School of Computing

FACULTY OF ENGINEERING AND
PHYSICAL SCIENCES

UNIVERSITY OF LEEDS

Final Report

ProjectFinder
A Web Application for PricewaterHouseCoopers using the Django Framework within the Python Language

Efthymios Kyriakou

Submitted in accordance with the requirements for the degree of
BSc Computer Science

2022/23

COMP3931 Individual Project

The candidate confirms that the following have been submitted*:*

Items	**Format**	**Recipient(s) and Date**
Final Report	*PDF file*	*Uploaded to Minerva (02/05/23)*
Link to online code repository	*URL*	*Sent to supervisor and assessor (02/05/23)*

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student) [illegible]

Summary

This project explored the resource management at PricewaterHouseCoopers (PwC). The Django framework within the Python Language was used to develop an alpha prototype web application that would advance the resourcing process in use at PwC currently. This would in turn benefit the client is providing an application that would make the resourcing more efficient and benefit the overall production to their clientele.

Acknowledgements

I want to thank my supervisor Dr. Nick Efford for his guidance throughout the project. I want to thank Susan Golligher for assisting with the requirements gathering process, finding PwC employees to test my application and her overall support on the project. I want to thank the PwC staff which assisted with the testing for all their time and effort.

Table of Contents

Chapter 1
Introduction and Background Research

1.1 Introduction

The aim of the project was to create a web application to assist PricewaterhouseCoopers (PwC) to support their resource management process. This covered assigning employees to suitable project roles and providing a more efficient way to manage their resources.

The company has an existing system that is not meeting users' needs, resulting in manual steps and additional effort from staff. A streamlined application was proposed to improve the overall process and benefit the organisation.

For the existing process, when a Director or Partner require resources for projects, they will direct their query to their department's Resourcer. In their request, they will include their required employee grade, skillset and potential length of the project and any other relevant information. The Resourcer will then shortlist the right contacts based on the specified criteria. The resource search, identification and shortlisting include manual input and multiple inefficient steps.

To overcome this inefficiency, the following web application was recommended. The web application partially automated the process and improved the overall efficiency. In addition, this report will analyse the process of assignment and selection by introducing a system that will assist the overall search and shortlisting for resources per project.

To achieve this, the current process, key stakeholder needs and their implications for the web applications were investigated. The project was guided and supported by academic research and analysis but also existing solutions within the field of resource management. To ensure that employees needs were at heart, consistent feedback was provided.

Furthermore, the different technologies were assessed to ensure that a tailored solution was generated by choosing the best available technologies with the required business capabilities. This also included considerations for implementation but also potential impacts to the current business architecture of the firm.

To evaluate the proposed solution, research around different testing methods and techniques was required when conducting testing and when presenting the application to PwC. A defined number of testers were selected based on existing software engineering and academic standards. This was so that we can get a clear idea of potential areas of improvement and to ensure that the required criteria were met.

1.2 Existing Solution

1.2.1 Introduction

To initialise the project, requirements needed to be defined. Requirements are a list of essential features needed for a system designed to provide value to the stakeholders (Young, 2002).

To effectively gather a list of concise requirements, there are several techniques recommended by Young R. R. (2002) that can be implemented. These include:

- interviews
- stakeholder conversations
- observations of users
- prototyping
- surveys

Due to the nature of the project, it was best to speak to the lead Resourcer at PwC to understand the current process to create an initial list of requirements. Consequently, use cases and prototypes were given to the Resourcer for feedback and to adjust and produce a clearly documented list of requirements.

1.2.2 Current Process

During the first meeting, the stakeholder defined the existing resourcing process. This is as follows:

1. A Partner, Director or Manager at PwC fills out a Google Spreadsheet to request staff for project roles.
2. The Spreadsheet is submitted to the Resourcer.
    - This form requires details such as: The client's name, the role scope, the skills required, the industry, the sector, start date, duration, location, and grade.
3. The Resourcer reads the form and transfers all the input into a Google Spreadsheet, which contains all the available roles.
4. Employees are given a link to the Google Spreadsheet, that contains all available roles.
5. If the employee is interested in an opportunity, they click on a link to a Google Form, where they enter specific details about the opportunity, they are interested in.
6. That Google Form is submitted and received by the Resourcer.
7. The Resourcer gathers the names of the applicants and co-ordinates with the requester to view the employees TalentLink profile.
8. The Resourcer and the requester then decide on which employee to assign to the available role.

9. Once a candidate is selected and confirmed a booking is made on TalentLink to show their availability for the future based on the length of the specified project.

All PwC staff roles and internal web application services are defined in the Glossary.

1.2.3 Current process inefficiencies and potential improvements

Following a discussion with the PwC Resourcer, the findings were presented. This included an analysis of current process weaknesses, inefficiencies, and potential solutions using a custom application. The application developed is named ProjectFinder.

The use of Google Spreadsheets and Google Forms to transfer information was time costly, due to the transfer time and manually copying information from one spreadsheet to the other. This would be improved by having one central database where the requester can directly create and store available project roles. The employee is shown the available roles from their end through database queries. This also eliminates the need for the Resourcer to act as a mediator in the information transfer process.

Secondly, with the constant accessing and transferring of online files there is a data integrity issue. This is because other employees may have access to this information within the spreadsheets which can jeopardise privacy and security. With the use of the ProjectFinder there would not be unauthorised access as the database will be secured and it will have user authentication and role-based access control.

ProjectFinder will make it easier for stakeholders to keep track and view applicants, their skills and any other required details. This will be done through ProjectFinder rather than accessing individual Google Forms and then TalentLink to view each employee’s information.

Finally, an additional implementation of a match rating system that would be displayed for both the applicants and requesters to view which employee matches best to their corresponding role. For example, matching required skills of a role to the skills held by an employee, the start date and duration of a project to the availability of the employee and further details (Wu & Sun, 2006).

1.3 Requirements Gathering

1.3.1 Requirement Gathering Techniques

Upon presenting the findings and suggested improvements, a list was generated which comprised of requirements. Furthermore, there was a discussion with the Resourcer to agree on the final requirements for the application.

As part of the requirements, it was important to consider both functional and non-functional requirements. These two categories define two separate parts vital to a successful software model. Non-Functional requirements focus on the quality of the product, such as the UI and the security of the software, whereas the functional requirements are essential features to meet the business needs (Jain & Ingle, 2011).

The requirements were documented in a table and listed along with any feedback received from stakeholders. The "MoSCoW" approach was implemented when documenting the final list of requirements. "MoSCoW" is a simple mnemonic that helps in prioritising requirements, assigning requirements to one of three categories, Must Have, Could Have, Would Like to Have. Prioritising requirements is essential in delivering a valuable product, on time and cost-effectively (Kuhn, 2009).

To assign each requirement to one of the three categories, two numbering systems were utilised when presenting the initial requirement list to the client. One focuses on the importance of the feature and the value it provides to the system. The other comments on the difficulty of developing that feature. These grading systems were agreed upon through collaboration with the stakeholders.

1.3.2 Agreed Requirements

Table 1: Requirements Table

ID	Requirement	Feedback	Difficulty (1-3)	Priority (1-3)	MoSCoW
1	Allow users to register to the web application.	Fields need to be editable in the future. Change of grades, location, addition of skills.	1	1	Must Have.
2	Allow users to log in to the web application.	N/A	1	1	Must Have.
3	Allow users to create a profile.	N/A	1	1	Must Have.
4	Create a different view for Directors/Managers/Partners to be able to create job availability for employees.	Partners and Directors should be able to only request for staff they do not need to be resourced. Managers and Senior Managers can request for	2	1	Must have 2 views apply and create. Could have hybrid

		staff and can be resourced to projects.			view for managers.
5	Create a view for the Resourcer to be able to view job applicants.	Partners, Directors and Managers should also view applicants.	2	1	Must have.
6	Create a matching system to show a percentage of match between employee and job comparing the skills the employee has with the skills required.	May be adapted in future depending on implementation.	3	2	Could Have.
7	Adjust matching system, linking availability from existing software within PwC.	Linked to requirement below. All present within one internal system.	3	3	Would like to have.
8	Link TalentLink web application to be able to view TalentLink Profiles directly.	N/A	3	3	Would like to have.
9	Generate user friendly UI.	N/A	2	2	Could Have.
10	Clean design that matches the PwC scheme.	N/A	2	2	Could Have.
11	Display job opportunities and projects to employees according to their grade.	Flexibility for certain roles as can be suitable for multiple grades.	2	1	Must Have.
12	Create a database model with a table of employees and available projects.	An extra table is required to store applications made.	2	1	Must have.

13	Profile needs to include: • Basic Details • Grade • Line of Service • Skills • Languages • Location	Do not store Date of Birth or Address of employees as PwC do not have those accessible in TalentLink.	1	2	Could Have.
14	Speed up process of assigning employees to project roles.	Difficult to assess as process depends on stage the project is at.	1	1	Must have.

1.4 Technologies

1.4.1 Web Application

From discussions with the stakeholder, it was agreed that this project would deliver a prototype application that will be used as a proof of concept. It will require further implementation and amendments in the future to be used by PwC.

For that reason, it was decided with key stakeholders that ProjectFinder would be implemented as web application rather than a mobile application or a software application. There were several benefits to prototyping the application as a web application. It bypassed a security issue, as development did not occur on company hardware. Deploying the web application via cloud-based services, such as PythonAnywhere, was simple to do and that gave access to the stakeholders and testers immediately (Amunategui Manuel and Roopaei, 2018). This facilitated the overall development process and prototyping.

1.4.2 HTML and CSS

The development of a web application required a variety of technologies. Hypertext Markup Language (HTML) and Cascading Style Sheets (CSS) are essential in web development, since all websites use HTML and CSS (Duckett & Schlüter, 2011). HTML allows for organisation of information with tags and is responsible for the display and view of a web page (Dauzon et al., 2016). HTML has support for contents to be displayed on web pages ranging from spreadsheets, text, videos and images and it is an essential part of web development (Ghimire, 2020).

1.4.3 Dynamic Web applications

To fulfil the requirements and deliver a product that was required by the client, there was a dynamic aspect involved with the web application. To produce this dynamic web application, it was necessary to develop a software application that can run on a website using a web development language and framework. There are many popular dynamic scripting languages such as JavaScript, PHP and Python. Node.js is a fairly new technology within JavaScript which has become popular in web development. PHP is outdated technology that is only suitable for small scale projects and although Node.js is known to be efficient and lightweight, it is ideal for large web applications. According to tests ran by Kai Lei, Yining Ma and Zhi Tan (2014), Node.js is a better performing technology when compared to PHP and Python in terms of users' requests and time per request. Python however provides a good standard of performance as well as being developer-friendly, simple to learn and good for large web architectures (Lei et al., 2014).

Through research and extensive usage of the language, Python was selected as the programming language to develop the backend of the web application, in combination with JINJA2, a python template engine , and JavaScript.

1.4.4 Python

Python was invented in December 1989 by Guid van Rossum in the Netherlands as a successor to the ABC programming language (Tulchak & Marchuk, 2016). It is a powerful programming language that is very simple to read and understand. It displays many useful features such as accessing databases, generating portable programs, and creating functions (Yuill & Halpin Harry, 2006). As a programming language it is one of the fastest growing languages and it is used for app, web, game development and other technologies (Chandra, R., T., 2021). Python has several advantages over other programming languages, including clean syntax, useful modules, and a powerful development environment (Tulchak & Marchuk, 2016). Furthermore, Python is a very appealing programming language due to its readability, the syntax and the code is very similar to the English language. This reduces time spent in learning to write code in Python as well as making it easier to understand existing code (Ghimire, 2020).

1.4.5 Python Web Development Frameworks

"A software framework is a set of standardised libraries and tools for a programming language" (Ghimire, 2020, p.1). Two of the most well-known Python web application frameworks are Flask and Django due to the fact that they enable developers to produce a functioning web application in short space of time (Ghimire, 2020)

Flask is a micro-framework designed for quick web application development. By only implementing core functionality and giving freedom to the developer to add features required during development. This means that Flask does not have support for accessing databases, validating forms and users and other high-level tasks (Grinberg, 2018).

Django is a widely used web development framework used in generating complex web applications. The framework is written in Python and it follows a Model-View-Controller (MVC) architecture, which assists in developing and maintaining web applications. The MVC approach is a way of developing software, where defining and accessing data is separated from the request-routing logic and they are both separated from the user interface. The key benefit of this approach is that each unique part of a Django web application has its own distinct purpose and can be altered without affecting any other part (Holovaty & Kaplan-Moss, 2009). The Django framework embraces the creating of complex, database-driven websites, whilst emphasizing reusability of code, quick development, reduced code and Don't Repeat Yourself (DRY) principle. In addition, Django provides administrative interface that is generated dynamically, this allows direct access to the database with the ability to delete, update and read database records (Shyam & Mukesh, 2020).

Both frameworks allow for an easy and quick structure for developing a web application. Django provides nested support for several features and it is simple to develop more complex web applications. Therefore, it was a better choice for a framework to develop with rather than Flask for this project.

1.4.6 Databases

For the project a database was essential to store relevant information such as employee accounts and records, projects and project roles and also applications submitted by employees.

Django can interface with many databases. However, the default configuration for Django uses SQLite as the choice of database and SQLite is also included within Python (Django, 2023). Therefore using SQLite did not require any additional installations. SQLite is an in-process library that implements a SQL database engine. SQLite is compact, fast and has connections to several programming languages such as C, C++, Python, Java, and JavaScript. SQLite is best used for portable and multi-user applications (Bhosale et al., 2015).

The proposed web application required portability and multi-user access. SQLite possesses those characteristics and it is part of the Django default configuration, hence why it was the selected database engine.

Chapter 2
Development Approach and Design

2.1 Development Approach

A software development methodology can be defined as a set of guidelines and procedures that are used as part of the several stages of software development, ranging from researching, development and deployment (Despa, 2014). Software development methodologies are used and relied upon in software development projects. It is essential that the best suited approach is selected for a software development project to succeed.

There are many popular software development methodologies; Extreme Programming, the Waterfall methodology, Iterative and Incremental, Scrum and many others (Saeed et al., 2019).

This project had a sole member taking the role of the developer and project manager. The different methods available were researched and analysed to make an informed decision on which software development methodology was best suited to the project.

The Waterfall methodology is the first software development methodology and it is based on a well-defined set of processes that must be performed sequentially. The current step in the methodology must be completed before the next can be initiated. Any comments or feedback from the stakeholders are received at the end when the development and testing processes have been completed. This methodology relies heavily on documentation and requires clearly defined and fixed requirements from early before any development can begin (Saeed et al., 2019).

The Iterative and Incremental methodology, Extreme Programming and Scrum are Agile methodologies. There are over 20 different agile methodologies. Agile methodologies are popular and focus heavily on constant communication and feedback from the clients. They are preferred to other more structured approaches due to the flexibility they offer in development (Rasnacis & Berzisa, 2017).

The iterative and incremental methodology is an approach based on a divide and conquer approach. It splits the task into parts and the most high risk and time consuming parts are completed first. This is because those are the parts most responsible for the project's success so it is better to concentrate and complete those first to ensure that the project is successful (Martin, 1999). Each part is designed, developed and tested and the feedback from testers and stakeholders is used to make adjustments and once a part is completed then the same stages begin for the next part. This method is preferred to the Waterfall method in certain cases because there is constant communication with the stakeholders and the feedback gained after each part is completed allows for flexibility.

Scrum is another Agile framework, consisting of a sequence of brief iterations called sprints. Each sprint performs parallel analysis, design, development and testing and is terminated with a release of a deliverable. Scrum methods are described as simple, flexible, providing insight and success (Rover et al., 2014). Each sprint produces a single deliverable, so it only has one goal and supports projects with numerous deliveries and ordering of user stories (Schwaber, 1997).

It was evident from the feedback, even though the requirements were decided early in the project, there was a need for flexibility for the requirements. The agreed method of identifying each individual users' needs would be by developing prototypes in mini sprints with incremental functionality per prototype.

The best approach aligning with the given nature of the project was a type of Agile methodology that would be altered to suit the project's needs. An initial list of requirements was generated with the stakeholder and the requirements would be separated into parts. The iterations delivered focused on feature implementation and incrementing the number of features as well as the capabilities of each feature.

2.2 Design

The design documents were presented to the client prior to commencing development. The final web application that was developed did not fully match the design documents.

2.2.1 Flowcharts

Flowcharts are a vital tool for any software development project as they provide a visual representation of processes and data flows involved in any application (Xinogalos, 2013). Flowcharts provide a better understanding of an application and its functionality as displayed by the results (David A. Scanlan, 1989). The stakeholder activities and involvement in the project plays a major role in the quality of the project and whether the product is delivered within budget and in time (McManus, 2004). The use of flowcharts ensured that the application can be understood better by the developer, the stakeholders and future developers, which in turn warranted for clearer communication between all parties involved.

Flowcharts may differ from hand drawn sketches to extensive diagrams generated with specialised software. Hence the need for standards for the symbols used in flowcharts. The American National Standards Institute (ANSI) set the first official standards for flowchart diagrams in the 1960s, which were then adopted by the International Organisation for Standardisation (ISO) in 1970 and have been updated and improved since (Asana, 2023). The latest standard, which was used in this project, were confirmed in 2019, ISO 5807 (International Organization for Standardization, 2019).

Lucidchart, a free web application was used for the development of flowcharts for the project. It is a cloud-based platform that can be used on any device, operating system and device (Lucid, 2023). This provided elasticity for the process.

2.2.1.1 Main Flow

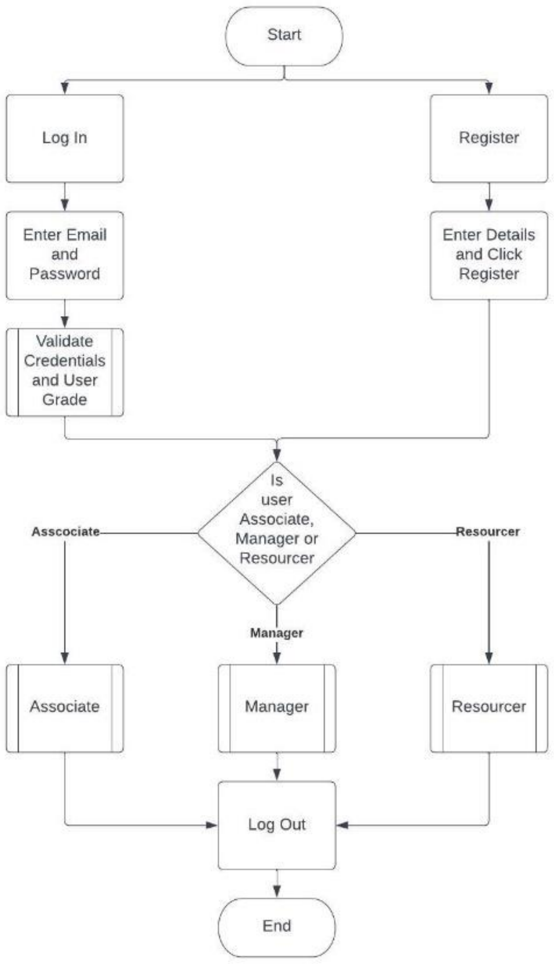

Start
Log In
Register
Enter Email and Password
Enter Details and Click Register
Validate Credentials and User Grade
Is user Associate, Manager or Resourcer
Asscociate
Resourcer
Manager
Associate
Manager
Resourcer
Log Out
End

Figure 1: Main Flow Flowchart.

The web application was split into three flows depending on the users grade; one for Trainees/Associates/Senior Associates, one for Managers/Senior Managers and one for Directors/Partners/Resourcers.

The main difference between the three flows is that Trainees, Associates and Senior Associates would be able to view roles and apply to those roles. Where as the Directors, Partners and Resourcers would only be able to create available project roles and view applicants for each role they have created. The Managers and Senior Managers would be able to do both. These were agreed upon with the lead stakeholder, the Senior Resourcer and can be linked back to the requirements ID 4 and ID 5.

The main flow of the web application can be seen in Figure 1.

The main flow is the flow which each user follows when starting the web application. They would have a choice to Log In or Register.

New users would have to Register. To register they will be redirected to a page with a form where they would enter their details. These details are comprimised of basic identifiers, such as first name, last name, email and a password. The rest of the details all relate to the clients needs, relating to the requirement ID 13.

Returning users would be able to log in with the credentials they have created. They would need to enter their email and password. The credentials would then be validated. If they are validated then the user would move on to one of three home pages and a sub-process is called. For simplicity, in the flowchart this was shown as Associate, Manager and Resourcer.

2.2.1.2 Associate Flow

If the user registers or logs in successfully and they belong to one of the grades Associate, Senior Associate or Trainee, they would follow the sub-process “Associate”. The flowchart is shown in Figure 2.

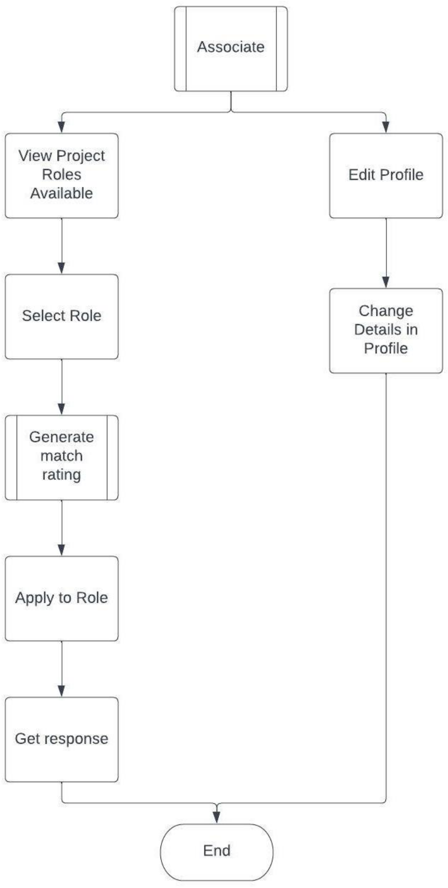


Figure 2: Associate Flow Flowchart

They could choose to view available project roles listed and more details of a role. When they view more details of a role, a match rating would be generated, a percentage of suitability for the user to the role selected. They could then apply for that role and they would get a response, whether they were accepted or rejected.

They could also edit their profile, as requested by the key stakeholder and Senior Resource Manager. If they choose to edit their profile, they can edit any of the fields they have filled in when creating their original profile. This may include, adding new skills, changing their line of service or grade.

2.2.1.3 Match Rating

For the Match Rating sub-process, as designed initially, the process would require access to the database. From the Project Roles table the skills required, duration, location and languages would be needed. From the Employee table the applicants skills, availability, location and languages would be required. The two would be compared and a match percentage rating would be generated and shown to the applicant. This can be seen in Figure 3.

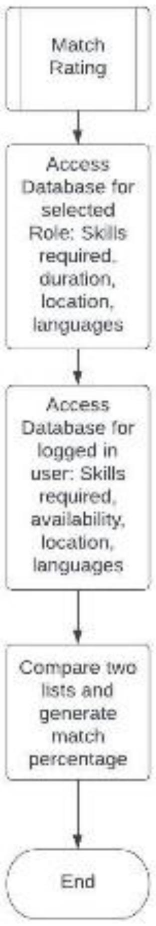

Match Rating
Access Database for selected Role: Skills required, duration, location, languages
Access Database for logged in user: Skills required, availability, location, languages
Compare two lists and generate match percentage
End

Figure 3: Match Rating Flowchart

2.2.1.4 Resourcer Flow

If the user registers or logs in successfully and they area a Director, Partner or Resourcer, they would follow the sub-process “Resourcer”. The flowchart can be seen in Figure 4.

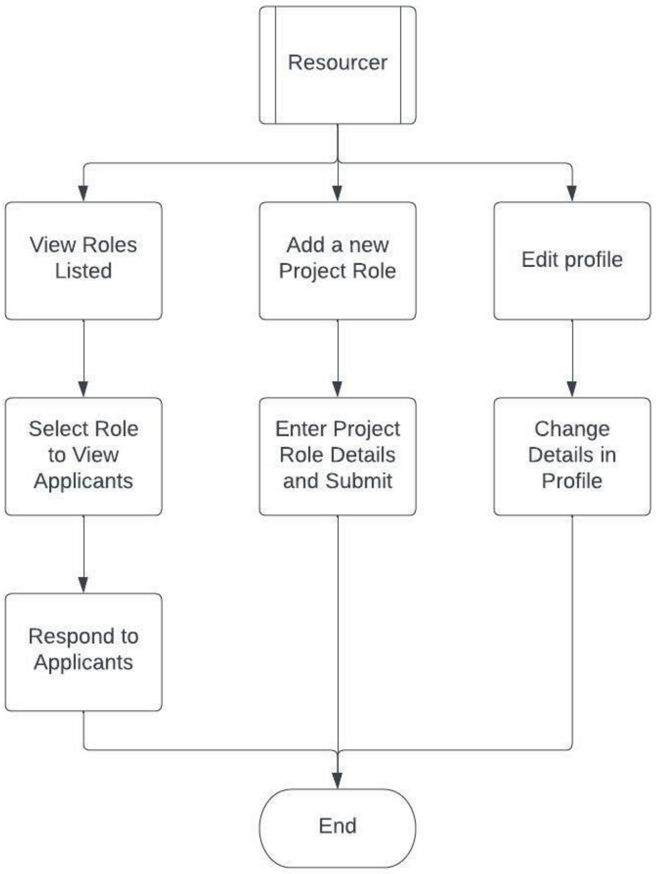

Resourcer
View Roles Listed
Add a new Project Role
Edit profile
Select Role to View Applicants
Enter Project Role Details and Submit
Change Details in Profile
Respond to Applicants
End

Figure 4: Resourcer Flow Flowchart

They could choose to add an available project role. This would entail adding the details of the project role available, such as the project title, the grade required for the position and the skills required.

Once they have added an available role, they would have the ability to view the role and whether it is listed. From there, they would be able to select to view the applicants for that position. The details of the applicants would be displayed along with the match rating of the applicant against that role. They could then accept or reject the applicant for that role based on their individual assessment.

They could also edit their profile similarly to the Associate sub-process. If the user chooses to edit their profile, they could edit any of the fields they have filled in when creating their original profile. This may include, adding new skills, changing their line of service or grade.

2.2.1.5 Manager Flow

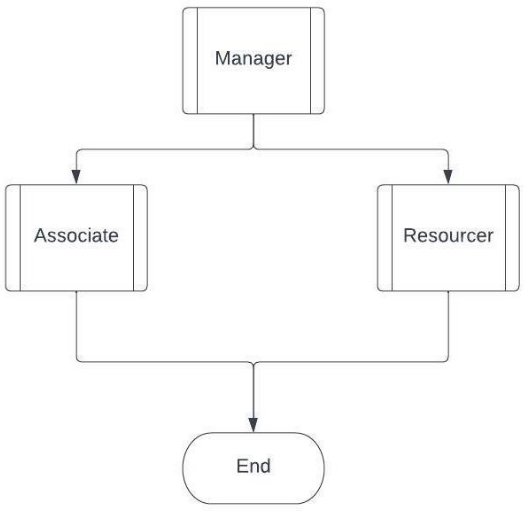


Figure 5: Manager Flow Flowchart

Finally, there is the "Manager" sub-process which would be followed by any user that successfully registers or logs in successfully and they belong to a grade Manager or Senior Manager. This flowchart can be seen in Figure 5.

As per the feedback for requirement ID 4, Managers and Senior Managers would be able to request for staff for a project they are already a part of, as well being able to apply for available roles and be resourced accordingly.

This flow has been split into two parts, one which follows the "Associate" sub-process and the other follows the "Resourcer" sub-process. This means that these users would have the ability to perform any tasks that belong to the "Associate " sub-process, as well as any tasks that belong to the "Resourcer" sub-process.

2.2.2 Wireframes

A wireframe is described as an approximate visual design of a proposed application (Lloyd, 2009). Wireframing is a type of prototyping, while the proposed application was seen as a prototype itself, the application was an alpha prototype, meaning that it provided partial functionality. In contrast a wireframe is a digitally produced sketch that has no interactive functionality. They were useful for presenting the functional scope of the application to the stakeholders. In addition, they were used in presenting a walkthrough of the application to the key stakeholder without developing any code (Roth et al., 2017).

To create the wireframes, which were presented to the stakeholders, the free online web application, Figma was used. Figma works on Windows, Chrome, Mac and Linux. It was the selected web application as it provided version control, is lightweight, quick and easy to use and it did not require any installations, to be used (Figma, 2023).

The wireframes generated and shown to the client were used as examples, the application would slightly differ from the designed wireframes.

2.2.2.1 Examples

Below there are some examples of the wireframe sketches used.

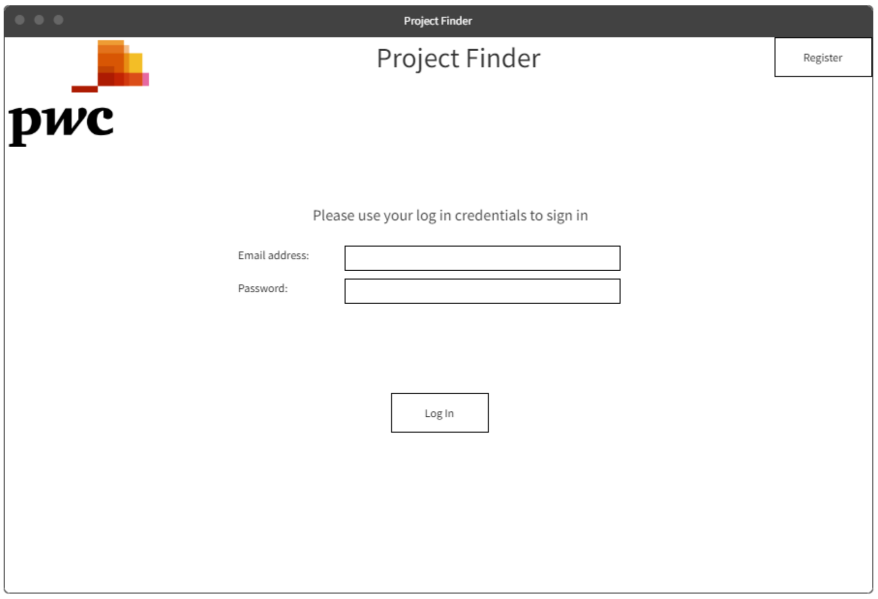


Figure 6: Log in Page Wireframe.

Figure 6 shows the log in page proposed to the client, relating to requirement ID 2. A web browser frame is used to represent the web application running on a web browser. The title of the app is at the top centre of the page. The logo of the company is used on the top left, standardised with other PwC web applications. For the labels, such as Email Address and Password, standard text and blank rectangles were used to represent the input boxes. Square buttons are used with text within to represent their functionality, e.g. Log In and Register.

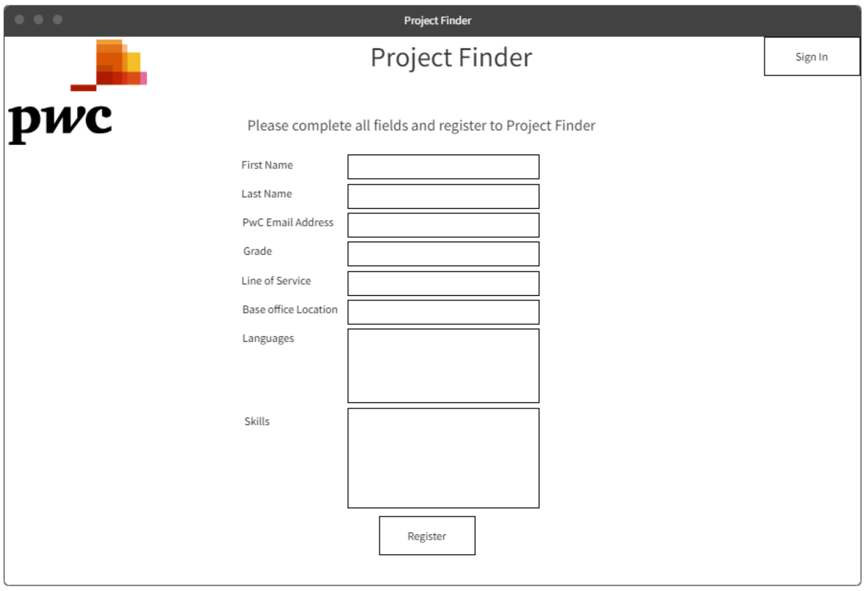


Figure 7: Registration Page Wireframe

Figure 7 is an example of the Register page. This provided clarity as to the details the users will have to enter for their profile when creating an account, relating to requirement ID 1 and 13. Once again, the logo is in the same position top left with the title of the web application top centre. The theme is standard across all wireframes. They are simple black and white in colour to increase time efficiency. The labels are alongside the entry boxes to make it clear to the user what they are required to input in each field.

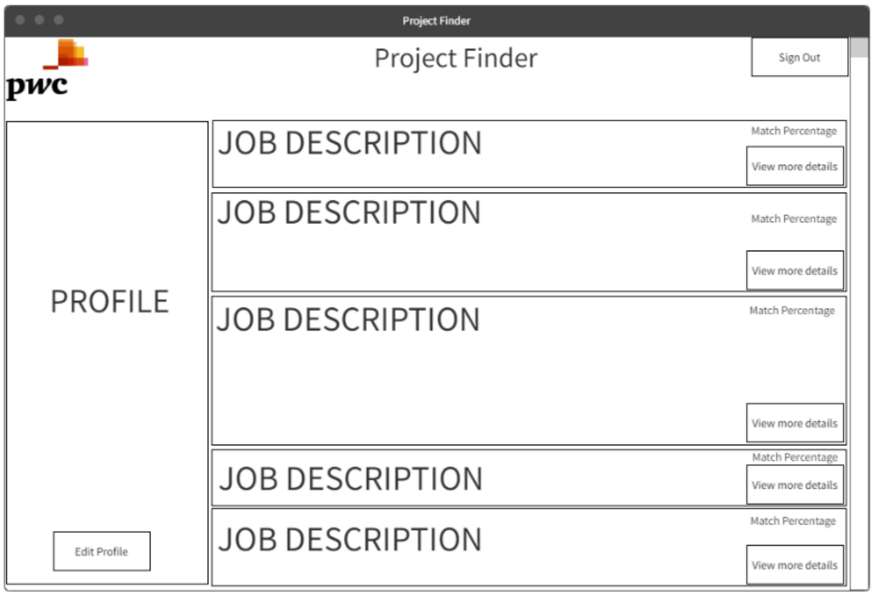


Figure 8: Available Roles Page Wireframe

Figure 8 shows the view when an employee would like to view the available roles listed to which they can apply to. As can be seen each rectangle labelled "Job Description" represents an available project role. The details listed will be brief, for example: name, role, title and grade required. The match rating will also be displayed. When the employee clicks on the "View more details" button on the bottom right of the rectangle they will be able to see further details such as the role description, the skills and languages required.

On the left a brief outlook of the user's profile will be displayed and they will have the ability to edit their profile.

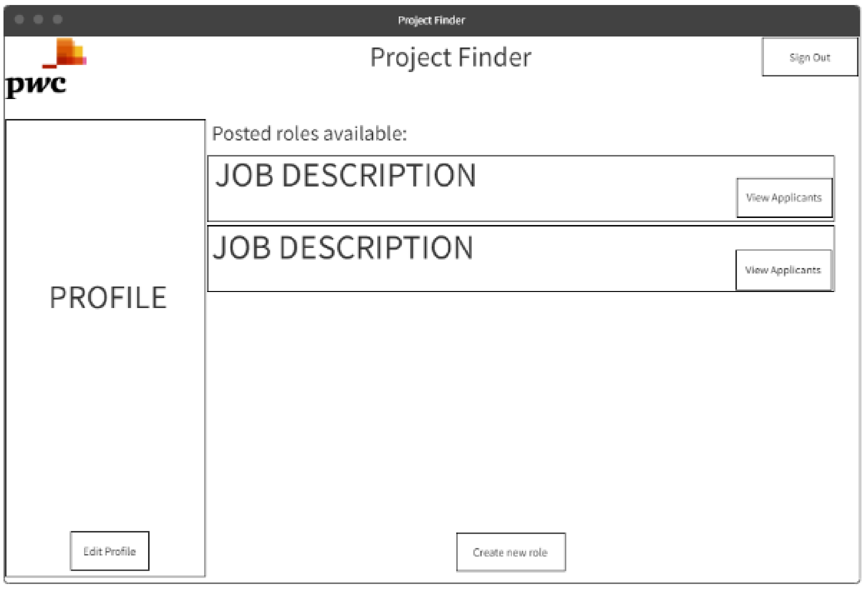


Figure 9: Requester Roles Listed Page Wireframe

Figure 9 shows the wireframe which was designed to represent the view which employees would see when they have listed roles for other employees to apply to. Figure below and figure above have a similar structure, profile is on the left with "Edit Profile" button at the bottom. On the right side of the page, applicants will be listed with brief overview, details such as name and location and match rating. The "View Applicant" button can be clicked to go to a page where the whole profile of the applicant would be listed, with their skills and languages. That page will include buttons for the requester to be able to respond to the applicant.

The rest of the wireframes presented to the client can be found in Appendix C.1, under the Wireframes section.

2.2.3 Database design

A database is an application which enables the storing and retrieving of data and a relational database model is a collection of data tables, where the tables are connected via a unique identifier field (Jatana et al., 2012). A database was essential for the proposed application to function. There was a need to store each employees' details which will be collected upon registration.

The employee records would be used for validation when the employees log in to their account. The database was required to store the project roles that created by requesters and there was a need to store the applications which employees have made to roles.

The Employees, Project Roles and Applications are three different types of records, so three different tables were required for the database and each table was connected to another table.

An Entity-Relationship diagram was used to model the database and assist with providing an explanation of how data would be stored and used to the stakeholder. An Entity-Relationship Diagram is way of modelling databases. Its naming convention for it originated because “it depicts data in terms of the entities and relationships described by the data” (Li Qing and Chen, 2009, p.126).

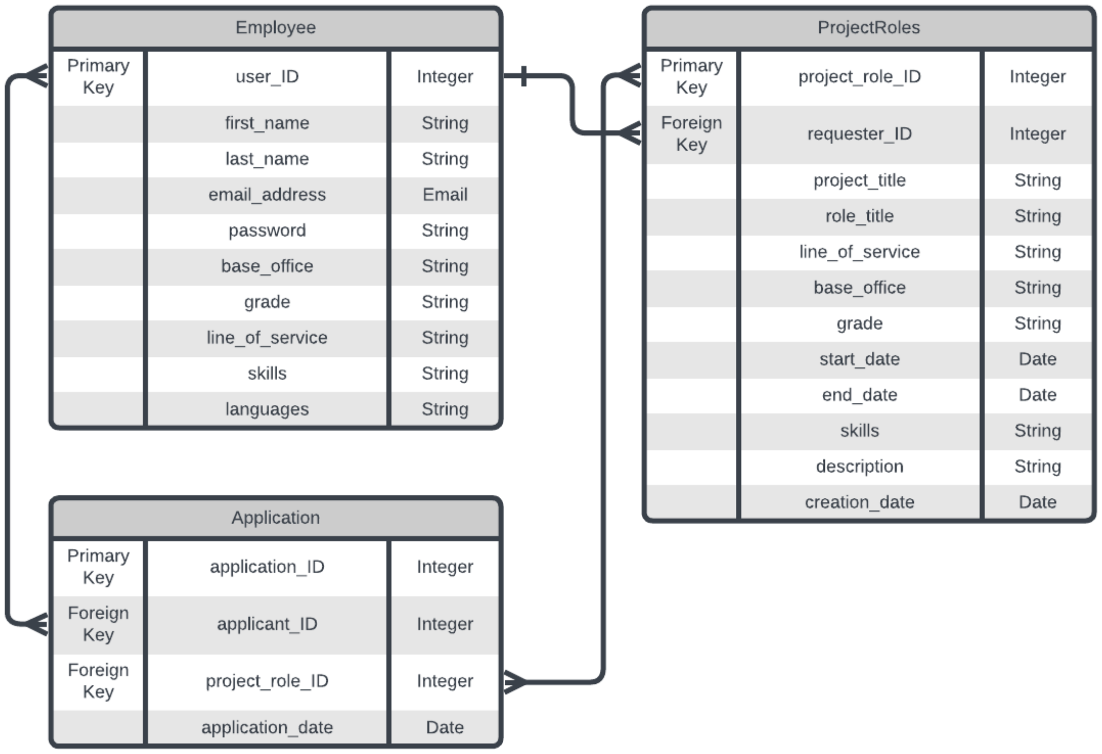


Figure 10: Entity Relation Diagram showing proposed database design for web application

Standard naming conventions were used to represent the fields in the diagram and those were all relevant details required for each of the entities, as requested by the stakeholders in the requirements. The free online web tool Lucidchart was used to create the ER diagram (Lucid, 2023).

Primary keys are the unique identifiers for each of the entities, where as a foreign key is a primary key of a table present in another table to establish a relationship between the two entities (Revesz, 2010).

In this case “user_ID” for the “Employee” table, “project_role_ID” for the “ProjectRoles” table and “application_ID” for the “Application” table are the primary keys of each entity. “user_ID” is used to connect the “Employee” entity to the “ProjectRole” entity and “user_ID” as well as “project_role_ID” are used to link those two entities to the “Application” entity.

The “ProjectRole” table was connected to the “Employee” table via a one to many relationship, because each project role would need to have one requester, therefore one employee. An employee could be a requester for many project roles.

The “Application” table has a many to many relationship with both the “Employee” table and the “ProjectRole” table. Employees could have many applications and project roles could have many applicants.

2.3 Coding Standards and Version Control

2.3.1 Version Control

Version Control is an key tool for a successful software development project. It provides many benefits of which include:

- Access to file history at any point. When a new change is committed the old version is stored in Version Control System (VCS) repository. If new additions cause an error, the code can be traced back to a version where it was functioning and developers can investigate the cause.
- The ability to branch and merge allows for team members to work simultaneously on the project and potentially on different areas. Branching creates individual streams from the central storage of the application and those branches can be merged back into the central storage of the software. Although currently the project has one developer, this provides a platform for future developments.
- It provides traceability. All commits are given comments which are beneficial for a project as each change is described and reasoned. This supports the documentation process by keeping track of the progress of the software.

(Spinellis, 2005)

The Version Control System of choice for this project was Git. Git was created by Linus Torvalds and is a powerful and flexible VCS that is simple and free to use as well as easily available (Loeliger & McCullough, 2012).

2.3.2 Coding Standards

Appropriate coding standards were followed throughout the project. Comments were utilised in development to describe code for future reference and to provide an understanding for other viewers, such as future developers. In addition, variables, functions and files were named appropriately, using clear and concise names to avoid confusion. Finally, the correct Python syntax was followed, through the use of Python Documentation (Python, 2023).

Chapter 3
Results

3.1 Prototype 1

As per selected development approach, a series of mini sprints were completed. A partially functioning web application was delivered at the end of each sprint for the client testers to test and give feedback.

For the first Prototype, some of the requirements were addressed. This was done to provide a proof of concept and an example for the idea presented.

For this prototype, the must have requirements are prioritised. These include the requirements with the IDs: 1, 2, 3, 4, 5, 11, 12 from the Requirements shown in Table 1 in Chapter 1.

3.1.1 Implementation

3.1.1.1 Database Implementation (Requirement 12)

Initially, a database was generated within the “models.py” file. Three models were created within the database one for each as per Figure 10:

- the employees
- the project roles
- the applications

For the employee model, the in-built Django “User” model was used. A “User” object is created and stored within the “EmployeeUser” model. The “User” model, contains the fields: First Name, Last Name, Username and Password (Django, 2023):

The username field was used to store the employee emails. The “EmployeeUser” model contained extra details as requested by the client such as: grade, line of service, skills, languages and location.

3.1.1.2 Registration (Requirement 1 and 3)

For the registration of users, a HTML file was created with a form tag. This created a form with labels and input boxes for the users to enter their details and create their profile.

All fields required manual entries for this iteration. This was not ideal, as it is prone to error input, but for the initial prototype it was sufficient. Each field was comprised of two tags, a label and an input box. The input box also contained a placeholder to prompt user input.

Upon clicking “Register” the data were sent from the client to the server via a HTTP “POST” request. A function in the “views.py” file was created to retrieve all the details sent by the HTTP “POST” request and store the details in local variables. The local variables storing the first name, last name, email address and password were used to create a “User” object. The rest of the local variables along with the “User” object were used to create an “EmployeeUser” object. The “User” object was utilised because it stored the password entered as a hash string which provides additional security.

A script tag introduced validation using JavaScript to ensure the “Password” and “Confirm Password” fields matched.

3.1.1.3 Log In (Requirement 2)

For the users to be able to log in, a HTML page was created, using a form tag. The form was made up of two inputs, one being the email address and the other being the password.

Once the “Log In” button is clicked, the two inputs were sent to the server via a HTTP “POST” request and a function created in the “views.py” file was then called. This function stored the inputs given to local variables and the “authenticate” function, which is part of the Django library, was used to authenticate the user. If the log in details are validated, then the user is logged in.

Two log in pages were implemented in this prototype, one for the Associate flow and the other for the Resourcer flow. This was a simple way to create and access the different views but was not the most efficient and user friendly way. The log in pages were accessible via the navigation bar of the home page. A “Register Here” button was added to the log in page to allow for new users to register.

3.1.1.4 Adding Project Roles (Requirement 4)

For the addition of available project roles, once again a HTML file was created with a form tag, to generate a form with labels and input boxes for the requesters to enter the role details and add the role to the available roles for employees to view. The fields included: the role and project title, the grade and skills required, the location, start and end dates.

Upon clicking “Add Role” the data was sent from the client to the server via a HTTP “POST” request. A function in the “views.py” file was created to retrieve all the details sent by the HTTP “POST” request and store the details in local variables. Then the variables are used to create the “ProjectRole” object using the “create()” function.

3.1.1.5 Viewing available Roles (Requirement 11)

To display to the employees the available roles, a function within the “views.py” file was created that queried the database for “ProjectRole” objects that have the same grade required and line of service as the employee currently logged in. The “User” object of the current logged in user was retrieved using the “request” function and that object was then used to query the database for the “EmployeeUser” object. Once that object was retrieved the grade and line of service fields were located and stored and then were used to query the “ProjectRoles” table for available roles.

The roles would be stored in a list and sent from the server to the client in a dictionary. JINJA was used to access all the different fields of the “ProjectRole” objects in the list and they would be displayed in a table container using a table tag in HTML.

3.1.2 Testing

Testing is one of the final steps in any application development. The two main goals of testing are to demonstrate to the developer and the client that the application meets the requirements proposed and to identify situations where the behaviour of the application is incorrect, unwanted or does not meet the specifications (Sommerville, 2011). These two goals lead to validation and verification testing. The difference between the two is that the validation testing focuses on whether the right product is built and the verification ensures that the application is built the right way and is functioning correctly (Boehm, 1979).

3.1.2.1 Unit Testing

To achieve the verification goal, unit tests had to be completed. Unit testing involves testing a single unit of code individually to confirm that it performs as it is supposed to (Olan, 2003). The designed tests should simulate the system’s expected use for verification testing (Sommervile, 2011).

A survey carried out by Per Runeson on 50 software companies researching the best practices of unit testing, identified that initially units should be as small as possible and they should be carried out independtently by the developers (2006).

The code was broken down into functions, each function implemented a single proposed feature. One unit of code to be tested would correspond to a single function. As the sole developer, the tests were executed individually. Directly after implementation, each unit was rigorously tested. Any issues found from the unit tests were fixed and once the functionality was at an acceptable level, it was ready for User Testing.

3.1.2.2 User Testing

To meet the validation goal user testing was needed. To show that the application built was the correct product, users had to test the application to confirm whether it met their needs.

Testers were requested from the client as they would be future users and they were provided test scripts to emulate real use. Five subjects were used for testing for each different type of role. This is supported by academic methodologies, for instance, Jakob Nielsen’s experiment on usability testing revealed that it is possible to find approximately 75% of usability issues and using more testers does not add a significant value (1994). This was a reasonable target because there is an additional cost associated with additional testers, there may be new flaws introduced by next iterations and finally because the biggest issues are most likely to be found during early in testing (Nielsen, 1994).

3.1.2.3 Feedback from User Tests

The user testing provided invaluable feedback from the employees. Several minor errors were spotted that were not found within unit testing as well as some adjustments that were suggested for a better user experience.

The non-functional bugs that were found mainly related to the user interface. For example, the PwC logo was missing at the top of the page on the deployed version of the web application. It was spotted that there was a spelling error on the Register page, where the placeholder had the text “Confir Password” rather than “Confirm Password”. There was another spelling mistake on the navigation bar, where it was spelled “Asscociate Login” instead of “Associate Login”.

It was noted that the pop-up boxes were not needed but it was explained that they would be used in the prototypes for debugging purposes.

In terms of usability and creating an overall better user experience, it was repeatedly suggested by the users that it would be an improvement if the Grade/Line of Service/Location fields in all the forms would be presented as dropdown boxes with options rather than text fields. This would be to eliminate user errors in typing, in addition to being user friendly and reducing time of form completion.

For the “Resourcer” users it was suggested that they should have the ability to leave certain fields blank because they do not need to have any of those fields attached to their profiles, since they are not resourced.

Users also suggested that it would be best that once they registered they should be directly logged in and redirected to the appropriate home page. When users failed to log in due to invalid credentials, they would prefer to be redirected to the appropriate log in page instead of the web home page so they can try logging in again in shorter time.

For the registration many users suggested for security reasons that passwords should have a stricter validation process. For the first prototype, users only had to create a password and confirm it but there was no further validation process or guidelines for a strong password. Users could create a one character password and that was seen as a security issue.

The log in system was described as confusing by multiple users, they preferred the idea of having a single log in page and being redirected appropriately given their grade rather than having to log in through different pages.

There were some positive comments on the overall idea and functionality. It was appreciated by the users that when the rest of the functionality would be implemented and the user experience would improve that this would be a very useful and efficient system. The users liked the overall ease of following the system through, being able to locate and apply for roles in quick fashion. Resourcers appreciated the speed up effect on the process of listing roles and finding applicants. The UI design although partial, had a positive effect on users, given its bright colours, matching the PwC theme and visibility of the text and buttons.

3.2 Prototype 2

3.2.1 Implementation

3.2.1.1 Improvements on Feedback

For the second sprint, development began on trying to improve and work on the first prototype, taking into consideration all the feedback gathered from the user testing. By addressing the feedback from the users, a baseline version of the application was created that the client would be content with its partial functionality.

The log in system was changed as it caused inconveniences for several users. The navigation bar was amended to include a single log in page. All users now would log in from the same page. JINJA was utilised to implement a conditional statement to decide which home page the user would be redirected to. JavaScript was used to dynamically redirect the user. Following a failed log in attempt, users would stay on the log in page.

The “Registration” and “Add Role” pages were altered to include dropdown boxes for the following fields: Grade, Location and Line of Service. This was done to eliminate risk of user error. In addition, following registration the users would be automatically logged in to reduce time spent on using the web application. For the registration page, the fields requested by the users, Skills and Languages, were made optional.

A minimum six character rule was added for the creation of a password as suggested by the testers, to improve the security of the system. JavaScript was utilised within the HTML file for the registration page. A script was used to compare the “Password” and “Confirm Password” text fields but further validation was added to check that the password entered by the user was a minimum of six characters. No further validation was added because security was not the main focus of this prototype.

An attempt was made to fix all the non-functional and design features, all the spelling mistakes were corrected. The rest of the design issues were not worked on due to the time restrictions of the project and the importance of functionality over design. This is because this web application is seen as a prototype.

3.2.1.2 Match Rating System (Requirement 6)

A match rating was implemented in this sprint, for the prototype a simple piece of code was introduced with logic that took into consideration the skills required for the role and the skills the applicant has and it compares the two lists and calculates the percentage of skills required which match the applicant’s skills. This differs from the proposed match rating in the design due to time restrictions. The percentage generated would be displayed to the applicant before applying and also it would be shown to the requester when viewing applicants. This system has potential and room for improved in future iteration as many other factors could be considered. For example, introducing availability review in conjunction with the current application Talent Link, adding certifications and security checks.

3.2.1.3 Applying to roles (Requirement 11)

A button was added for each available role listed to the employees, where they would be able to click on it to view more details about the role listed. For example, role description, skills required and the match rating of the employee to the role. Once they view all the details of the role, they would have the option to apply through a button at the bottom of the page. If they click the button, a record would be created in the “Applications” table and it will store the “EmployeeUser” object, the “ProjectRole” object and the match rating generated.

3.2.1.4 Viewing Applicants (Requirement 5)

Once employees have applied to a role, the requester would be able to view the applicants of the roles they have listed. This feature was implemented through the use of a query of the database and retrieving any “Application” objects that have the desired “ProjectRole” object stored. Once the applications are located then the “EmployeeUser” objects stored within those records are used to query the database and locate the details of the employees. Those employees with their details are displayed in a table format for the requester to view. They could also view the match rating of each individual for that role. Then they would have the option to view the applicant details in more depth by clicking the “View Applicant” button.

3.2.1.5 Edit Profile (Requirement 1)

As requested by the client, when discussing registration, it was very important that the employees would have the ability to update certain details on their record. An employee may get a promotion and need to change their grade, they may move location or they may change their line of service over their career at PwC. These fields have an impact on the roles they apply to and the abilities they have within the web application.

An edit profile page was added on the navigation bar of the web application. If a user selects the edit profile option on the navigation bar, the database is queried to find the “EmployeeUser” object of the currently logged in user. They are taken to a page, where their details are displayed along with editable input boxes and dropdown boxes. They can change their line of service, grade, location, skills and languages. Once they were happy with the changes the input was sent from the client to the server via a HTTP “POST” request. That record that was found earlier was then updated with the new details and the object was saved and updated in the database. Any changes required logging out of the app and then logging back in to update the user's view.

3.2.1.6 Clean UI design (Requirement 9)

The design was not a focus in this iteration, however an attempt was made to adapt it so that it is more user friendly. Text was aligned centrally and an introduction of more instructional text was added to enhance user interaction. The text font size was increased to improve overall visibility as well as the use of the PwC colour scheme to deliver a contrast.

3.2.1.7 PwC design (Requirement 10)

To match the PwC design scheme the colours used on the web page were Orange, White and Black. This colour scheme appeared in the background of the page as well as the buttons and containers in the HTML code. This colour scheme matched other internal software and web application, as well as their logo. The logo of the company was used throughout the web application and was it was imported into the "index.html" HTML file, which is inherited by all other templates.

3.2.2 Testing

3.2.2.1 Unit Testing

Unit tests were performed once again after the completion of development or improvement of each individual feature (unit). This was done to ensure that the features were implemented correctly as well as to ensure that the newly implemented features did not have an impact on the previous prototype and it did not introduce any bugs to the existing working web application.

3.2.2.2 User Testing

For the user testing, the same approach as prototype 1 was taken. The same testers were used because of their pre-existing familiarity with the system, which in turn would be able to provide more insightful feedback in a shorter period of time.

The test scripts were altered, the pre-existing functions were re-tested as part of regression testing to ensure that the new features did not break any of the previously functioning features. In addition, the previous functions required re-testing by the users to confirm that their feedback from the first prototype were addressed and if they were content with the updates. Additional tests were generated to test the newly added features, both proving that they are functioning to the client's needs as well as performing as expected.

3.2.2.3 Feedback from User Testing

Feedback generally improved on the second prototype from the testers. The usability was much improved and the features were further developed.

Some usability issues still appeared. For example, users could apply to the same role several times, which is unwanted. Users could flood requesters with applications for a specific role, possibly unknowingly because there was no marker to show that a user has already applied for a particular role.

It was made clear that users would like to be able to filter through roles themselves rather than having predetermined filters. As of the second prototype, users could only see available roles that matched their grade. However, as per the requirements, the requesters would like to have flexibility on the grade of the applicants. This could be possible with the addition of an extra dropdown box to add possibly multiple grades for a role and then having a manual filter on the roles listed page for the employees.

Furthermore, the usage of dropdown boxes could be improved to reduce confusions caused to the users. The dropdown boxes are prepopulated with the first option. If a user mistakenly forgets to choose from the dropdown box, it will take the first option as the answer. This was suggested to be altered with a blank spot or a “Select” option to remind the users to select a choice for that particular field.

When filling in the skills field it was suggested by some testers that, a set of pre-existing options could be added and the user can select from those. If a skill does not exist in the system that user can manually add it and expand the dataset of skills. This would also benefit the user experience and limit user errors. This in turn would be beneficial in terms of the match rating system because the skills are currently strings that are compared and if they are spelled differently they would not be taken into consideration.

Some minor spelling mistakes were identified throughout the web application. The PwC logo still was not showing on the page. The overall design was described as having a large room for improvement on the whole, the text could be cleaner, the overall layout could be improved and the pages could be filled with more content to emulate the real application. However, these comments were to be expected as the focus was on implementing functional requirements rather than non-functional requirements.

All the feedback and test scripts are stored within the Git repository. A link to the folder has been provided in the Appendix C.2 User Tests and Feedback section.

Chapter 4
Discussion

4.1 Conclusions

The proposed goal of the project was to develop a web application that would improve the overall resourcing process at PwC. Based on the development, a functional prototype has been created with great potential for further improvements.

The technologies chosen were appropriate given the scope of the project and the overall product produced in the end. The web application was successful in providing a proof of concept with partial functionality. The use of the Django framework and Python proved beneficial in development. Prototypes could be developed with sufficient functionality and relative ease within suitable time frames. With further knowledge of the Python coding language, major improvements can be made to the application.

The limitations mainly concern the integration of ProjectFinder with existing internal applications such as Talent Link, which proved to be a challenge. The lack of access to those applications meant that ProjectFinder, although a minimum viable product, could not emulate its real use within PwC.

There is space for improvement concerning the overall functionality of the final prototype, its security and design. However the end product was able to meet the majority of the requirements set and the expectations set at the beginning of the project.

4.2 Ideas for future work

A major improvement for the application would be to focus on the user interface more, improving the overall user experience the design of the system. More research into HTML and CSS could provide further knowledge in design and displaying of HTML web pages. This can also be extended to dynamically adjust views depending on the device the application is being used on. PwC provides laptops and mobile phones for the users, so it would be beneficial to the client if the application was usable on mobile phones to provide flexibility and make the process of applying and finding applicants more efficient and portable.

For the future of this web application, it is imperative that there is integration between ProjectFinder and other internal firm services such as Talent Link and Workday. They would provide further insight to employee details, skills, certifications, promotions and availability.

One way that integration would enhance the applications usability would be with having access to the employees availability and taking that into consideration when applying for roles. It would improve massively, possibly filtering automatically roles of which the duration matches their availability.

This integration would allow for a significant improvement to the match rating system. The match rating system at the moment only compares the skills acquired by the applicant and the skills required for the role. Further studies could be conducted to discover other beneficial factors for roles. One idea could be to implement a way of testing the applicants skills and providing different levels so that employees can showcase the level of their abilities in specific areas. In addition to that other details such as employee availability and location could aid in providing a more suitable match rating, as per Match Rating Flow design in Figure 3.

This can be extended to the use of Artificial Intelligence and Machine Learning enabling direct resourcing and find the most suitable employee for a role automatically using data mining and analysis. This could potentially change the way current resourcing processes work. In turn, this will require further business considerations such as change management principles and new target operating model design for the firm.

As this is an application concerned with the UK branch it is necessary that the architecture of the application is compliant with the Data Protection Act 2018 (UK Government, 2018). Measures will have to be taken to make sure that the how the data is obtained, stored and used align with the DPA 2018. Introduction to hashing algorithms, hashing the database and encrypting the source code could be improvements in the security of the app.

Furthermore, there can be a validating process for employees' grades and line of service. At the moment, employees have the ability to register by inputting and editing their details without any verification in place. However this process should be validated to ensure compliance with identity and access management principles (Thakur & Gaikwad, 2015).

Better authentication features, such as a requirement for stronger passwords for the users and validation on the PwC emails used to log in could be implemented to reduce unauthenticated access and improve the overall security of the app.

Based on these improvements, this web application can be developed at the enterprise level to support PwC's future architecture and potentially commercial offerings to their clients.

List of References

Amunategui Manuel and Roopaei, M. 2018. Introduction to Serverless Technologies. In *Monetizing Machine Learning: Quickly Turn Python ML Ideas into Web Applications on the Serverless Cloud*. Berkeley, CA: Apress, pp.1-37.

Asana. 2023. *What is a flowchart? Symbols and types explained*. [Online]. [Accessed 10 January 2023]. Available from: https://asana.com/resources/what-is-a-flowchart

Bhosale, S. T., Patil, M. T., & Patil, M. P. 2015. SQLite: Light Database System. *International Journal of Computer Science and Mobile Computing*. **44**(4), pp.882–885.

Boehm, B. W. 1979. Software Engineering: R&D Trends and defense needs. In P. Wegner (ed.), *Research Directions in Software Technology*. MIT Press, pp.1-9.

Dauzon, S., Bendoraitis, A., & Ravindran, A. 2016. *Django: Web Development with Python*. Birmingham: Packt Publishing Ltd.

Despa, M. L. 2014. Comparative Study on Software Development Methodologies. *Database Systems Journal*. **5**(3), pp.37–56.

Django. 2023. *Django documentation.* [Online]. [Accessed 12 February 2023]. Available from: https://docs.djangoproject.com/en/4.2/

Duckett, J., & Schlüter, J. 2011. *HTML & CSS*. Indianapolis, IN: John Wiley & Sons, Inc.

Figma. 2023. *UI Design Tool*. [Online]. [Accessed 19 December 2022]. Available from: https://www.figma.com/ui-design-tool/

Ghimire, D. 2020. *Comparative study on Python web frameworks: Flask and Django*. Bachelor’s thesis, Metropolia University of Applied Sciences.

UK Government. 2018. *Data Protection*. [Online]. [Accessed 22 April 2023]. Available From: https://www.gov.uk/data-protection

Grinberg, M. 2018. *Flask Web Development: Developing web applications with Python*. 2nd ed. Sebastopol, CA: O’Reilly Media Inc.

Holovaty, A., & Kaplan-Moss, J. 2009. *The Definitive Guide to Django: Web development done right*. Berkeley, CA: Apress.

International Organization for Standardization. 2019. *ISO 5807*. [Online]. [Accessed 12 January 2023]. Available from: https://www.iso.org/standard/11955.html

Jain, S., & Ingle, M. 2011. Software Security Requirements Gathering Instrument. (*IJACSA) International Journal of Advanced Computer Science and Applications.* **2**(7), pp.116-121

Jatana, N., Puri, S., Ahuja, M., Kathuria, I., & Gosain, D. 2012). A Survey and Comparison of Relational and Non-Relational Database. *International Journal of Engineering Research & Technology*. **1**(6), pp.1–5.

Kuhn, J. 2009. Decrypting the MoSCoW Analysis. *The workable, practical guide to Do IT Yourself.* **5**.

Lei, K., Ma, Y., & Tan, Z. 2014. Performance Comparison and Evaluation of Web Development Technologies in PHP, Python and Node.js. In: *17th International Conference on Computational Science and Engineering,* 2014, Chengdu, China. IEEE, pp.661-668.

Li Qing and Chen, Y.-L. 2009. Entity-Relationship Diagram. In: *Modeling and Analysis of Enterprise and Information Systems*. Springer, pp.125-139

Lloyd, D. (2009). *Evaluating human-centered approaches for geovisualization*. Doctoral dissertation, City University London.

Loeliger, J., & McCullough, M. 2012. *Version Control with Git: Powerful tools and techniques for collaborative software development*. 2nd ed. Sebastopol, CA: O'Reilly Media Inc.

Lucid. 2023. *Lucidchart*. [Online]. [Accessed 13 January 2023]. Available from: https://www.lucidchart.com/

Martin, R. C. 1999. Iterative and Incremental Development (IID). *Engineering notebook column.*

McManus, J. 2004. A stakeholder perspective within software engineering projects. In: *IEEE International Engineering Management Conference (IEEE Cat. No. 04CH37574).* IEEE, Vol. 2, pp.880–884.

Nielsen, J. 1994. Estimating the number of subjects needed for a thinking aloud test. *International Journal of Human Computer Studies*. **41**(3), pp.385–397.

Olan, M. 2003. Unit testing: test early, test often. *Journal of Computing Sciences in Colleges*. **19**(2), pp.319–328.

Python. 2023. *Python 3.10.11 documentation*. [Online]. [Accessed 30 January 2023]. Available from: https://docs.python.org/3.10/

Rasnacis, A., & Berzisa, S. 2017. Method for Adaptation and Implementation of Agile Project Management Methodology. *Procedia Computer Science.* **104**, pp.43–50.

Revesz, P. 2010. *Introduction to Databases*. Springer.

Roth, R. E., Hart, D., Mead, R., & Quinn, C. 2017. Wireframing for interactive & web-based geographic visualization: designing the NOAA Lake Level Viewer. *Cartography and Geographic Information Science*. **44**(4), pp.338–357.

Rover, D., Ullerich, C., Scheel, R., Wegter, J., & Whipple, C. 2014. Advantages of agile methodologies for software and product development in a capstone design project. In: *Frontiers in Education Conference (FIE) Proceedings*, October 2014, Madrid, Spain. IEEE, pp.1–9.

Runeson, P. 2006. A survey of unit testing practices. *IEEE Software*. **23**(4), pp.22–29.

Saeed, S., Jhanjhi, N. Z., Naqvi, M., & Humayun, M. 2019. Analysis of software development methodologies. *International Journal of Computing and Digital Systems*. **8**(5), pp.445–460.

Scanlan, D. A. 1989. Structured Flowcharts Outperform Pseudocode: An Experimental Comparison. *IEEE Software*. **6**(5), pp.28–36.

Schwaber, K. 1997. SCRUM Development Process. In: *Business Object Design and Implementation: OOPSLA'95 Workshop Proceedings, 16 October 1995, Austin, Texas*. Springer London, pp.117-134

Shyam, A., & Mukesh, N. 2020. A Django Based Educational Resource Sharing Website: Shreic. *Journal of Scientific Research*. **64**(1), pp.138–152.

Sommervile, I. 2011. *Software Engineering* (9th ed.). Pearson Education Inc.

Spinellis, D. 2005. Version Control Systems. *IEEE Software*, ***22***(5), pp.108–109.

Thakur, M. A., & Gaikwad, R. 2015. User identity and access management trends in it infrastructure- An overview. *International Conference on Pervasive Computing: Advance Communication Technology and Application for Society, (ICPC),* January 2015, Pune, India. ICPC, pp. 1-4

Tulchak, V. , L., & Marchuk, O. , A. 2016. *History of Python*. Doctoral dissertation, Vinnytsia National Technical University (VNTU).

Wu, M. C., & Sun, S. H. 2006. A project scheduling and staff assignment model considering learning effect. *International Journal of Advanced Manufacturing Technology*. **28**, pp.1190–1195.

Xinogalos, S. 2013. Using flowchart-based programming environments for simplifying programming and software engineering processes. *IEEE Global*

Engineering Education Conference (EDUCON), 2013, Berlin, Germany. IEEE, pp.1313–1322.

Young, R. R. 2002. Recommended Requirements Gathering Practices. *CrossTalk*, **15**(4), pp.9–12.

Yuill, S., Halpin, H. 2006. Python. [Online]. [Accessed 20 October 2022]. Available from: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=1f2ee3831eebfc97bfafd514ca2abb7e2c5c86bb

Glossary

Associate: Position at PwC

Director: High Position at PwC

Manager: High Position at PwC

Partner: Shareholder at PwC

Resourcer: Staff at PwC responsible for resourcing staff

Senior Associate: Position at PwC

Senior Manager: High Position at PwC

Talent Link: Internal CV web application used at PwC

Trainee: Position at PwC

Appendix A
Self-appraisal

A.1 Critical self-evaluation

The outcome of the project was positive because I developed a functioning prototype to support a large enterprise like PwC. Given that I did not have any previous experience in web and project development, this was a great learning journey as I was able to understand the multidimensional nature of an IT project in business.

There was a steep learning curve for both web application development and project management. There were many errors made, mainly due to the limitation of time. As part of my learning journey, I spent a lot of time learning new methodologies and new frameworks like Django. Hence, time was limited and continuous upskilling was required. This was a great experience for me as a developer and a professional. Technologies are continuously changing and it is very important that I am able to learn and adapt dynamically based on project requirements. I also learned the importance of planning and organising time to ensure the project was progressing in a timely manner in accordance to stakeholder requirements. This proved to be very critical, as there was a lack of time for testing and working on improvements. Time delays should have been taken into consideration when expecting external testers to test the application and provide feedback.

As a result of time restrictions, some features were not implemented and others were partially implemented. The complexity of the concept was greater than originally assumed. This was mainly due to security issues meaning that integration with internal software and applications was not feasible with the existing restrictions.

However the overall application definitely provided a successful proof of concept of a very useful idea for PwC. This could be advanced and pursued and for me as a developer to experience a real-life project.

A.2 Personal reflection and lessons learned

As a result of this project, I have improved my understanding in software project development. Especially, when it comes to understanding the steps within the software development lifecycle and the implications in terms of project and time management when defining a functioning product.

I was able to improve my skills around web development by researching and exploring different tools used and their functionality. This helped me become a better Web Developer and Software Developer as a whole.

I was able to put my existing skills from all my modules, such as Web Development, Databases, Software Engineering Principles and Software Engineering Project and the skills I have learnt into practice. I was able to discover my strengths and weaknesses along the way whilst working on tight deadlines and engaging with multiple stakeholders from PwC.

The experience has been very transformative and challenging at times. I can say with confidence that my capabilities as a developer and a professional have improved significantly. This project will benefit me in my future and my career as a Computer Scientist, furthering my expertise as a developer and project manager.

A.3 Legal, social, ethical and professional issues

A.3.1 Legal issues

The main legal issue faced through this project was the use of data. It was important that partially real data was used to emulate the use of the web application in actual use. However due to the lack of security, development taking place on non-company hardware and the application as well as the database being deployed to the cloud, unauthenticated access to real data would be very detrimental to the company.

The testers were suggested to use real data for their profile, however the project roles created were fake to not risk making public internal business and expose any of their clientele.

The testers names were not included in the test scripts they provided so that they remain anonymous.

In addition, all of the data stored in the database of the web application will be wiped and cleared upon completion of the project so the data will not be stored any longer than required in compliance with the Data Protection Act 2018.

A.3.2 Social issues

Social issues were avoided in this project because testers were kept anonymous and were selected by the client through volunteers. There were no biases in the selection of testers. As for the development team, it was compromised of a single person therefore there were no social issues within the project team.

A.3.3 Ethical issues

There was an ethical issue regarding the data that would be stored of the employees. It was made clear from the beginning of the project that no details regarding the employees age, gender or home address would be stored. Therefore only the employees base office location was requested and stored upon registration. To register also employees were not required to input their date of birth or gender.

A.3.4 Professional Issues

Professional issues could have arisen in the project due to the time and skill restriction, however these were explained to the client at the beginning of the project. It was evident that all of the requirements and goals may not be met at the end of this project. However, through constant and clear communication with the stakeholders professional issues were prevented.

Appendix B
External Materials

B.1 External Materials Used in Code

Bootstap 4.6 CSS and JavaScript for design:

https://getbootstrap.com/docs/4.6/

Libraries used:

https://docs.djangoproject.com/en/4.2/

https://www.django-rest-framework.org/

Appendix C

C.1 Wireframes

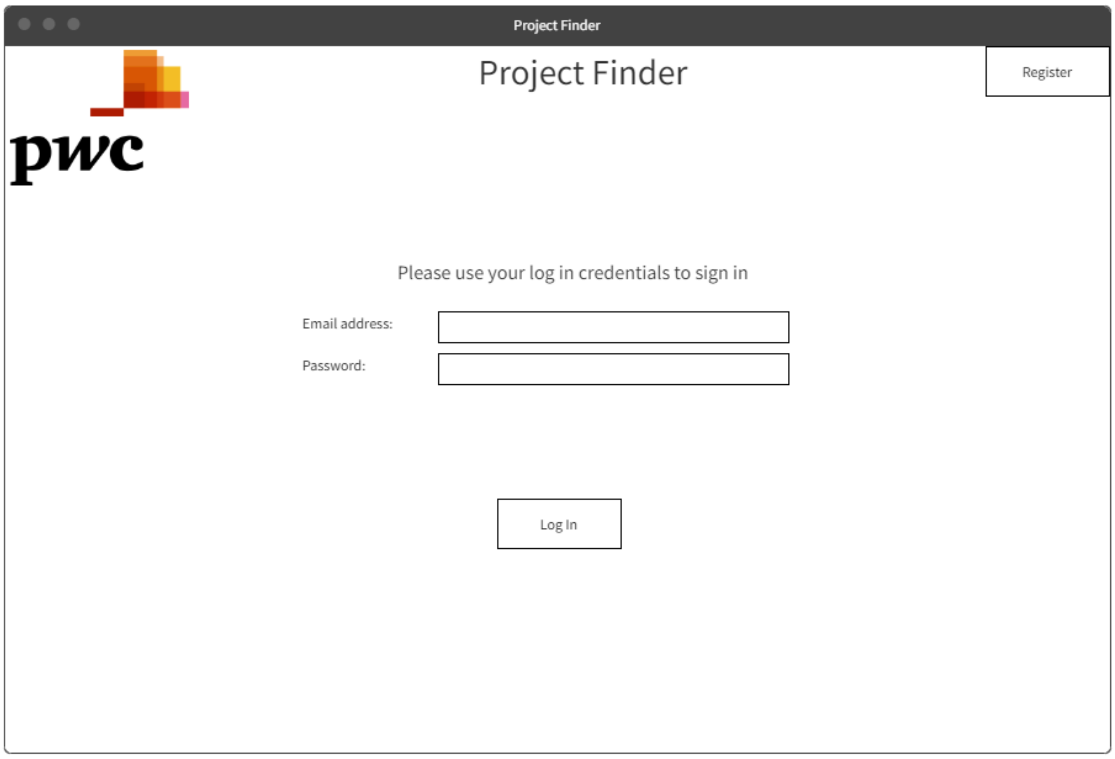


Figure C.2.1: Log in page Wireframe

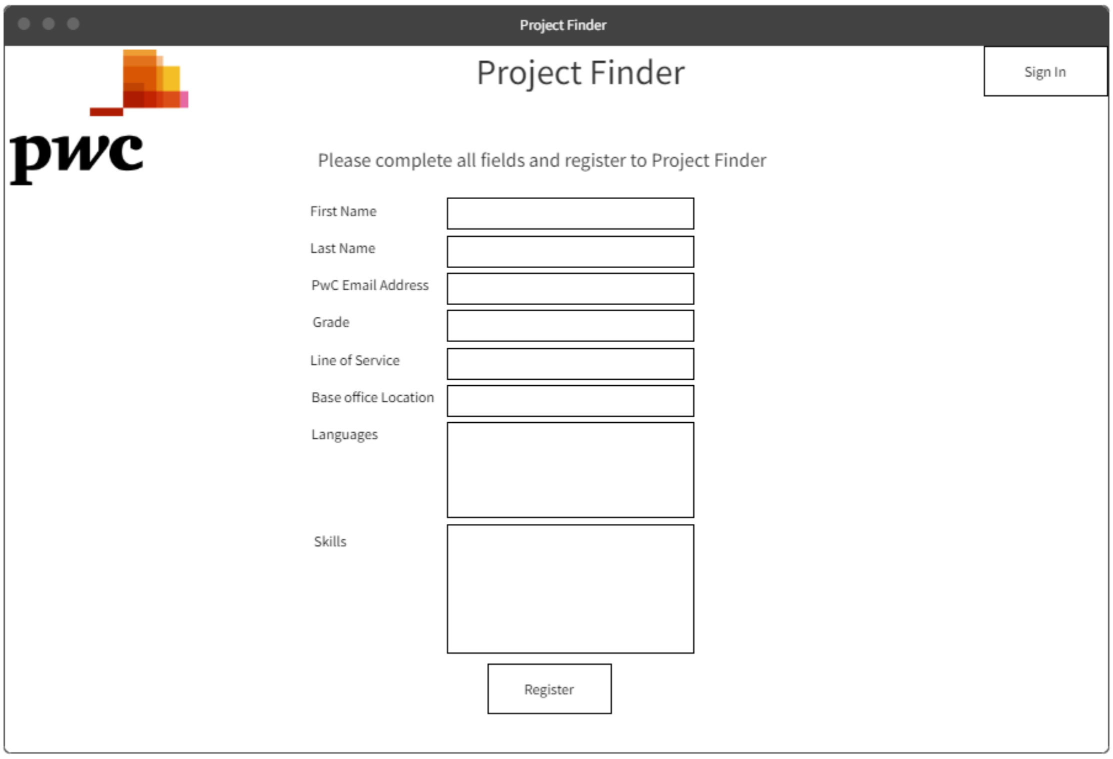


Figure C.2.2: Register page Wireframe

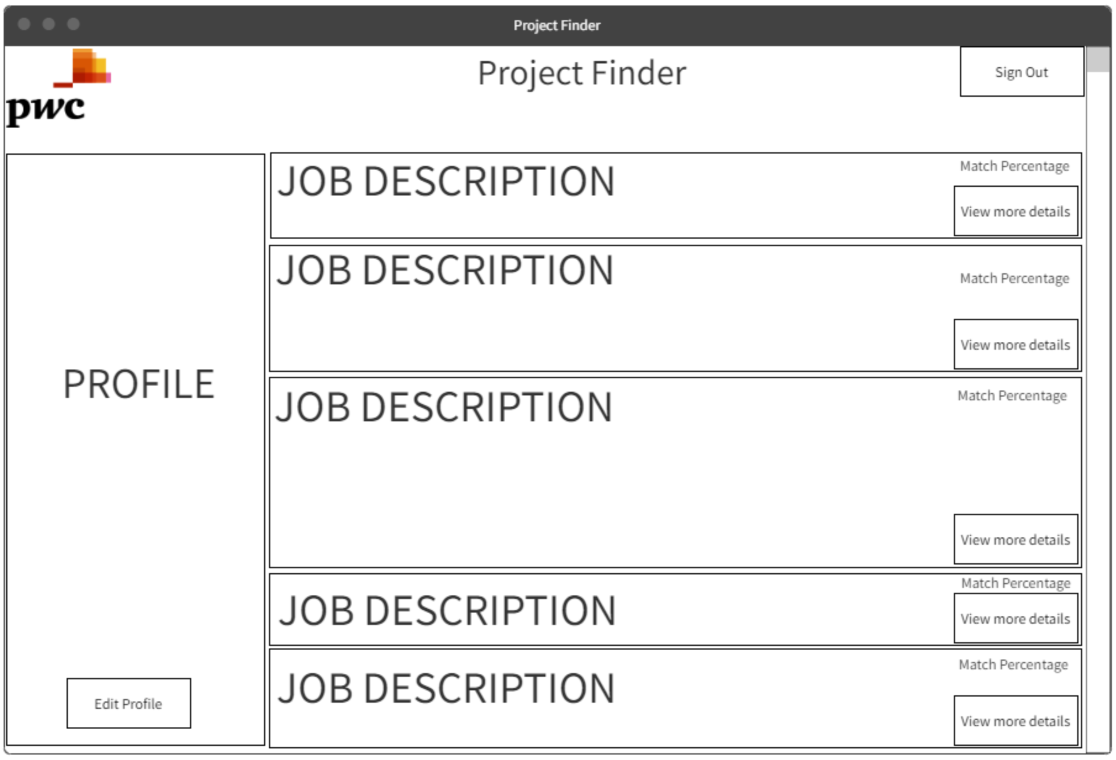


Figure C.2.3: Available project roles page for Trainees/Associates/Senior Associates Wireframe

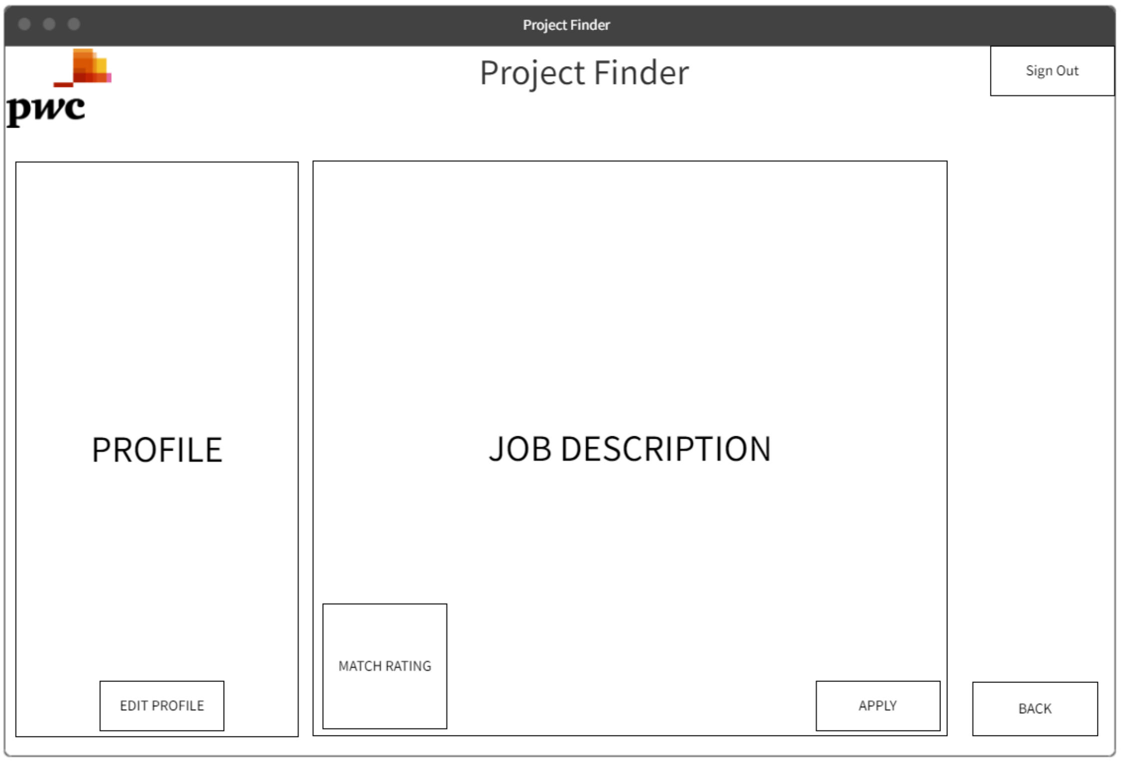


Figure C.2.4: Project role further details page for Trainees/Associates/Senior Associates Wireframe.

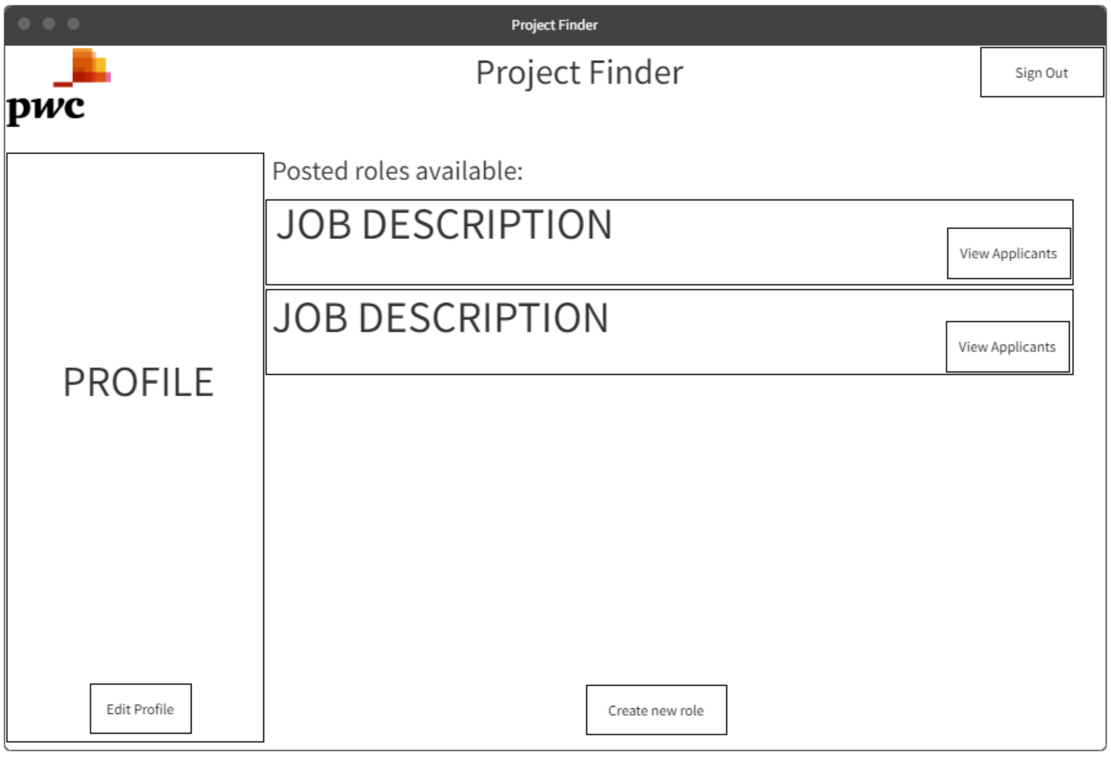


Figure C.2.5: Wireframe showing page with job roles listed by a requester.

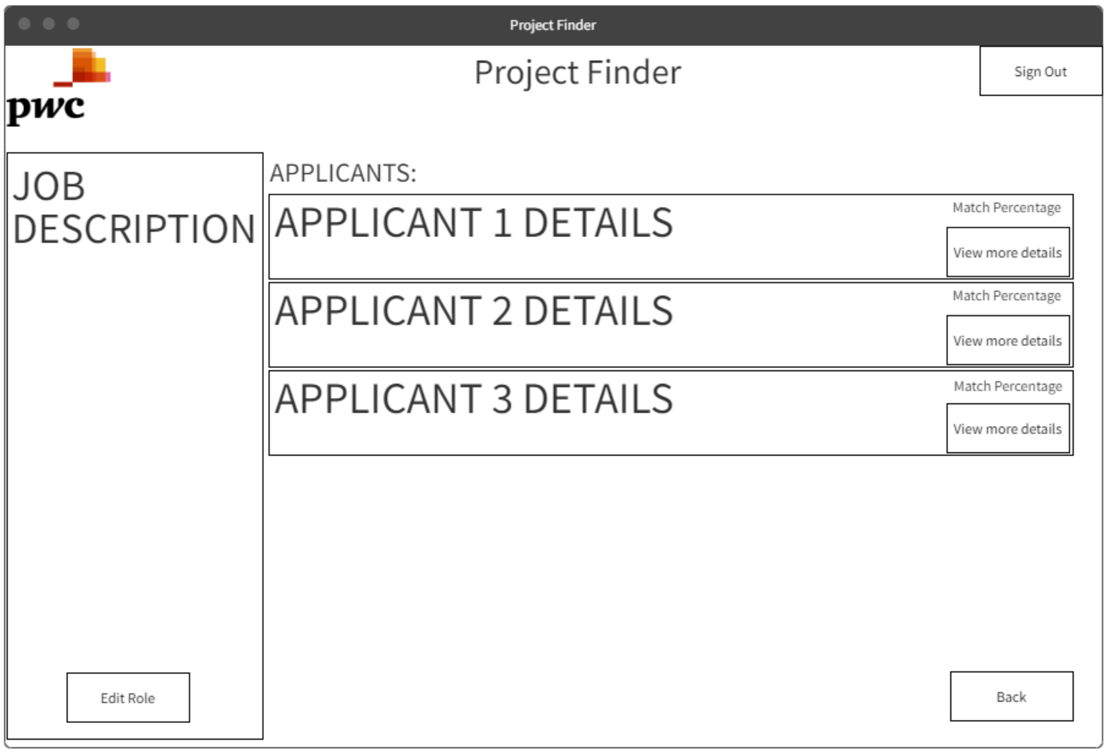


Figure C.2.6: Wireframe showing page with applicants that have applied for an available role.

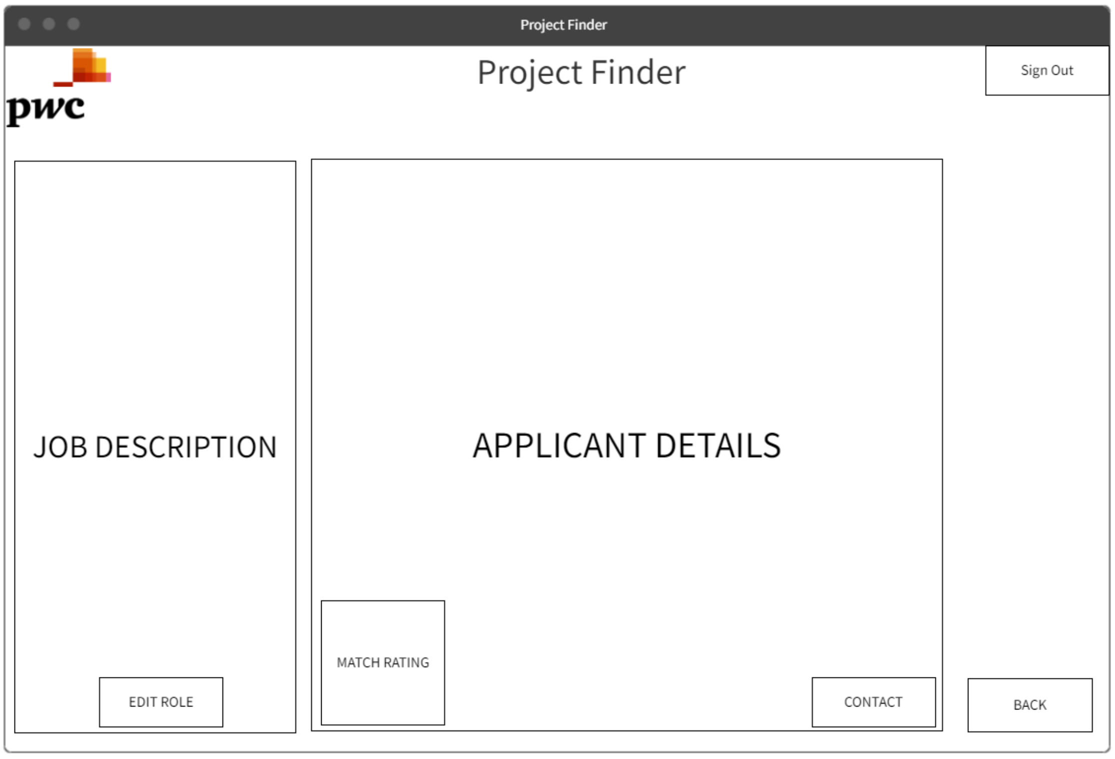
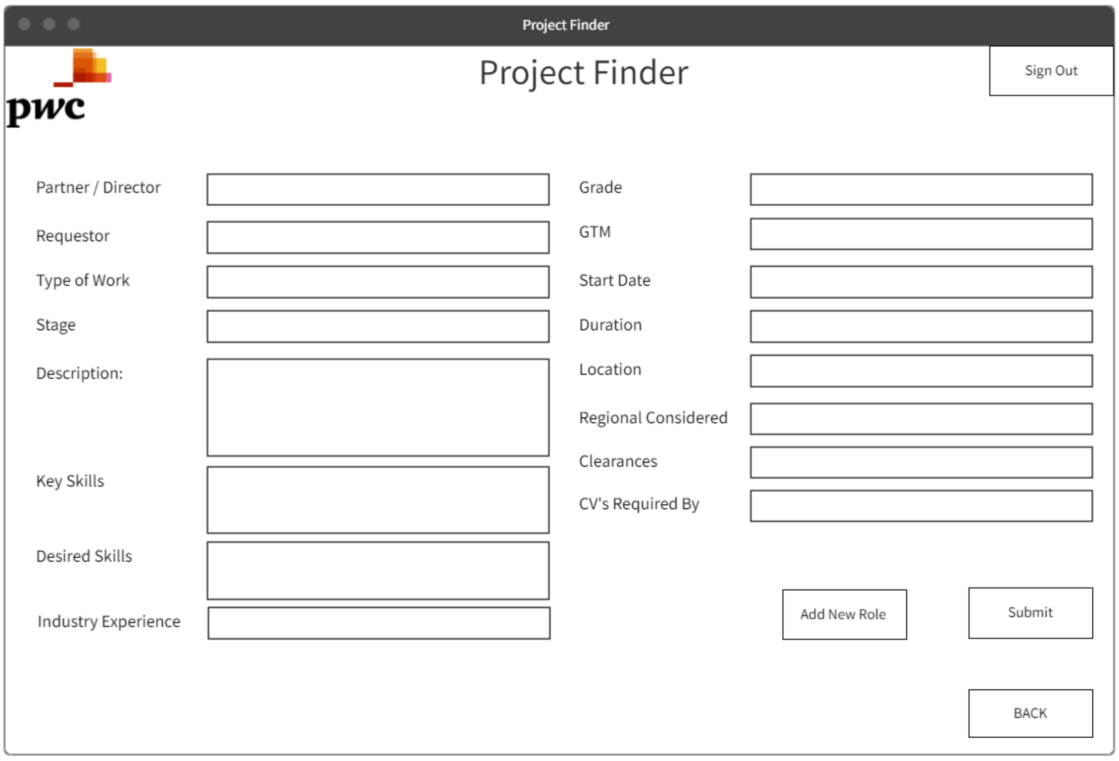


Figure C.2.7: Wireframe showing page with the applicant details that requester will view.



Partner / Director		Grade	
Requestor		GTM	
Type of Work		Start Date	
Stage		Duration	
Description:		Location	
		Regional Considered	
Key Skills		Clearances	
		CV's Required By	
Desired Skills			
Industry Experience			

Figure C.2.8: Wireframe showing add available role page.

C.2 Test Scripts and Feedback

Test Scripts with feedback responses can be found in GitHub repository of the code in the Test Scripts and Feedback Folder.

Link : Project-Finder-FYP/Test Scripts and Feedback at main · sc19ek/Project-Finder-FYP (github.com)