I want to be able to view information about the test so I can gather a good understanding of how the test works and how my results were calculated	9	4	Passed
<b>Session Two</b>			
As a user I want to be able to create an account so that my own data is associated with me	34	1	Passed
As a user I want to be able to log into my account so that I can access the data that is held for me on the website and make use of it in my career search	35	1	Passed
As a user I want to be able to see a representation of the strength of each of my matches between user personality and job so I can assess which ones are best	3	2	Passed
As a user I want to be able to store my results to the personality quiz and close my browser so I can return at any time and view the results and use them in further analysis of choosing a career	19	3	Passed

As a user I want to be able to navigate away from the page and come back later to view my results so I can use the information to carry out my own searches on the web	11	4	Passed
<b>Sprint Three</b>			
As a user I want to be able to add my quiz results to my account upon creation if I have taken the test so I don't need to take the test again	39	1	Passed
As a user I want to be able to order my matches that are closer/further away from my personality so that I can see the strongest or the weakest matches	16	2	Passed
As a user I want to be able to filter results based on certain matching personality traits so I can see careers based around the traits I feel are my strongest	25	3	Passed
As a user I want to be prevented from immediately changing my answers after the test has been completed so I can be prevented from changing my true results to something I would prefer	13	4	Passed
As a user I want to be informed on the risk of taking the test again to soon so I can make an informed decision about whether I wish to risk harming the results	20	5	Passed
As a user I want to be able to retake the test after some time has passed so I can keep my results in line with how my personality has changed and matured over time	17	5	Passed

Appendix H. Acceptance Test and Story Mapping Table