

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

SINGAPORE

**SCSE22-0477: Illuminating Singapore Careers with Data Driven
Knowledge Graph Exploration**

Supervisor: Asst Prof Long Cheng

Examiner: Dr Vidya Sudanshan

Submitted by: Isabelle Ng

Matriculation Number: U1922243C

A Final Year Project presented to Nanyang Technological University

in Partial Fulfilment of the requirements for

the Degree of Bachelor of Engineering (Computer Science)

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
ACADEMIC YEAR 2023/2024**

Abstract

In today's rapidly evolving job market, characterized by rapid technological advancements and disruptive business models, the pursuit of professional qualifications has become increasingly complex. While Online Professional Networks like LinkedIn and Learning Sites like Pluralsight provide platforms for job seekers to explore opportunities, they often lack clear insights into the key skills necessary to qualify for specific roles. This gap has highlighted the need for a more comprehensive and transparent system that offers personalized skill development and job recommendations. In a landscape where job seekers demand actionable insights and tangible pathways to career success, this project will present a transformative solution. By seamlessly integrating personalized skill development, job recommendations, and a structured guide for skill-pivoting, this initiative empowers individuals to navigate the complexities of the modern job market with clarity, confidence, and a practical understanding of how to adapt their existing skill sets effectively.

Acknowledgement

I would like to extend my sincere appreciation to Dr Long Cheng for his remarkable assistance and mentorship during this project. His accessibility and profound perspective greatly contributed to the project's development. I derived substantial knowledge from his remarks, and his constructive feedback played a pivotal role in the ongoing enhancement of this project.

Furthermore, I would like to thank all my friends and family for the emotional support.

Finally, I would like to express my gratitude to Nanyang Technological University for granting me the privilege to undertake this final year project.

Table of Contents

Abstract	2
Acknowledgement.....	3
Chapter 1	8
Introduction.....	8
1.1. Project Motivation	8
1.1.1. Rapidly Changing Job Market Landscape	8
1.1.2. Lack of Definitive Guidelines on Skills Required	8
1.1.3. Lack of Existing Products that Caters to Job Seekers Need	8
1.1.4. Lack of Singapore Related Knowledge Graph	9
1.2. Project Objectives	10
1.2.1. Intuitive Skills Path Mapping Tool:	10
1.2.2. Personalized Job Role Recommendations:	10
1.3. Project Scope	10
1.3.1. Creating a Singapore Skill-Job based Knowledge Graph	10
1.3.2. Developing a data-driven Web Application.....	10
1.4. Report Organization	11
Chapter 2	12
Literature Review.....	12
2.1. Existing Products in the Market	12
2.2. Knowledge Graph.....	14
2.2.1. What is Knowledge Graph?	14
2.2.2. Knowledge Representation	14
2.2.3. Advantages of Knowledge Graph Usage	14
2.3. Current Methodologies in Job Recommendation Model.....	15
2.3.1. Traditional Recommenders	15
2.3.2. Other Advanced Recommenders	17
2.3.3. Similarity Score Calculation	17
2.3.4. Recommendation Generation	18

2.4. Research Gaps and Limitations	18
Chapter 3	19
Requirement Analysts and Overview	19
3.1. Overall Design and Technologies Used	19
3.1.1. Next.js	19
3.1.2. Aura Neo4J	20
3.1.3. Dataset.....	20
3.2. Use Case	21
3.3. Functional Requirements.....	21
3.4. Non-Functional Requirements.....	24
3.4.1. Usability.....	24
3.4.2. Maintainability	24
3.4.3. Scalability	25
3.4.4. Reliability	25
3.4.5. Security	25
3.5. Project Resources Specification	25
3.5.1. Software Development Tools.....	25
3.5.2. Services	26
3.6. User Interface Design	26
3.7. Project Milestones	26
Chapter 4	28
Implementation	28
4.1. Backend.....	28
4.1.1. Database - Knowledge Graph Construction	28
4.1.2. Recommendation Model	35
4.1.3. Routing API Calls	38
4.2. Front-End	41
4.2.1. Front-End Implementation Details	41
4.2.2. Front End Walk Through.....	44
4.3. System Testing.....	53
4.3.1. Black Box Testing	53

4.3.2. Unit Testing	53
Chapter 5	55
DevOps	55
5.1. Tools Used for CI/CD	55
5.1.1. Vercel	55
5.1.2. Google Cloud and Google App Engine	56
Chapter 6	59
Conclusion and Future Works	59
6.1. Conclusion	59
6.2. Future Works	59
6.2.1. Improvements Knowledge Graph Construction	59
6.2.2. Future Enhancement: Bridging Skills Gap with Actionable Steps	61
6.2.3. Enhancing Testing and Deployment Pipeline	61
Bibliography	62

List of Tables

Table 1 Comparison of Job-Related Platforms	13
Table 2 Software Development Tools	25
Table 3 Services Used.....	26
Table 4 Project Milestones	27
Table 5 Properties of Nodes	32
Table 6 Relationships Created	34
Table 7 Collaborative Filtering User-Item-Skill Matrix.....	35
Table 8 Content Filtering Job-Skills Matrix.....	36
Table 9 List of Available APIs	40

List of Figures

Figure 1 Overall Design and Architecture	19
Figure 2 Use Case Diagram	21
Figure 3 Dialog Map.....	26
Figure 4 Original Job Skills Map.....	28
Figure 5 Original Dataset Technical Competencies Skill (TSC) Map	29
Figure 6 Entity Relation Diagram.....	30
Figure 7 Knowledge Graph Schema	30
Figure 8 Queries to Populate the Knowledge Graph	34
Figure 9 Loading Card	41
Figure 10 Error Card	41
Figure 11 File Structure Next.js Application	42
Figure 12 Landing Page	44
Figure 13 Login Page	45
Figure 14 Register Page.....	45
Figure 15 Forget Password Page.....	45
Figure 16 Dashboard Page (Logged In).....	46
Figure 17 Dashboard Page (Not Logged in or Learner Profile Not Created)	46
Figure 18 Unauthorized Access	46
Figure 19 Acquired Skills Collapsed Accordion View	47
Figure 20 Acquired Skills Expanded Accordion View	47
Figure 21 Dashboard Help Modal.....	48
Figure 22 Learner Profile Creation Part 1	49
Figure 23 Learner Profile Creation Part 2	49
Figure 24 Switch Learner Profile Page.....	50
Figure 25 Edit Learner Profile Page	50
Figure 26 Recommendation Page.....	51
Figure 27 Recommendation Help Modal.....	52
Figure 28 Setup Testing Environment.....	53
Figure 29 Using AAA Pattern for Testing	54
Figure 30 Output of Passed Testcases	54
Figure 31 Automatic Deployment on Vercel.....	55
Figure 32 GitHub Action Workflow	57
Figure 33 GitHub Actions Workflow Script	58

Chapter 1

Introduction

1.1. Project Motivation

1.1.1. Rapidly Changing Job Market Landscape

The contemporary job market is undergoing rapid and profound shifts, with the COVID-19 pandemic playing a significant role in reshaping perspectives on work and careers. The aftermath of the pandemic has highlighted that work is no longer confined to traditional office spaces, emphasizing the importance of well-being and opening doors to diverse work arrangements such as freelancing and gig work [1]. This changing landscape underscores the value of possessing both adaptable skills and core competencies specific to one's field. In this dynamic environment, staying competitive demands a robust skill. However, individuals often face challenges in recognizing these essential core competencies, which are pivotal not only for career advancement but also for successful job transitions

1.1.2. Lack of Definitive Guidelines on Skills Required

Today, career seekers often encounter difficulties in comprehending the expectations of the professional world and the essential components to construct their career portfolios. Job seekers often obtain information regarding available positions, necessary skills, and potential career trajectories through interactions with adept career advisors. Nevertheless, the availability of these advisors was limited. Furthermore, even these experts grappled with keeping up with the rapid and extensive changes unfolding in the job market. A report from the NTUC Youth TaskForce highlights that young people are looking for more resources to help their careers. This resource encompasses a range of needs, including heightened demand for career mentorship, increased support for upskilling, and improved access to definitive masterclasses aimed at comprehending key technical and generic competencies [1].

1.1.3. Lack of Existing Products that Caters to Job Seekers Need

While numerous job search platforms and career websites exist, they often provide limited insights into the specific skills required for career advancement. For instance, platforms like NodeFlair, LinkedIn, and Indeed lack the ability to guide individuals on the concrete skills needed to progress

at each stage of their careers or successfully transition laterally. For example, On NodeFlair, a developer might find job listings but not detailed guidance on the skills required to move from a junior to a senior developer role. Similarly, LinkedIn and Indeed primarily display job opportunities rather than offering a comprehensive skill development roadmap.

There also lack customization and personalization. For example, an experience profession with different past career backgrounds might have overlapping skill set and require less time to transition to another industry. Although LinkedIn has made efforts to assist with job transitions, this feature is tucked away in the LinkedIn Economic Graph Page, lacking user-friendly personalization and consideration of career stages. It generically suggests skill transitions based on similarity scores without accounting for customization and individual career levels.

1.1.4. Lack of Singapore Related Knowledge Graph

Numerous knowledge graphs have been constructed using the ESCO dataset, such as those focusing on career mapping and skill relationships in the European job market. However, an evident gap exists in the realm of knowledge graph development when it comes to the Singapore SkillsFuture dataset. Despite the prevalence of ESCO-based graphs, there are limited instances of similar initiatives dedicated to showcasing skill linkages and career pathways within Singapore's SkillsFuture ecosystem [2].

1.2. Project Objectives

The objective of the project is to build a data driven web application to allow users to acquire job skill related knowledge. It is an attempt to provide a standardize job skills knowledge by the official framework provided by the SkillsFuture Singapore.

1.2.1. Intuitive Skills Path Mapping Tool:

Users will have access to an intuitive web-based tool designed to map their career and skills journey. Through the tool, users can conduct a comprehensive analysis of their skill gaps by comparing their existing skill sets and qualifications against those necessary for their desired career roles. This analysis aims to illuminate areas of knowledge deficiency and provide users with a clearer understanding of their career advancement trajectory.

1.2.2. Personalized Job Role Recommendations:

The project entails the creation of a recommendation system that will propose suitable job positions tailored to individual users. This system will consider a user's skills, work experience, and preferred job sector to generate recommendations. The recommendation system will leverage on the baseline models of collaborative filtering and content-based filtering.

1.3. Project Scope

The scope of this project involves:

1.3.1. Creating a Singapore Skill-Job based Knowledge Graph

The Singapore SkillsFuture Framework is selected as the primary data source for this project. The project excludes integration with external job platforms, emphasizes the utilization of existing job skills data derived from the SkillsFuture Framework, and functions within the limitations of publicly available data and the presumed accessibility of SkillsFuture Framework data.

1.3.2. Developing a data-driven Web Application

The application will provide an intuitive career skills mapping tool for users to

- (1) Assess their skill gaps and qualifications
- (2) Recommend suitable job roles based on skills, experience, and desired sectors using collaborative and content-based filtering.

1.4. Report Organization

The structure of the report is as follows:

1. Literature Review
2. Requirement Analysis and Overview
3. System Implementation
4. DevOps
5. Conclusion and Future Works

The report begins with a literature review, providing information on the existing research surrounding the use and storage of knowledge graphs. This section will also explore prevalent recommendation system algorithms and offer a comparative overview of current market offerings. The subsequent segment delves into requirement analysis, offering insights into the project's design considerations, algorithmic choices, and defining critical milestones. In the system implementation section, a thorough walkthrough of the web application will be provided, emphasizing its main features. The DevOps section will focus on deployment strategies and methods to ensure seamless scalability and robustness of the application. The concluding section encapsulates the entire project, discussing outcomes and envisioning possible future enhancements.

Chapter 2

Literature Review

2.1. Existing Products in the Market

In the era of digital transformation, various platforms aim to assist users in the job-seeking journey, ensuring they are well-equipped with the necessary skills and knowledge. While these platforms cater to a global audience, there seems to be a gap in specialized solutions for the Singapore market. There's also a lack of specific for jobs skills mappings. The following is a breakdown of some noteworthy products in this domain:

Types	Features	Limitations	Charges
Online Professional Network			
LinkedIn	Largest professional networking platform globally, allows users to create profiles detailing their professional experiences, facilitates networking, job hunting, and knowledge sharing.	While it provides endorsements and skill listings, there's no robust mechanism to assess and indicate skill gaps. Skill endorsements can often be arbitrary and differ across people and jobs	Free basic version: charges for Premium features which include more detailed insights and better job recommendations.
NodeFlair	APAC based networking platform tailored to facilitate job search.	Consists of some information about what skills the market is looking for but not explicit. Similar to LinkedIn, it might allow for skill listing but not necessarily a detailed assessment or clear indication of skill gaps.	Free
Skills Assessment Platform			
Pluralsight	Offers technology skill assessments alongside its vast library of courses.	While it assesses skills, the primary focus is on providing	Subscription-based model only with

	Users can take assessments to gauge their proficiency in various tech skills.	courses. Direct feedback on career-related skill gaps or how skills align with specific job roles might be lacking.	varying plans for individuals and businesses.
Dedicated Career Development Platform			
MyCareer sFuture.sg	Career platform that might offer job listings, career pathway suggestions, and possibly training or course recommendations.	While it can guide users on potential career paths, there isn't a clear mechanism to point out specific skills users are missing.	Free
JobStreet's Career Enhancement tool	Offers tools to help enhance careers, such as CV builders, career path insights, and possibly training recommendations.	While it aids in career development, it might not provide clear, detailed feedback on what skills a user lacks or needs to improve.	Basic job-seeking features are usually free, with additional charges for premium services.

Table 1 Comparison of Job-Related Platforms

While several platforms cater to the Singapore job market, most have a narrow focus, either highlighting job listings or company details. We lack comprehensive tool that pinpoints and addresses skill gaps in users. It's vital for individuals not just to know their strengths but also to identify and rectify their weaknesses. Such insights would better equip them for a rapidly changing job landscape. Additionally, there's a noticeable absence of tailored guidance for navigating Singapore's unique job market and mentorship opportunities. The subscription-based pricing of many platforms can also limit their accessibility to a broader audience.

2.2. Knowledge Graph

Introduced by Google in 2012, the knowledge graph revolutionized information representation, especially in intelligent search engines¹. Its swift adoption spanned academia and industries, where it had spurred major software companies like Microsoft, Amazon, and LinkedIn to create their own knowledge graphs. Microsoft's "Microsoft Satori"² enhances Bing search, Amazon's Product Graph³ optimizes product recommendations, and LinkedIn's Economic Graph⁴ harnesses career trajectories and expertise. These instances highlight the broad impact of knowledge graphs beyond their inception.

2.2.1. What is Knowledge Graph?

Hogan and colleagues characterized knowledge graphs (KGs) as data graphs designed to amass and communicate real-world insights and their interrelationships. Within these graphs, nodes symbolize significant entities, while edges denote varied potential connections between these entities. [3].

2.2.2. Knowledge Representation

Its fundamental component consists of the structure known as resource description framework (RDF) in an 'entity-relationship-entity' format, alongside the entity itself and its associated attribute-value pairs. These elements are interconnected through relationships, resulting in the formation of a network comprising knowledge structures [4].

2.2.3. Advantages of Knowledge Graph Usage

Knowledge graphs offers several advantages in terms of enhancing information representation and data utilization. Knowledge graphs capture relationships between entities, enabling a more holistic and contextual understanding of data. It's much more human readable and easier to visualise.

The application of Knowledge Graphs (KGs) to employability-related inquiries has been extensively studied and employed. There are numerous instances where KGs have been utilized to address questions related to employability. For instance, in the realm of (1) **job**

¹ https://en.wikipedia.org/wiki/Knowledge_Graph

² <https://www.cnet.com/news/microsofts-bing-seeks-enlightenment-with-satori/>

³ <https://www.aboutamazon.com/news/innovation-at-amazon/making-search-easier>

⁴ <https://economicgraph.linkedin.com>

recommendation and construction, KGs have been employed for constructing the graphs using Joint BPR, graph embedding techniques have been integrated with Joint Margin for recommending suitable jobs. Additionally, KGs have been harnessed to create a graph for job transitions, proposing algorithms for node embeddings, and enhancing job recommendations through methods like Skipgram. Within the **(2) talent intelligence and skill matching category**, KGs have been utilized to aggregate job titles using collective multi-representation learning and predict links using techniques such as link prediction. These examples underscore the diversity of applications, showcasing how KGs play a pivotal role in answering employability-related questions [5]. Integrating KGs into employability-related questions in this use case is particularly fitting, as these queries often necessitate the creation of a knowledge graph.

The inherent characteristics of the connection established among the entities within the knowledge graph render it an essential instrument for examining the correlation between occupations and competencies. This approach allows for a structured representation of complex relationships between various elements, like jobs, skills, competencies and talents. Notably, such a graph is inherently extensible, accommodating the integration of new information and facilitating in-depth analysis of relationships over time. This adaptability is crucial in the ever-evolving job market landscape, where insights can emerge, and trends can change rapidly. Consequently, the use of KGs not only enhances the accuracy and depth of responses but also lays the foundation for continuous exploration and refinement of employability-related insights.

2.3. Current Methodologies in Job Recommendation Model

2.3.1. Traditional Recommenders

2.3.1.1. Collaborative Based Recommenders

There are two primary algorithms in collaborative filtering mainly:

(1) User-User Collaborative Filtering (Memory Based CF)

User-user collaborative filtering is a method wherein the system identifies users who share similar preferences to a particular user. For this collaborative filtering approach to be effective, there needs to be interaction data from multiple users across various items to create a user-item matrix [9].

The primary step in this method involves constructing the user-item matrix and then determining the similarity between users. By examining the patterns and preferences of users who have similar preference, the system can make predictions about items

that the current user might find appealing. The principle behind this technique is that if users have had similar interactions or preferences historically, they will likely continue to have shared interests.

One of the prominent challenges with user-user collaborative filtering is data sparsity. Often, in many practical contexts, users might only engage with or rate a tiny subset of the plethora of available items, leading to a predominantly sparse matrix. This sparsity can pose significant challenges in pinpointing users with overlapping preferences. Moreover, if there aren't enough users or sufficient interaction data, it further exacerbates the data sparsity problem, making it even more challenging to provide accurate and meaningful recommendations – also often known as the cold start problem [7] [8].

(2) Item-Item Collaborative Filtering

Item-item collaborative filtering is a method within the realm of memory-based collaborative filtering, focused on the determination of similarities between items rather than between users. The idea is that if a user has interacted with or shown an inclination towards an item in a particular manner, they're likely to have a similar level of interest in other items viewed as similar by the system [10]. For instance, if users A, B, and C all liked both items X and Y, items X and Y are similar. Recommendations are then made by identifying items similar to those the user has interacted with, and suggesting other items perceived as similar.

2.3.1.2. Content Based Recommenders

In the content-based approach, users and items are characterized based on item content. In contrast to the Collaborative Filtering approach, which derives user and item representations from user-item interactions. The objective of content-based recommendations is to propose items that are presently available and align closely with what a user, such as prospective job seekers are searching for [9].

The features used for filtering can vary, encompassing a user's background (e.g., past, and current jobs), skill proficiency levels, and even intricate textual features like job descriptions and associated tasks [6].

For content filtering, features can be derived in a variety of ways. They can be as direct as quantitative metrics (Binary, Nominal, Numerical) or involve sophisticated Natural Language Processing techniques such as term frequency-inverse document frequency

(TF-IDF) and word embeddings to understand pertinent details from textual descriptions of items [21].

2.3.2. Other Advanced Recommenders

Most advance recommenders are hybrid recommenders which uses a combination of algorithms above with combinations of other techniques such as deep learning or reinforcement learning or additional data structures like knowledge graph/graphs to further improve the results of the recommendations. [11]

2.3.3. Similarity Score Calculation

The likeness between items or user profiles is assessed to produce an initial list of potential recommendations for the user. Different metrics, such as Cosine similarity, Euclidean distance, Spearman Rank or Jaccard similarity, can be employed to measure this similarity. Typically, cosine similarity is the preferred choice because of its clear interpretation, resilience to data sparsity, and independence from scale [12].

2.3.3.1. Cosine Similarity

Cosine similarity measures the similarity between two n-dimensional vectors by evaluating the angle between them in a vector space. In the context of Recommendation Systems (RS), items (or users) are treated as these n-dimensional vectors. The closer the angle is to zero, the more similar the items (or users) are to each other. It ignores the magnitude. In the context of job recommenders, this ensures that even if two jobs rate skills on different scales, say 1-5 and 1-10, the similarity between a user's skill set and both jobs remains consistent. Despite the scale differences, the cosine similarity focuses on the direction of the vectors. This approach is beneficial, especially when items, though rated differently, can still be considered similarly based on their orientation [22].

$$similarity = \cos \theta = \frac{u \cdot v}{|u| |v|} = \frac{\sum_{i=1}^n u_i v_i}{\sqrt{\sum_{i=1}^n u_i^2} \sqrt{\sum_{i=1}^n v_i^2}}$$

The cosine similarity is determined by the dot product of two vectors in the numerator, which represents the combined magnitude of the vectors and the cosine of the angle between them. The denominator computes the magnitude of individual vectors, u and v . Thus, cosine similarity is essentially the ratio of the dot product to the product of their magnitudes. The resulting values lie between 0 and 1, with 1 indicating identical vectors and 0 signifying orthogonal vectors.

2.3.4. Recommendation Generation

In both content and collaborative filtering, for an item or user to be considered for recommendation, its similarity score must surpass a predefined threshold. Once these scores are determined, the recommendations are curated by ordering the items from the highest to the lowest similarity score and the top N recommendations are chosen. Additional factors, such as item popularity, recency, or diversity, may further refine the ranking to ensure a more nuanced recommendation [10].

2.4. Research Gaps and Limitations

Within the scope of this project, the primary objective is to develop an effective system for skill identification using a foundational knowledge graph. This knowledge graph will act as a structured database, capturing and representing relationships between various skills, professions, tools, and concepts. By doing so, it will facilitate the identification of pivotal skills required for various job roles.

Leveraging the knowledge graph, the system can pinpoint not only the fundamental skills associated with a particular job but also understand the interconnectedness of these skills with other relevant domains. This holistic view provided by the KG can guide users to comprehend the broader skill landscape, enabling them to make informed decisions about skill acquisition and career progression.

While the recommendation system is a valuable addition. This project will primarily focus on building a foundational knowledge graph for skill identification and integrating the two baseline algorithms (Content, Collaborative) utilized in Job Recommendation models. Further details on how the algorithms are implemented will be highlighted in [Section 4.1.3](#). In the future, there's potential to adopt more advanced recommendation techniques leveraging the constructed knowledge graph. It's worth noting that numerous job boards and existing products already feature comprehensive recommendation systems directly linked to job opportunities and thus shall not be the focus of this project.

Chapter 3

Requirement Analysts and Overview

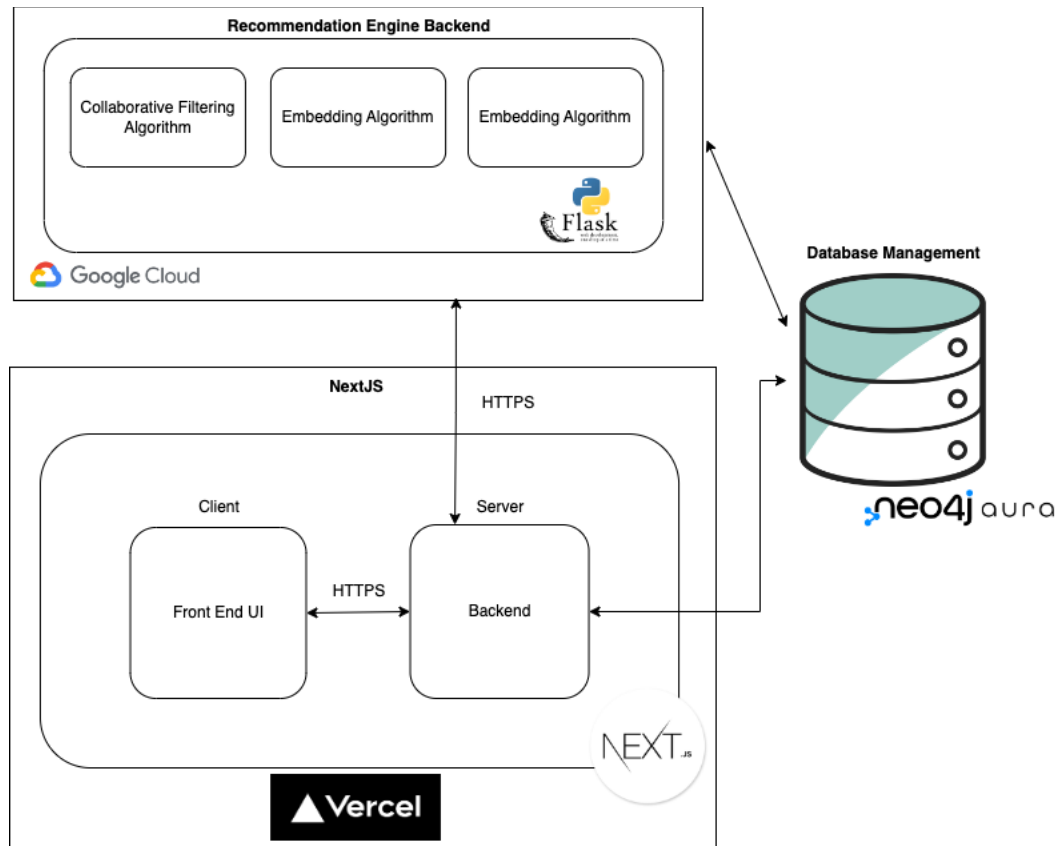


Figure 1 Overall Design and Architecture

3.1. Overall Design and Technologies Used

3.1.1. Next.js

Next.js was chosen as the main framework for development as it is an all-encompassing framework, adept at handling both the visual aspect of websites and backend functionalities. This dual capacity negates the necessity for multiple systems. For context, when one uses React.js, there's often a need for another backend mechanism like Express.js to manage APIs. With Next.js, this becomes a unified process. Next.js seamlessly support for automatic APIs. Where other platforms, such as Express, demand a separate server configuration for API interactions,

Next.js offers a more integrated approach. Through the '/api' structure, developers can readily set up serverless procedures, thus making a typically complex task more straightforward.

A primary incentive for selecting Next.js for full stack projects is its inherent ability for server-side rendering. While react mainly centers on client-side rendering—leading to slower initial page loads and potential SEO issues—Next.js offers a straightforward way to adopt server-side rendering. By doing so, pages are pre-rendered on the server and delivered to the client as HTML, ensuring faster load times and improved SEO, which ultimately boosts performance and online visibility [23].

In essence, selecting Next.js as the primary web development framework encapsulates the strengths of React.js, but with added efficiency and a user-centric approach.

Design Library

In this project MaterialUI was chosen as the design library. It is a popular React UI framework that allows for versatility and robust component library. The comprehensive set of pre-design components allows developers to quickly assemble the user interface without starting from scratch.

3.1.2. Aura Neo4J

Neo4j Aura was selected as our primary database system because it offers a hassle-free, fully managed cloud service, eliminating the complexities of self-hosting and maintaining traditional Neo4j knowledge graphs [24]. Our application aims to pinpoint discrepancies in user skills and suggest suitable job roles. Leveraging a graph-based model aids in effectively mapping the relationships among skills, users, and job roles. Moreover, Aura's advanced querying using the Cypher language enables a deeper understanding of these connections, streamlining the process of identifying skill gaps and providing precise job recommendations based on complex data patterns.

3.1.3. Dataset

The SkillsFuture dataset is a primary data source utilized for associating job roles with specific skills and their proficiency levels, as well as detailing job descriptions. This information was accessible via the SkillsFuture API.

3.2. Use Case

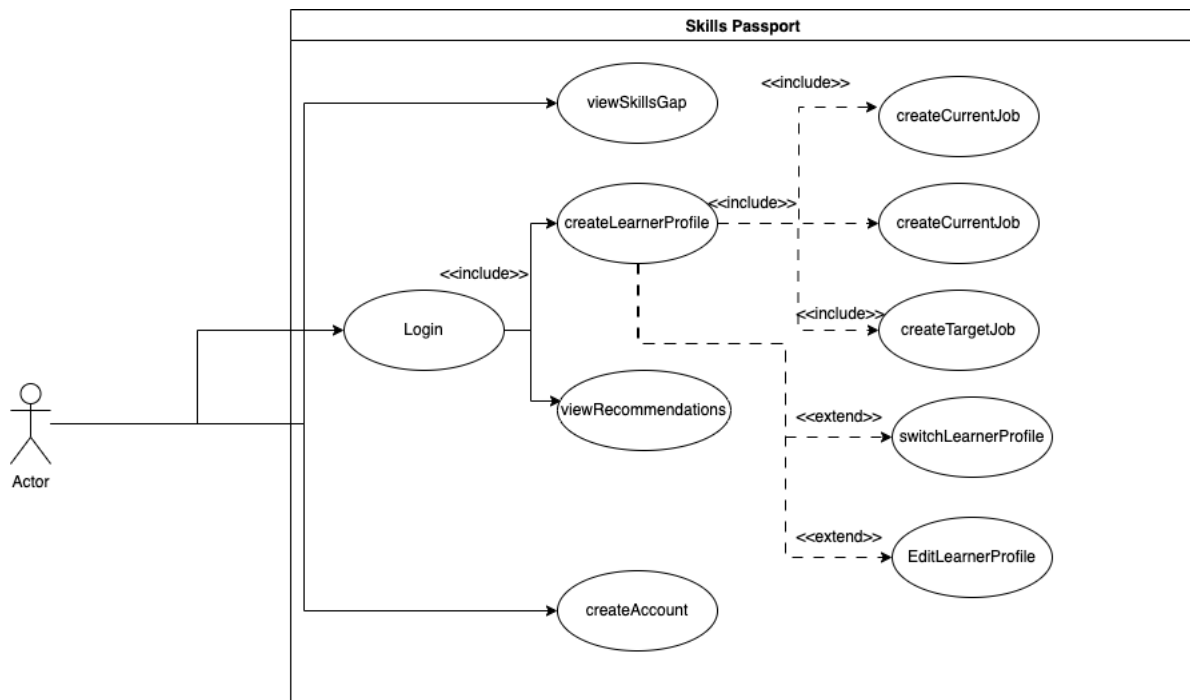


Figure 2 Use Case Diagram

An initial overview diagram was created to visually represent the system's necessary operations and use cases.

3.3. Functional Requirements

Functional requirements provide a clear blueprint for system development, preventing scope creep. They ensure user needs are addressed, streamline future modifications, and contribute to a more successful user-centric product.

3.3.1. Front End

3.3.1.1. Account Module

1. The system must allow the user to login.
 - (a) The user prompt login entry for all users.
 - a. The system must provide an input field for the user to input his/her email.
 - b. The system must provide an input field for the user to input his/her password.
 - (b) The system must be able to validate the credentials entered by the user.
 - a. The system must validate the email field is filled and password during login.

- b. The system shall provide validation checks to ensure both email and password fields are populated before proceeding.
- (c) The system must be able to cross verify the user credentials against those stored in the database.
 - a. The system should provide immediate feedback if login fails, informing the user of potential issues (e.g., incorrect password, non-existing account).
- (d) The user password should be stored securely (i.e. hashed) in the database.
- 2. The system must allow the user to create an account.
 - (a) The system must prompt the user to create his/her credentials during registration.
 - a. The system must prompt the user for his/her name, email, and password.
 - b. The system must ensure proper validation checks are in place.
 - i. The system must ensure the email is in an appropriate email format.
 - ii. The system must ensure the password is at least 8 characters long.
 - iii. The system shall provide validation checks to ensure all fields are populated before proceeding and provide feedback if not.
 - c. Upon successful account creation, the system shall notify the user and redirects the user to the login page.
 - (b) The system must allow the user to log out.
 - a. Upon logging out the system shall redirect the user to the landing page.
 - (c) The system must allow the user to change their password.
 - (a) The user must be able to access the forgot password page from the login page using a 'Forgot Password' button.
 - (b) The user must be able to log in with their new password.

3.3.1.2. Dashboard Module (My Skills)

- (a) The Dashboard Module shall consists of the Learner Profile Module as well as the Acquired Skills module.
- (b) The user shall have easy access to the navigation bar to navigate back to the dashboard module and the recommendation module.
- (c) The dashboard page shall provide a popup to guide the user in understanding how to use the application.

3.3.1.2.1. Acquired Skills Module

- (a) The skill gap identification and user's skill proficiency comparison against the database should be displayed in a visual format (e.g., stepper).
- (b) If the user's skill proficiency is greater than or equal to the job proficiency:

- a. All steps up to the job proficiency level shall be displayed in green.
- (c) If the user's skill proficiency is less than the job proficiency:
 - a. Steps representing the skills that the user has, up to their proficiency level, shall be displayed in green.
 - b. The subsequent step, which represents the minimum proficiency level required for the job but not met by the user, shall be displayed in red. Any steps beyond the user's proficiency and up to the job's requirement should just be displayed as grey.
- (d) The user shall be able to click on each proficiency level and view the description of the knowledge and the ability that is required at that level.

3.3.1.2.2. Learner Profile Module

- (a) The user shall be able to create learner profile(s)
 - a. Users must have the capability to enter their past employment positions, their current role, and the job they aspire to enter.
 - b. Based on the user's employment history, the system will estimate the user's skill proficiency for the desired job role.
 - i. The system will showcase an anticipated skill proficiency level for the user. Users should have the flexibility to modify this estimation based on their personal assessment.
 - c. After users finalize their skill proficiency modifications, the system should consolidate this information and store it as a unique learner profile.
- (b) The user shall be able to switch between learn profile(s) created.
 - a. The system shall support the display of multiple learner profiles, showcasing each individual's employment history.
 - b. Each learner profile, along with its corresponding employment history, shall be presented in a structured tabular format for easy viewing and comparison.
 - c. Users must have the capability to seamlessly navigate between different learner profiles within the table.
 - d. An intuitive interface should be provided to allow users to quickly switch and view details of each specific learner profile and its employment history without any hindrance.
 - e. Upon selecting the "back" option, the system shall immediately redirect the user to the Dashboard page with the newly updated learner profile.

3.3.1.2.3. Recommendations System Module

- (a) The user shall be able to view the top 10 recommended jobs by selecting the type of recommendation algorithm – collaborative, content or embedding.
- (b) There shall be a popup available for the user to understand how the different types of algorithms works and guide the user in his/her usage.

3.3.2. Backend

3.3.2.1. Data Security

- (a) All routes that require and queries user private information shall be protected.

3.3.2.2. Response Efficiency

- (a) The software must handle and revert to every request (POST, GET, DELETE, PUT) within a 15-second window.
- (b) Each response should deliver a JSON payload accompanied by its relevant status code.

3.3.2.3. Front-end Coordination

- (a) Although the software is expected to work cohesively with the front-end, it is essential that the two remain distinct entities.

3.4. Non-Functional Requirements

3.4.1. Usability

1. The system's user interface should prioritize intuitiveness, ensuring that users receive timely feedback after each action. This can be realized by adhering to a streamlined page layout, straightforward navigation, and by integrating help buttons or tooltips to guide users.
2. To maintain a user-centric design, the system interface should remain neat and free from distractions. Adopting [Ben Schneiderman's eight golden rules](#) of design will significantly enhance the user experience.

3.4.2. Maintainability

1. For enduring system operation, it's imperative to generate comprehensive documentation, preferably during the development phase or immediately upon its conclusion.
2. The system's architecture should be clear and organized, emphasizing the use of reusable components, and maintaining a consistent, understandable naming convention.

3. Configuration settings and sensitive information should be managed using environment variables. This allows for flexibility across different deployment environments without altering the codebase, simplifying the maintenance process.

3.4.3. Scalability

1. As user demand grows, the system should seamlessly scale to meet these needs. Utilizing cloud technologies, especially platforms like Vercel and Google Cloud, will be instrumental as these platforms offer dynamic scaling based on traffic patterns.
2. Flexibility is key when it comes to algorithmic improvements. Incorporating the strategy design pattern in the backend and a modular component structure in the frontend will ensure that recommendation algorithms (whether new or integrated) can be efficiently modified or implemented.

3.4.4. Reliability

1. Rapid deployment of code alterations or feature updates is crucial. Leveraging deployment platforms like Vercel can significantly streamline this process.
2. The system should rigorously test user input to identify and appropriately handle errors.

3.4.5. Security

1. Given the sensitivity of user job data, it's vital to prioritize security. All data transmissions should be secured using HTTPS. Additionally, using JWT tokens for authentication and bcrypt for hashing will further safeguard user information.
2. There should be secure storage of information like API keys, database credentials and other secrets. This can be achieved with the use of environment variables.

3.5. Project Resources Specification

3.5.1. Software Development Tools

Tool	Description
Visual Studio Code	Project Code Editor
Neo4J Aura	Online Database Management Tool for Data Storage, data exploration and management
Github and Git	Version Control Tool
Postman	User Friendly Platform for Building, Testing and documenting API

Table 2 Software Development Tools

3.5.2. Services

Tool	Description
SkillsFuture Developer Portal	For fetching Skills, Jobs, Job Task, Job Description Data

Table 3 Services Used

3.6. User Interface Design

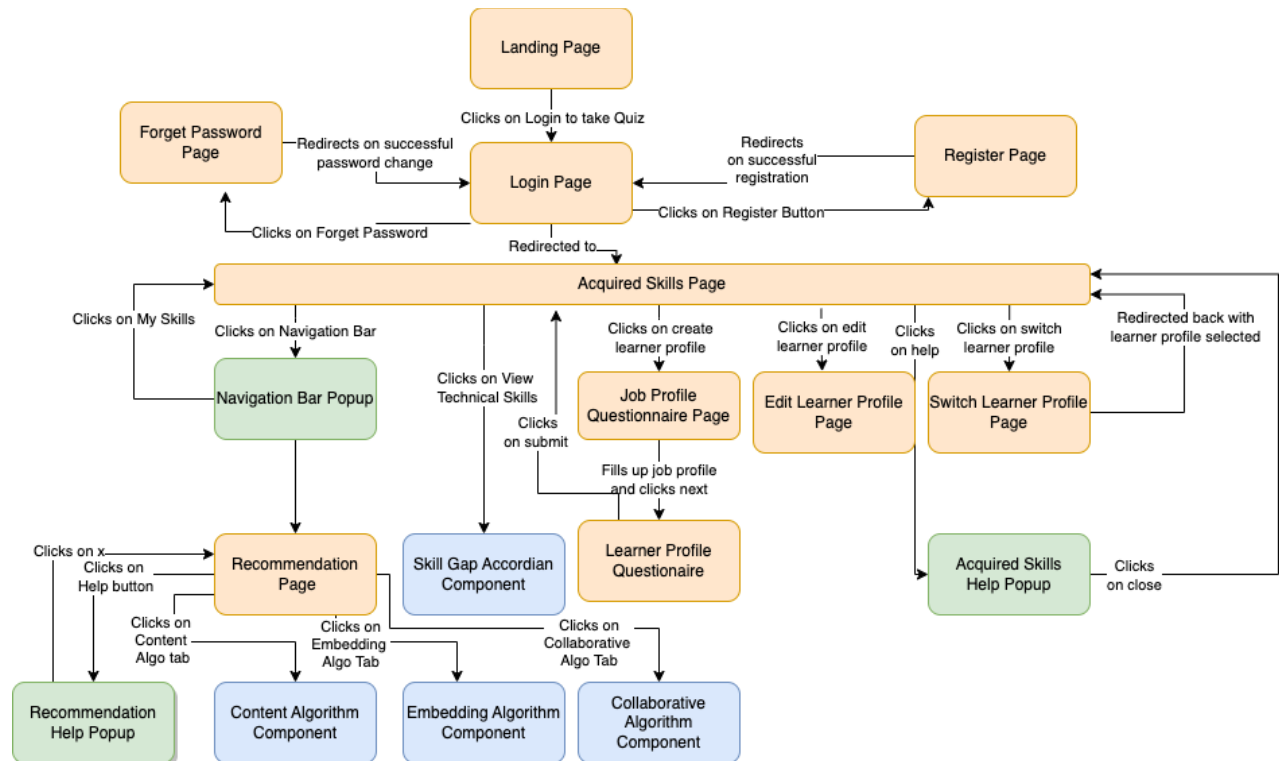


Figure 3 Dialog Map

After evaluating the user interface design (Figure 3) , a high-fidelity mock-up was created on Figma.

3.7. Project Milestones

Phase	Deadline	Milestones
Phase 1	1st August - 8th August:	<ul style="list-style-type: none"> Preliminary Research and Feasibility Study Defining the scope of the project
	9th August - 15th August:	<ul style="list-style-type: none"> Understanding relationship and normalizing the tables in the data source Learning neo4j database and Cypher Language

		<ul style="list-style-type: none"> Construct Knowledge Graph
	16th August - 23rd August:	<ul style="list-style-type: none"> Learning Next.js for front end and back-end usage Develop APIs to query knowledge graph – work on the first part which is to identify Skill Gap
Phase 2	24th August - 31st August:	<ul style="list-style-type: none"> Set up front end front and integrate APIs
	1st September - 7th September:	<ul style="list-style-type: none"> Further enhancement of front-end and back-end code (i.e. protected routes, using JWT tokens and refactoring code base) Set up deployment on Vercel
	8th September - 15th September:	<ul style="list-style-type: none"> Work on Machine Learning for Recommendation System
	16th September - 23rd September:	<ul style="list-style-type: none"> Create APIs for backend to fetch results. Deploy the API and model onto Google Cloud App Engine
	24th September - 1st October:	<ul style="list-style-type: none"> Work on front end integration of the recommendation system
	2nd October - 9th October:	<ul style="list-style-type: none"> Deployment and extra time for code enhancement
	10th October - 16th October:	<ul style="list-style-type: none"> Code Maintenance: Refactor code, delete unused components in application, unused comments, and code
	17 Oct – 23 Oct	<ul style="list-style-type: none"> Submission of Report

Table 4 Project Milestones

Chapter 4

Implementation

4.1. Backend

4.1.1. Database - Knowledge Graph Construction

4.1.1.1. Dataset Selection

The SkillsFuture Skills Framework⁵ is a comprehensive tool developed by the Singapore government to guide individuals in their skills development and career progression. It serves as a roadmap for identifying and acquiring the necessary skills and competencies required to excel in various industries and occupations. The framework encompasses a wide range of industries, from healthcare and manufacturing to finance and information technology. Data is pulled from Skills Future API as well as SkillsFuture Website. Figure 4 and Figure 5 shows the columns of the original dataset.

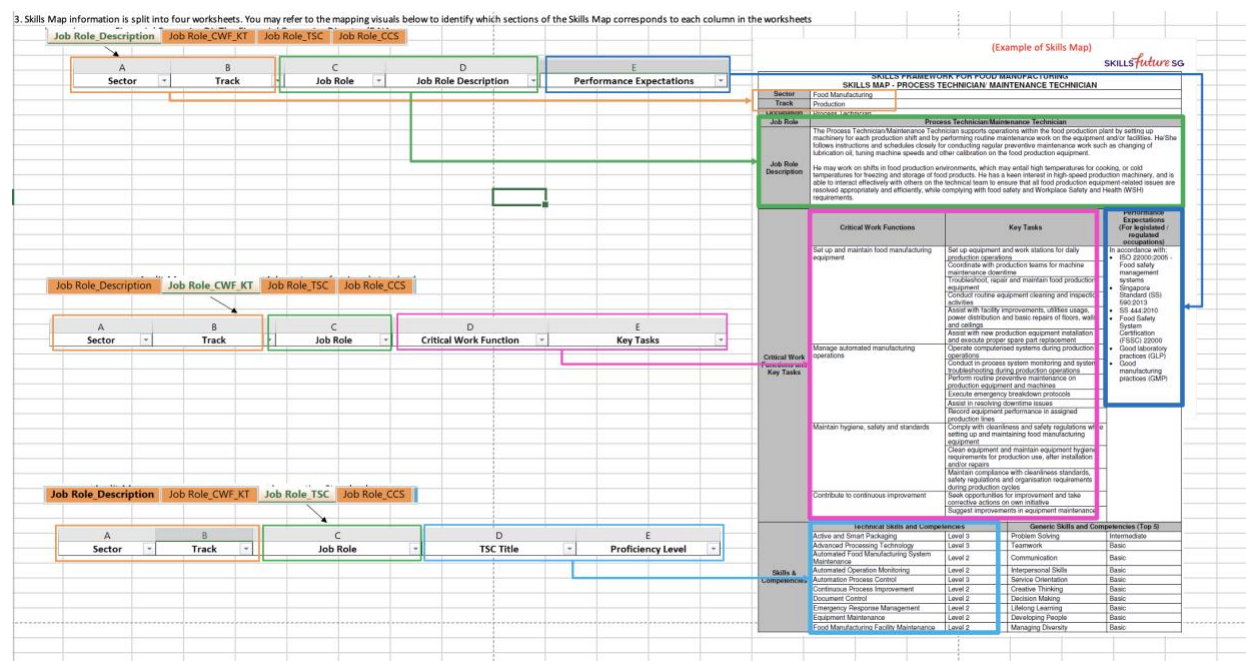


Figure 4 Original Job Skills Map

⁵ <https://www.skillsfuture.gov.sg/>

3. TSC information is split into three worksheets. You may refer to the mapping visuals below to identify which sections of the TSC reference document corresponds to each column in the worksheets

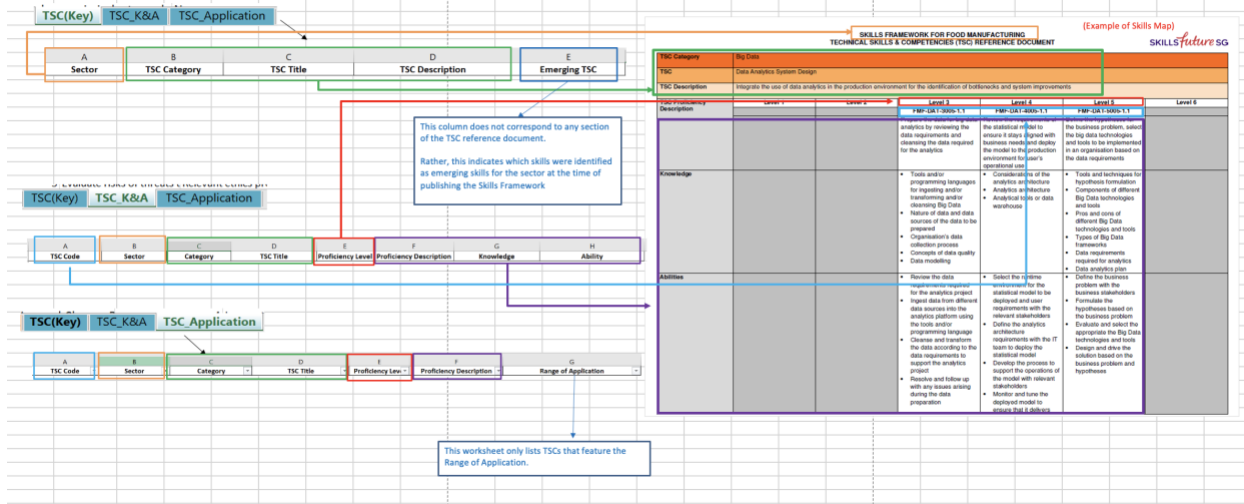


Figure 5 Original Dataset Technical Competencies Skill (TSC) Map

4.1.1.2. Data Pre-processing

Data undergoes pre-processing to facilitate the understanding of interconnections between tables. As tables within the dataset are interlinked, a combination of matching and key generation is employed to establish coherent relationships among data elements.

This initial pre-processing phase ensures the accuracy of relationships formed within the knowledge graph. [Figure 6](#) shows the reconciled relationship between the different tables and columns. Note that the ERD was not decomposed further into normalized form because our intention wasn't to depict the database in a traditional tabular format. When creating nodes in the KG, these intermediate tables would need to be joined eventually to populate the nodes in Neo 4J, ensuring the KG remains streamlined and interconnected without redundant granularity.

* CCS - Core Critical Skills
 * TSC - Technical Skills Competency

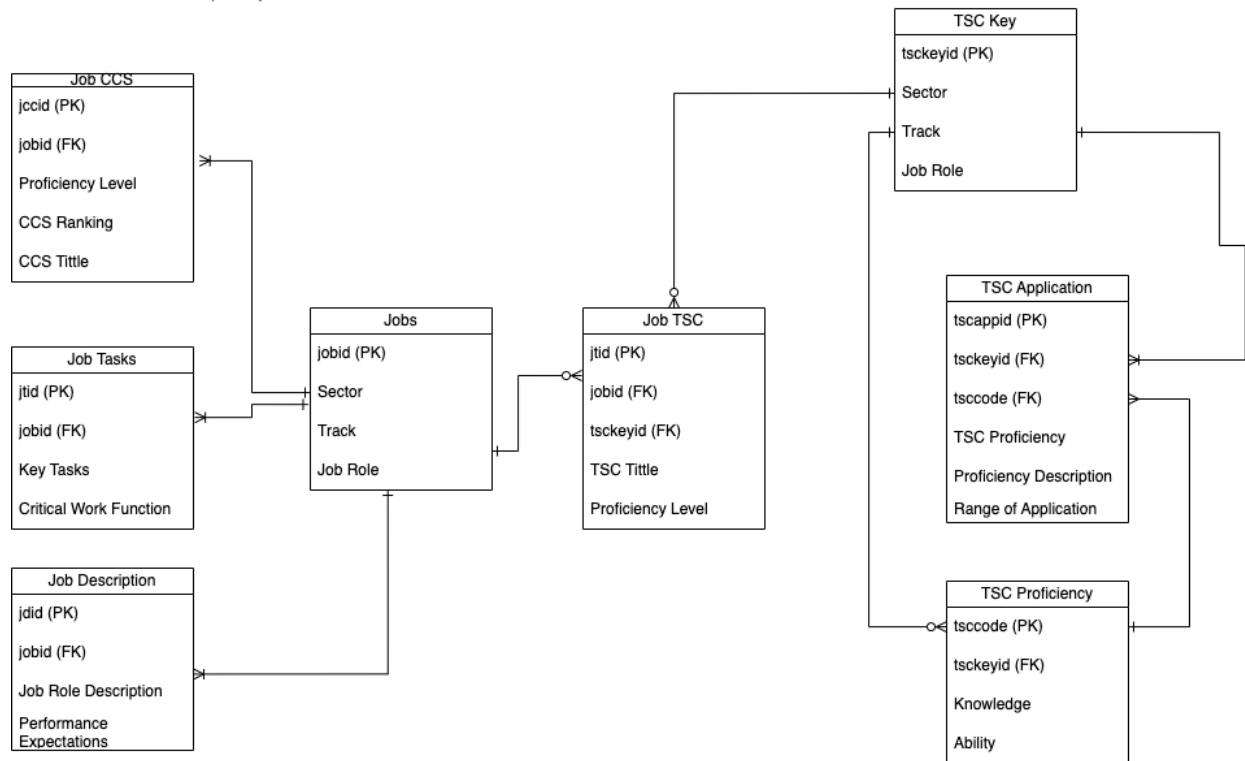


Figure 6 Entity Relation Diagram

4.1.1.3. Knowledge Graph Schema

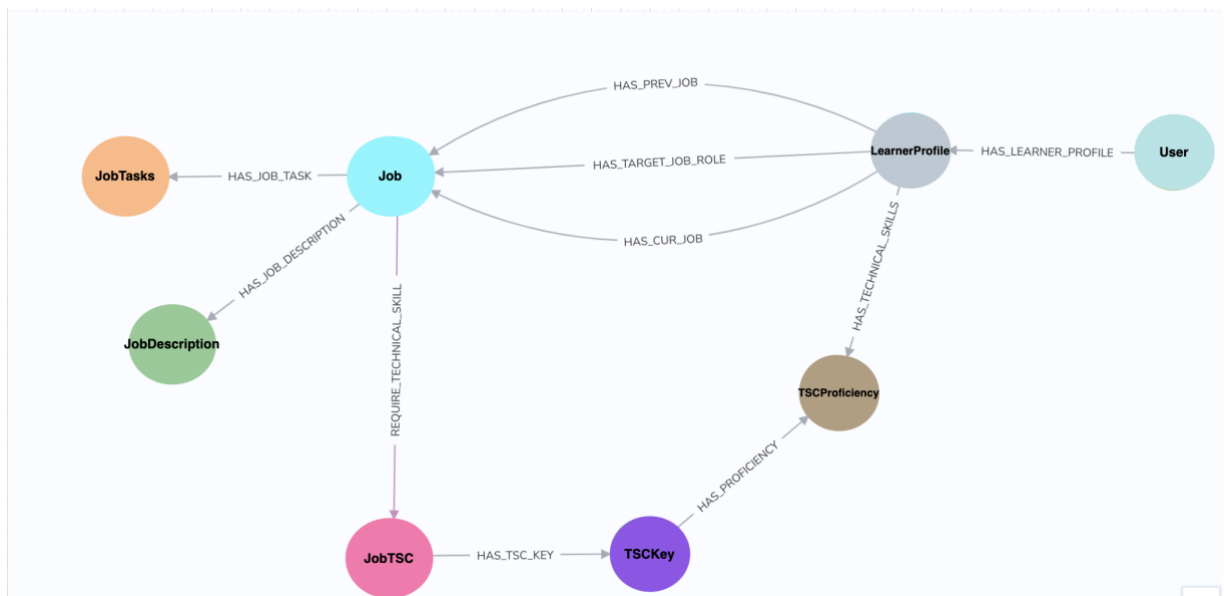


Figure 7 Knowledge Graph Schema

In the construction of the knowledge graph, we have taken a manual and contextual approach of breaking down the complex one-to-many/many-to-one relations depicted in [Figure 6](#). This approach is outlined by the Neo4J Developer Guide [25]. Tables (i.e Job CCS) and some columns (i.e Range of Application) that will not be used in the feature of the application will be omitted from being created in the Neo4J database.

Job, JobTSC, JobTasks, JobDescription Nodes

The direct one-to-many relationship between the 'Jobs' table and the 'Job Tasks', 'Job Description', and 'Job TSC' tables has been transformed into nodes. Here, each row corresponds to a node, with the columns from the table representing properties within these nodes.

TSCKey, TSCProficiency Node

To avoid unnecessary duplication of nodes, we combined the 'TSC Application' table with the 'TSCProficiency' table. This decision was based on the fact that every TSC code signifies a distinct proficiency level, a detail that's also present in the 'TSC Proficiency' table. We only want the 'Proficiency Description' attribute from the 'TSC Application' table. Thus, we can establish a join using the TSC Code in both tables. As a result, this creates the TSCProficiency node in the KG where each node represents the knowledge and ability, proficiency description and proficiency level which can be identified based on the TSCKey Node given the proficiency level.

Nodes labelled as 'TSCKey' originate from the 'TSCKey' table. Each node distinctly represents a TSCKey and its associated proficiency, linked via the 'HAS_PROFICIENCY' relationship to the 'TSCProficiency' node. Moreover, these nodes connect to specific job roles in the 'JobTSC' nodes through the 'HAS_TSC_KEY' relationship.

User and Learner Profile Nodes

New nodes have been introduced to capture user information comprehensively. During our requirements gathering phase, it was determined that users should have the capability to generate several learner profiles. This flexibility ensures that they can consistently navigate and pinpoint specific job roles they're interested in, catering to different facets of their professional aspirations or evolving skill sets or potential interest arising from the job recommendation capabilities of the app. Consequently, the 'User' node will encompass attributes such as name, email, a securely hashed password, and a distinctive UUID for the learner profile that allows us to track the current active learner profile.

The Learner Profile node describe the individual learner experience and tracking of progression or professional profile, detailing their skills and job experiences. The 'HAS_LEARNER_PROFILE' relationship serves as a connector between the 'User' node and their corresponding 'LearnerProfile'.

Nodes	Properties
Job	Sector Track Role Job jobid
JobTSC	TSC Title Proficiency Level TSC Key ID
JobTasks	Key Tasks
JobDescription	Job Role Description
TSCKey	TSC Key ID Sector TSC Category TSC Title TSC Description
TSCProficiency	Proficiency Description Proficiency Level Knowledge Ability
User	Email Name Password learneruuid
LearnerProfile	uuid createdTimestamp

Table 5 Properties of Nodes

Relation	Description	Subject & Object Entities	Relationship Properties
HAS_JOB_TASK	Connects a job to its associated tasks.	(Job, JobTasks)	
HAS_JOB_DESCRIPTION	Links a job to its textual job description.	(Job, JobDescription)	
REQUIRE_TECHNICAL_SKILL	Links Job to the specific technical proficiency level they demand	(Job, JobTSC)	
HAS_TSC_KEY	Connects a technical skill category to its unique identifier.	(JobTSC, TSCKey)	
HAS_PROFICIENCY	Associates a technical skill category with its required proficiency level and other details	(TSCKey, TSCProficiency)	<ul style="list-style-type: none"> • Proficiency Level • TSC Key ID
HAS_LEARNER_PROFILE	Links a user to their associated learning or professional profile.	(User, LearnerProfile)	
HAS_TECHNICAL_SKILL	Relates a LearnerProfile to its acquired technical skill proficiency level and key competencies.	(LearnerProfile, TSCProficiency)	
HAS_PREV_JOB	Relationships that connect a user's	(LearnerProfile, Job)	

	learner profile to previous job		
HAS_CUR_JOB	Relationships that connect a user's learner profile to current job	(LearnerProfile, Job)	
HAS_TARGET_JOB_ROLE	Relationships that connect a user's learner profile to desired job	(LearnerProfile, Job)	

Table 6 Relationships Created

```

# between Job TSC and TSCKey
query = """
LOAD CSV WITH HEADERS FROM "https://drive.google.com/uc?export=download&id=lvVDejXixlzHAaaqAgFiKyfmlNYhNoCNf" AS row
MATCH (k:JobTSC {'jtid': toInteger(row.`jtid`)})
MATCH (n:TSCKey {'TSC Key ID': toInteger(row.`TSC Key ID`)})
MERGE (k)-[:HAS_TSC_KEY]->(n)
return n, k
"""
create_rs(query)

[ ] # Between Job and Job Tasks
query = """
LOAD CSV WITH HEADERS FROM "https://drive.google.com/uc?export=download&id=1hBR1Llg09tAFxB10WX9cW7qbhMShYHZ" AS row
MATCH (k:Job {'jobid': toInteger(row.`jobid`)})
MATCH (n:JobTasks {'jtid': toInteger(row.`jtid`)})
MERGE (k)-[:HAS_JOB_TASK]->(n)
RETURN n, k;
"""
create_rs(query)

[ ] # Between Job and Job Description
query = """
LOAD CSV WITH HEADERS FROM "https://drive.google.com/uc?export=download&id=12zzLD78uxaLmU87WOTv6jR0pKLMdEIf" AS row
MATCH (k:Job {'jobid': toInteger(row.`jobid`)})
MATCH (n:JobDescription {'jdId': toInteger(row.`jdId`)})
MERGE (k)-[:HAS_JOB_DESCRIPTION]->(n)
RETURN n, k;
"""
create_rs(query)

```

Figure 8 Queries to Populate the Knowledge Graph

Once we have finalised the design of the knowledge, the schema was being modelled into a Neo4J property graph model and the data from the CSV was imported into Neo4j data using a python script as shown in [Figure 8](#).

The completed knowledge graph can be harnessed to develop APIs that pinpoint user skill gaps, construct recommendation systems, and create other APIs that facilitate user interaction with the front end of the system.

4.1.2. Recommendation Model

As mentioned earlier, in [Section 2.4.](#) , the two basic recommendation that will be implemented will be the baseline models – content and collaborative recommender to allow the user to explore the job recommendations generated by different models.

4.1.2.1. Collaborative-Based Model

In collaborative filtering, we adopted a user-user collaborative filtering approach. Based on users created during the testing phase. The following steps are taken:

A. Construct User-Job-Skill Matrix

	Job1	Job1	JobN	TSCKey1	TSCKeyN
LearnerProfile1	1	1	0	4		6
:						
:						
LearnerProfileN	0	0	1	0		1

Table 7 Collaborative Filtering User-Item-Skill Matrix

In the matrix:

The columns labelled as "Job1, Job2, ... JobN" represent a binary interaction: "1" if the user has that job before or is currently in the job and "0" otherwise.

The columns labelled as " TSCKey1, TSCKey2, ... TSCKeyN" represent the user's proficiency or self-assessed rating in each skill represented by the TSCProficiency nodes. The range of each skill is not fixed and differs across skills.

B. Construct User Vector

Based on the user current active learner profile stored in the neo4j database, the user vector is then constructed using the user past jobs and the skills they have.

C. Calculate Cosine Similarity

Using the combined User-Job-Skill matrix along with the user vector structure, we calculate the cosine similarity between users. This similarity score reflects both their interactions with jobs and their skill proficiency levels. Users with higher similarity scores are more aligned with the reference

user's skills and job preferences. Consequently, we suggest jobs from the histories of the top 10 users who have the highest cosine similarity scores.

4.1.2.2. Content-Based Model

Two primary methods were used in providing content-based recommendation:

(1) Skill Matching Model

The first method involves analysing the user skills and then subsequently recommending jobs that aligned with those skills. Skills Passport focuses on assisting users in recognizing their existing skill levels and pinpointing areas where they may lack proficiency. Given that we possess the users' skill information, we can leverage shared skills to aid users in locating appropriate job opportunities. In this model, we generate recommendation's for the user by directly comparing the skills provide by the users and the jobs.

A. Matrix Construction (Job-Skills Matrix)

Based on our knowledge graph schema, we've established that the **JobTSC** node forms a bridge between the **Job** node and the **TSCKey** node. By leveraging this connection, we're able to construct a job-skill matrix that maps each unique jobid to its corresponding **TSCKey** and associated **Proficiency Level**.

For clarity, consider the example of a 3x3 matrix:

	TSCKey1	TSCKey2	TSCKey3
JobID1	0	2	2
JobID2	0	2	0

Table 8 Content Filtering Job-Skills Matrix

Given this matrix, each row represents a distinct job, each column represents a specific **TSCKey** (or skill), and the values within the matrix denote the **Proficiency Level** for that particular job-skill pairing.

Given the importance and potential reusability of this matrix, it's stored as a pickle file. By doing so, we circumvent the need for redundant computations, ensuring that the matrix can be rapidly accessed and queried in subsequent operations.

B. User Vector

Each user is represented by the row based vector which serves as a representation of the user's skill proficiencies based on the data they've provided and some implicit assumptions we make based on their employment history.

When a user rates their skills, it typically pertains to a desired job they're interested in. Thus, we can take this explicit data (self-rated proficiencies) and incorporate it into the user vector. However, merely using this self-assessment might not offer a complete picture of the user's skillset.

To compensate for potential gaps in the user's self-assessment, we make some implicit assumptions based on their previous and current jobs. For instance, if a user has worked in a specific role in the past, we can infer that they possess certain skills associated with that role, even if they haven't explicitly rated themselves on those skills. By taking both the explicit ratings and these implicit ratings into account, we get a more holistic representation of the user's skills.

C. Similarity Calculation

To generate job recommendations, we calculate the cosine similarity between the user vector and each row in the job skills matrix. Jobs with higher similarity scores indicate a closer alignment with the user's skill set. Thus, by prioritizing these roles, we can provide users with the most relevant job matches based on their proficiencies.

(2) Textual Feature Extraction

In the second approach, we transform recommendations into an NLP task given the abundance of data related to job names, tasks, and descriptions. NLP can extract nuanced information, like sentiment or context, which can lead to more tailored recommendations. We opted for the BERT model, as it has been pre-trained on vast amounts of text, granting it a nuanced understanding of language. This makes it especially adept at capturing subtle semantics and relationships in user profiles, job descriptions, and job titles, potentially resulting in more precise recommendations [19][20].

A. Constructing the Matrix

For each job, we extracted the Job Name, Job Title, Sector, and Job Description. Multiple pieces of textual information for each job role (e.g., job-role-sector combination, job role description, tasks) are processed through BERT to generate embeddings. Each of these embeddings is represented as a high-dimensional vector. Since each job role has multiple embeddings due to different textual attributes, we compute a weighted sum of these

embeddings. Weights are assigned based on the perceived importance of each textual attribute. In this experimental model, the combination of JobName-Sector is deemed most important and carries a weight of 0.5, followed by 0.3 for Job Description and 0.2 for Job Tasks. Once the final weighted embeddings for all job roles are determined, they are stacked vertically to form the matrix. Each row in this matrix corresponds to a unique job role, with each column representing one dimension of the embedding.

B. Constructing the User-Vector

To create the user vector, there's no need to recompute the embeddings as the recommendation system primarily relies on information provided by the user in the questionnaire which directly aligns with our database information. Utilizing past job IDs and current job IDs, we fetch and construct the embedding vector. In our setup, the embeddings remain consistent for the user.

C. Calculating Similarity Vertices

As in other models, we compute the cosine similarity between the user-vector and the embeddings for all jobs. Subsequently, we recommend the top 10 jobs boasting the highest cosine similarity scores.

4.1.3. Routing API Calls

4.1.3.1. Endpoints Available

Functionality	Endpoint	Description
User Account Management	/login	GET login endpoint receives an email and password parameters and returns an authentication status based on their validity.
	/signup	GET register endpoint receives an email, name, and password as parameters and returns a registration status based on their validity.
	/forgetPassword	GET forgetPassword endpoint receives an email as a parameter and returns a status indicating the initiation of a password reset process for that email.
Learner Profile Management	/createLearnerProfile	POST endpoint accepts an email and related questionnaire data from the request body, processes job roles and proficiency data, and then

		interacts with a database (Neo4j) to create learner profile associated with that email, linking it to various job roles and technical skills. If successful, returns a status code indicating the status of the creation process
	/getJobs	GET endpoint retrieves all jobs available in database to populate the dropdown for the questionnaire
	/getFormProficiency	GET endpoint retrieves proficiency data based on the user's current and previous jobs to determine and suggest their skill levels in the questionnaire, drawing from their past employment history.
	/editLearnerProfile	PUT endpoint enables users to modify their learner profiles, including job proficiencies associated with their target roles. Submitting the relevant data to this endpoint ensures that updates are stored in the database, keeping the user's learning profile aligned with their current skills and professional aspirations.
	/getAllLearnerProfile	GET endpoint retrieves a list of all learner profiles to populate the front-end allowing users to choose their active profile.
	/updateActiveLearerProfile	POST endpoint provides a mechanism to designate the Learner Profiles associated with the user as the active profile, ensuring that the chosen profile is highlighted or prioritized in user interactions or system operations. This allows users to easily switch between and manage multiple learning profiles while focusing on one primary profile at a time.
Skill Gap/Acquired Skill Module	/getAcquiredSkills	GET endpoint retrieves a list of skills that the user has acquired over time which is determined by the questionnaire and when the user update their

		profiles. By accessing this endpoint, users can view a comprehensive summary of their skillset, which can be beneficial for self-assessment, career planning, or sharing with potential employers.
Recommendation Management	/recommendation	The GET endpoint /recommendation accepts a "strategy type" – either embedding, content or collaborative as a parameter to determine the recommendation method. Based on the chosen strategy, it delivers personalized suggestions, such as courses, jobs, or learning resources, aligning with the user's profile and preferences.

Table 9 List of Available APIs

4.1.3.2. Design Patterns and Architecture Used

In the construction of the APIs, the **Backend-for-Frontend (BFF) design pattern**, a strategic approach tailored to optimize the connection between distinct front-end experiences and the underlying services. This microservice architectural pattern facilitated streamlined interactions, ensuring that each front-end application or client received data optimized for its unique requirements. By compartmentalizing the backend services for individual front ends, we achieved improved performance, reduced unnecessary data transfers, and enhanced the modularity of our codebase, paving the way for easier feature integrations and maintenance in future iterations [13].

To enhance the scalability of our recommendation engine and allow developers the flexibility to interchange the deployed recommendation mechanism, we've adopted the **Strategy Design Pattern** [14] for our recommendation endpoint. This approach promotes easy incorporation of multiple recommendation algorithms by essentially decoupling the recommendation logic into swappable strategies. This adaptability means the system can fluidly transition between various recommendation techniques like content-based, collaborative filtering, or future methodologies, based on user preferences, server demands, or other immediate determinants. Presently, the frontend displays three algorithmic options. This design ensures performance and precise recommendations, eliminating the need for system redesign with each algorithm modification.

4.2. Front-End

4.2.1. Front-End Implementation Details

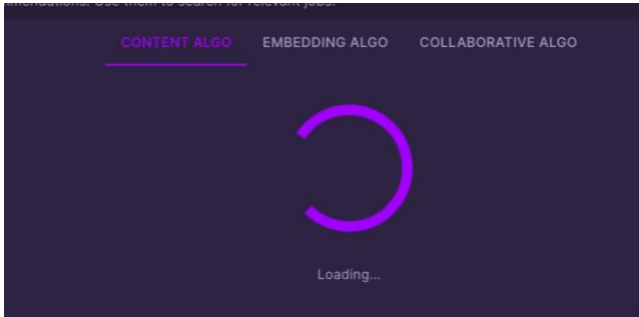
4.2.1.1. Design and Prototyping's

Shneiderman's "Eight Golden Rules of Interface Design" were used to create an intuitive and efficient application. The three most commonly applied rules from the eight will be detailed below.

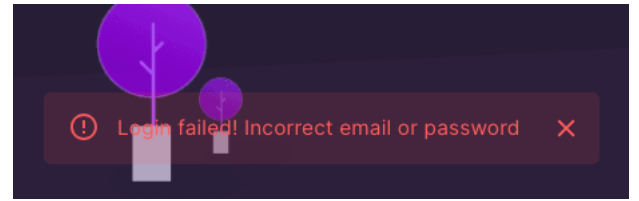
(A) Strive For Consistency

This principle emphasizes maintaining a uniform appearance, colour scheme, structure, and functionality in the interface. By doing so, users can quickly grasp the workings of the interface, thereby reducing the time it takes to get accustomed to it. The consistent use of typography, font, and purple theme in the application emphasis on consistency.

(B) Offer Informative Feedback

 <p><i>Figure 9 Loading Card</i></p>	<p>For every action a user takes, it's crucial to provide feedback. In the context of SkillsPassport, when users initiate an action, a "loading card" appears. This card not only indicates that the system is processing the user's request but also provides a visual cue, ensuring that users understand something is happening in the background.</p>
--	---

(C) Error Handling

 <p><i>Figure 10 Error Card</i></p>	<p>Effective error handling is vital to ensure users aren't left frustrated or confused. In the context of SkillsPassport, when users encounter a login failure, the system provides a clear and informative error message.</p>
--	---

4.2.1.2. File Structure of Front-End Components

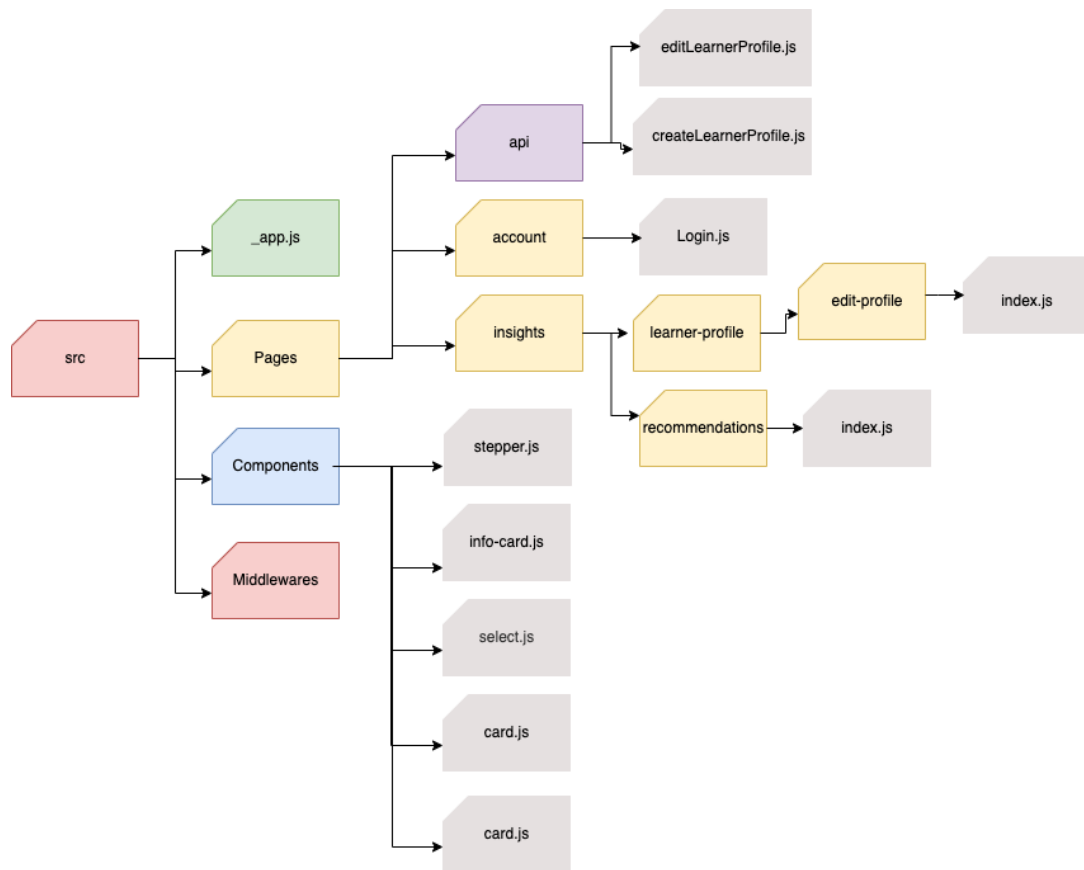


Figure 11 File Structure Next.js Application

[Figure 11](#) showcases a systematic representation of a typical Next.js application's directory and file structure. At the core is the **src** directory, serving as the primary source folder for the application.

Nested inside **src**, we have three significant directories: **Pages**, **Components**, and **Middlewares**.

- **Pages:** This is a special directory in Next.js where the routing mechanism is intrinsically tied to the file structure. Any JavaScript file inside it becomes a route that Next.js will automatically handle. Within this directory, we see further organization:
 - **api:** This directory typically holds API route handlers, enabling the creation of API endpoints. An example file, **createLearnerProfile.js**, suggests functionality to handle profile creation requests.
 - Outside the **api** directory, the **index.js** files in various subdirectories serve as the main entry points for rendering frontend content for their respective routes. These

are essentially React components and dictate the structure, layout, and content of the corresponding pages in the frontend.

- **insights/learner-profile/edit-profile:** It would translate to a route named **/insights/learner-profile/edit-profile** in the application route. The **index.js** inside the **edit-profile** directory will contain all the frontend components, logic, and styling necessary to render the **Edit Profile** page.
- **Components:** This directory would store reusable UI components, promoting the DRY (Don't Repeat Yourself) principle, and fostering consistency across the application. Components like **info-card.js**, **select.js**, and **card.js** are UI elements that can be utilized in multiple places.
- **Middlewares:** Middlewares are used to modularize code, for tasks like error handling, and authentication.

This structure allows for modularity and clear separation of concerns allowing for reusability, as components and functionalities can be easily reused across different parts of the application without duplication.

4.2.2. Front End Walk Through

4.2.2.1. Landing Page

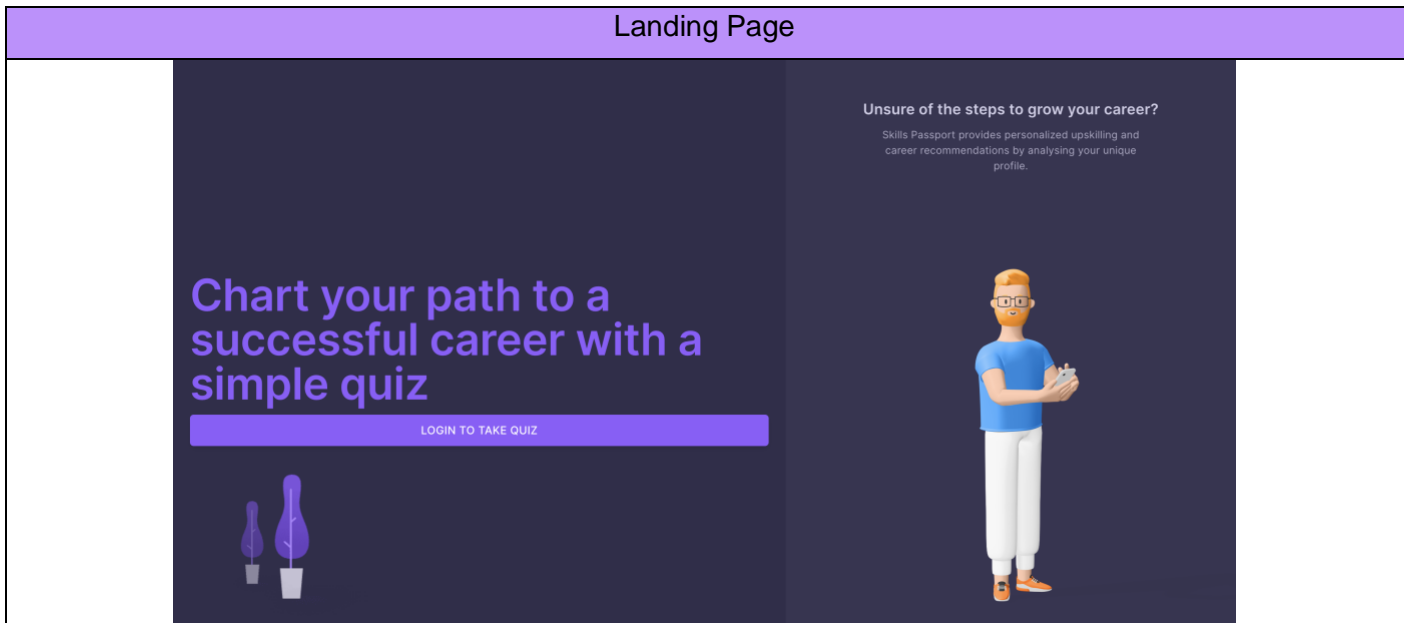
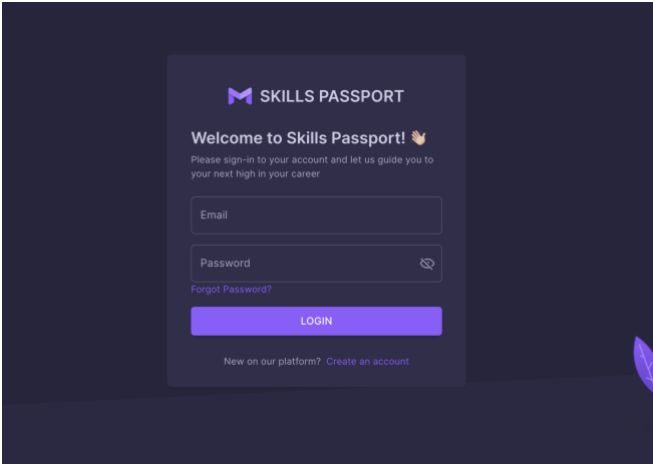
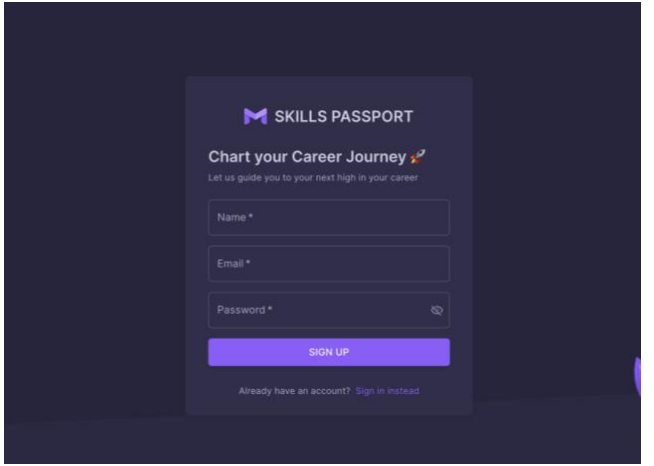
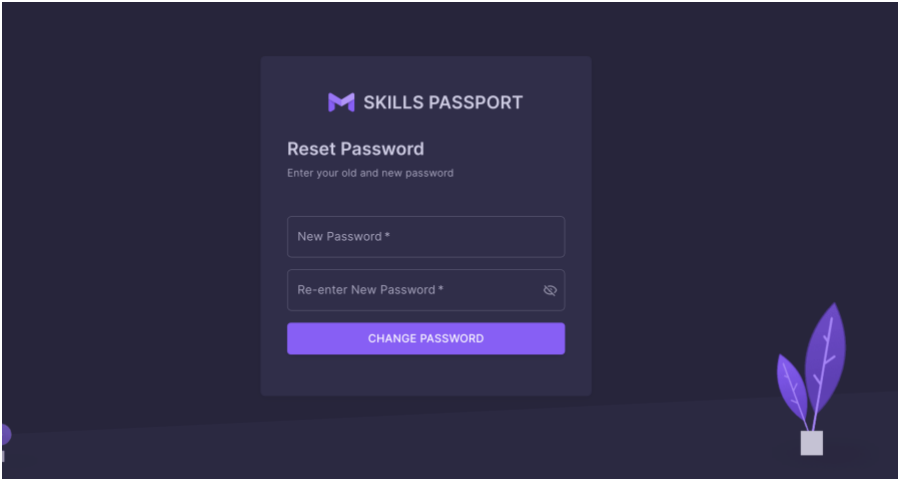


Figure 12 Landing Page

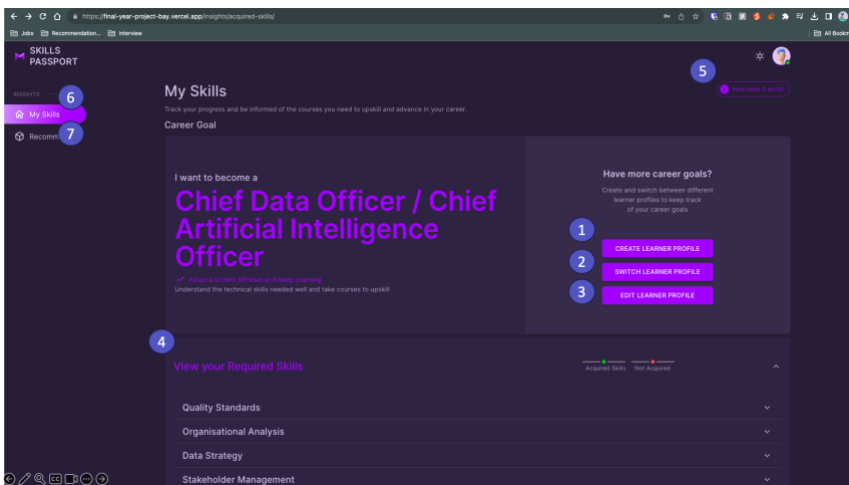
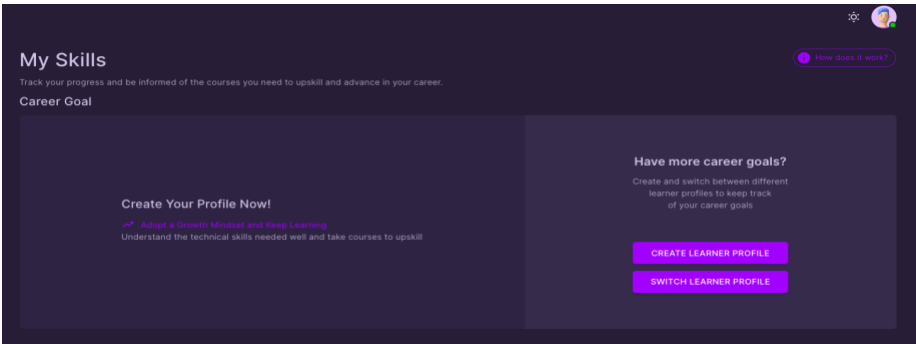
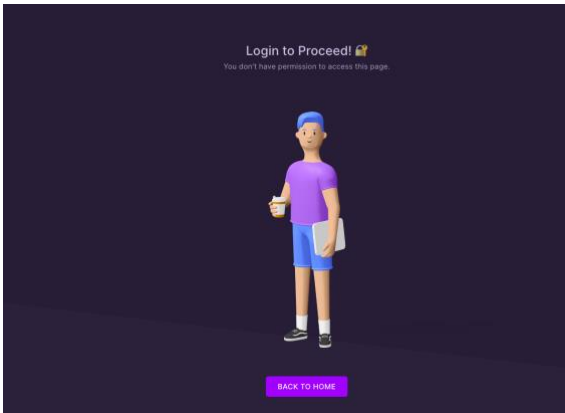
Landing Page: The Landing Page serves as the introduction to the platform, highlighting its main features, advantages, and standout qualities. For users ready to engage, "Login To Take Quiz" Call to Action guide their next steps.

4.2.2.2. User Account Module

Login Page	Register Page
	
<p>Figure 13 Login Page</p>	<p>Figure 14 Register Page</p>
Allows the user to enter their email and password to access the Dashboard Module; incorrect credentials will trigger an error alert. When the user submits their data, the /login GET endpoint is called to authenticate the user.	Allows user to sign up with their details to join our platform; once registered, user be directed to the Login Page. When the user submits their data, the /register endpoint POST the data to the database
Forgot Password Page	
	
<p>Figure 15 Forgot Password Page</p>	
Enter your email to receive a reset link; follow the instructions to regain access. This shows the page where user can reset their password facilitated by the /forgotPassword endpoint.	

4.2.2.3. Dashboard Module

4.2.2.3.1. Dashboard Page

Dashboard Page (If Logged In)	Description
 <p>The screenshot shows a user logged into the 'My Skills' dashboard. The main heading is 'My Skills' with a subtitle 'Track your progress and be informed of the courses you need to upskill and advance in your career.' Below this is a 'Career Goal' section with the objective 'I want to become a Chief Data Officer / Chief Artificial Intelligence Officer'. To the right, there's a 'Have more career goals?' section with buttons for 'CREATE LEARNER PROFILE', 'SWITCH LEARNER PROFILE', and 'EDIT LEARNER PROFILE'. At the bottom, there's a 'View your Required Skills' section with a list of skills: Quality Standards, Organisational Analysis, Data Strategy, and Stakeholder Management. A navigation bar on the left contains icons for 'My Skills' (6) and 'Recommendation' (7).</p> <p>Figure 16 Dashboard Page (Logged In)</p>	<p>From the Dashboard Page, the user will be able to access</p> <p>1-3: Learner Profile Module</p> <p>4: Acquired Skill Component</p> <p>5: Dashboard Help Modal</p> <p>6-7: Belongs to the navigation bar</p> <ul style="list-style-type: none"> Clicking on 6 brings the user back to the current page Clicking on 7 brings the user to the Recommendation Module <p>The information on this page is fetched from the /getAcquiredSkills endpoint.</p>
Dashboard Page (If not logged in or learner profile not created)	Description
 <p>The screenshot shows the dashboard for a user who is not logged in or has not created a learner profile. The main heading is 'My Skills' with a subtitle 'Track your progress and be informed of the courses you need to upskill and advance in your career.' Below this is a 'Career Goal' section with the objective 'Create Your Profile Now!'. To the right, there's a 'Have more career goals?' section with buttons for 'CREATE LEARNER PROFILE' and 'SWITCH LEARNER PROFILE'. At the bottom, there's a 'Login to Proceed!' message with a 'BACK TO HOME' button.</p> <p>Figure 17 Dashboard Page (Not Logged in or Learner Profile Not Created)</p>	<ul style="list-style-type: none"> Public users can freely visit the Landing Page, Register Page, and Acquired Skills Page. If a user isn't signed in and arrives at this page, their career objective won't be displayed, nor will the skill-gap. Tapping on options 1-3 and 7 Figure 18 will be shown and user will be redirect to Login Page User will also be seeing this UI if there's no learner profile being created yet and no edit Learner Profile Button shown
 <p>The screenshot shows an unauthorized access page. It features a character holding a folder and a 'Login to Proceed!' message. Below the character is a 'BACK TO HOME' button.</p> <p>Figure 18 Unauthorized Access</p>	

4.2.2.3.2. Acquired Skill Component

Acquired Skill Component

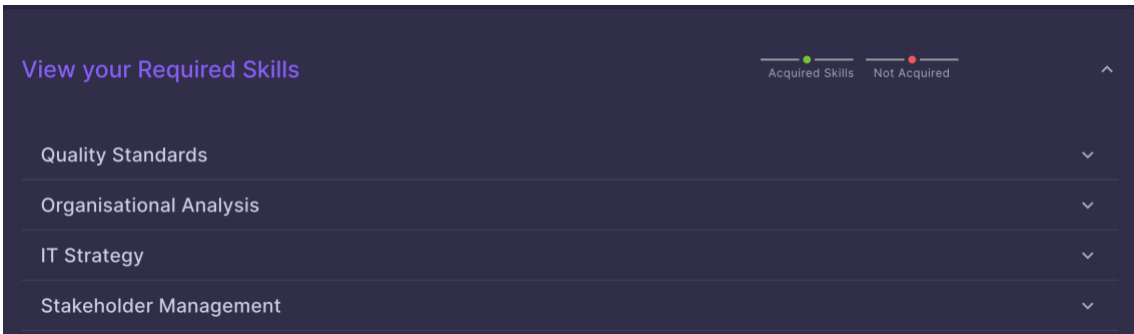


Figure 19 Acquired Skills Collapsed Accordion View

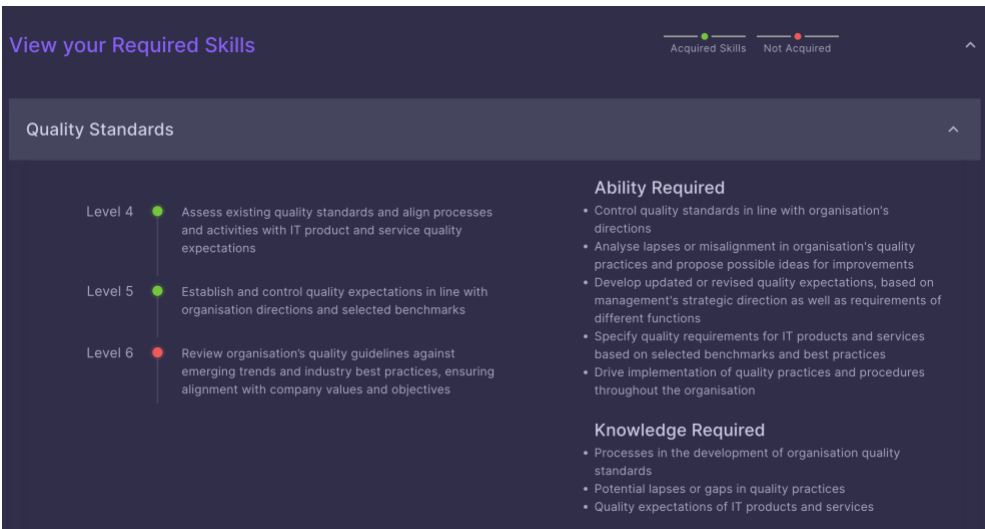


Figure 20 Acquired Skills Expanded Accordion View

This component allows users to gauge their skill proficiency against desired job requirements, identify their skills gap

- **Knowledge** is defined by the understanding one has regarding a particular skill.
- **Ability** represents the practical application of that knowledge. It details what a user can do with the knowledge they possess, such as controlling quality standards and analyzing lapses in quality practices.

The interface uses a color-coded system:

- **Green:** Represents the proficiency level a user has met or exceeded concerning job requirements.
- **Red:** Highlights the proficiency levels required for a job that the user hasn't attained.
- **Grey:** Denotes levels beyond the user's current proficiency up to the job's requirement.

In essence, the platform aids users in identifying their competencies and areas for improvement concerning specific job requirements.

4.2.2.3.3. Dashboard Help Modal

Dashboard Help Modal

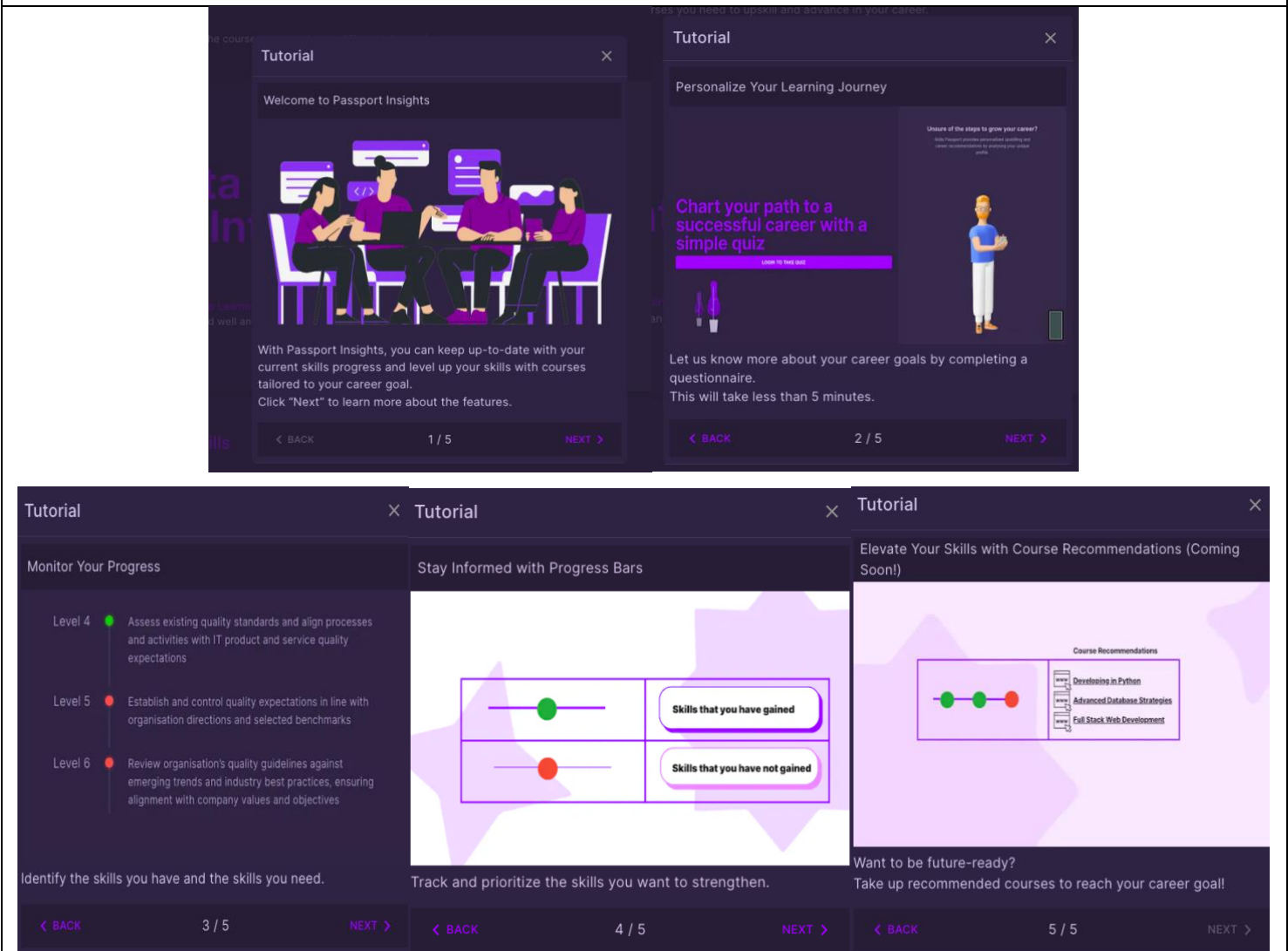
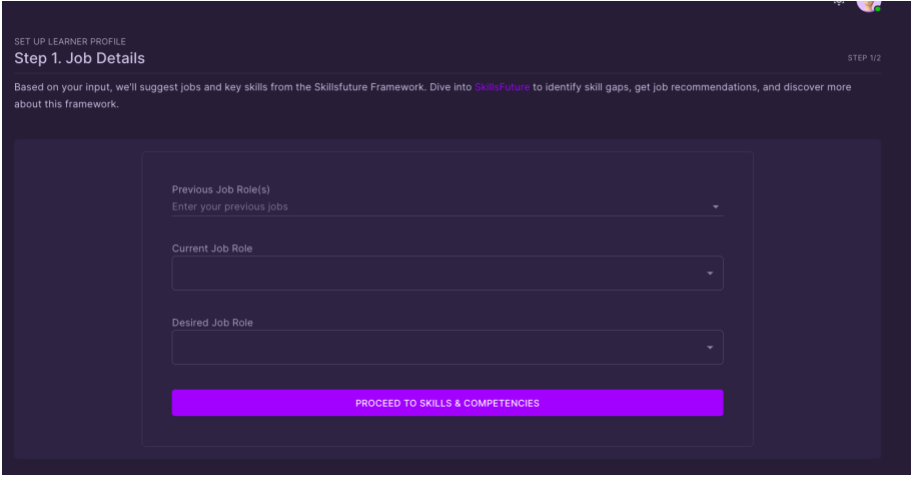
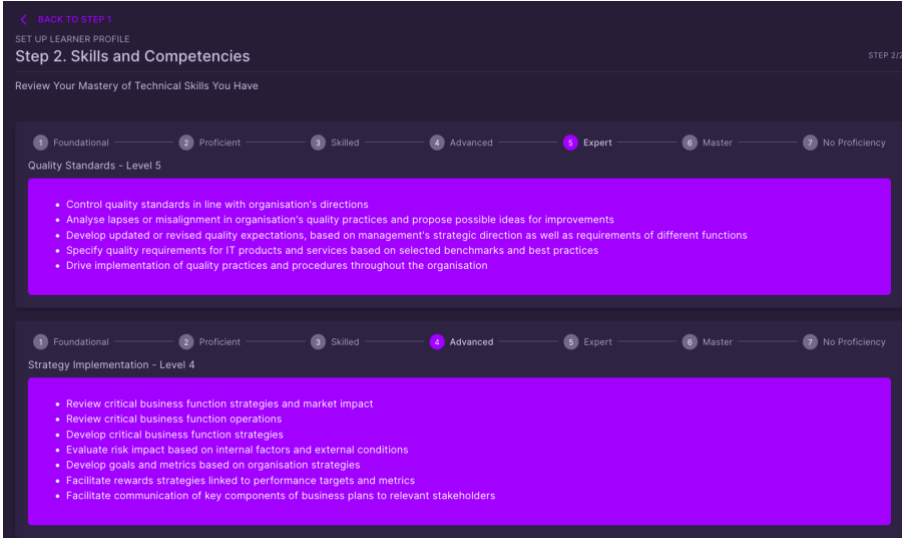


Figure 21 Dashboard Help Modal

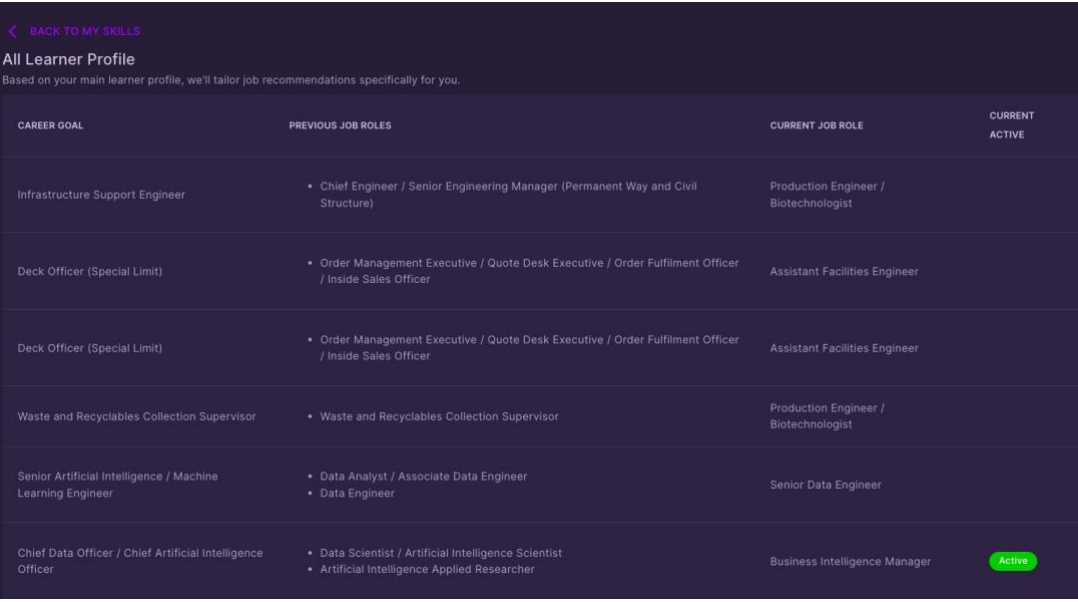
A dashboard help modal provides the users with assistance, explanations, or additional information about the features and functionalities how to start charting their career goal and progression.

4.2.2.4. Learner Profile Module

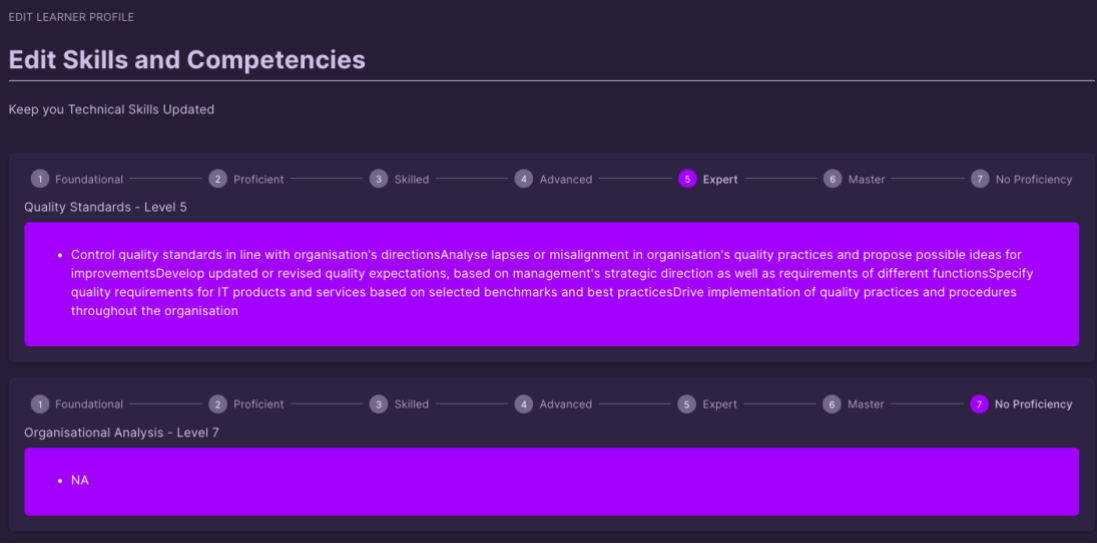
4.2.2.4.1. Create Learner Profile

Questionnaire Part 1	Description
 <p>Figure 22 Learner Profile Creation Part 1</p>	<p>The GET /getJobs endpoint is called once and its results are cached using Next.js. This ensures efficient data retrieval, enabling users to populate dropdown menus for their previous and desired job roles seamlessly. On clicking proceed he/she is routed to questionnaire part 2.</p>
Questionnaire Part 2	Description
 <p>Figure 23 Learner Profile Creation Part 2</p>	<p>Using the user's current and previous job roles, the GET request /getFormProficiency endpoint determines the user's proficiency level and automatically fills the form. While the system provides default settings, users can adjust based on their self-assessment. Upon finalizing their details, users can submit the form, triggering a POST request to /createLearnerProfile on clicking submit.</p>

4.2.2.4.2. Switch Learner Profile

Switch Learner Profile	Description
 <p>Figure 24 Switch Learner Profile Page</p>	<p>On this page, all learner profiles are retrieved using the GET /getLearnerProfile endpoint. The currently active learner profile is highlighted with a green tag. When any row is clicked, the system updates the active learner profile by making a call to the /updateLearnerProfile endpoint.</p>

4.2.2.4.3. Edit Learner Profile

Edit Learner Profile	Description
 <p>Figure 25 Edit Learner Profile Page</p>	<p>If the user has a learner profile, the /getAcquiredSkills endpoint retrieves their proficiency for the targeted job. Displayed in the UI, users can adjust proficiency. After changes, clicking "Submit" sends a PUT request to /editLearnerProfile to update the profile.</p>

4.2.2.5. Recommendation Module

4.2.2.5.1. Recommendation Page

Recommendation Page

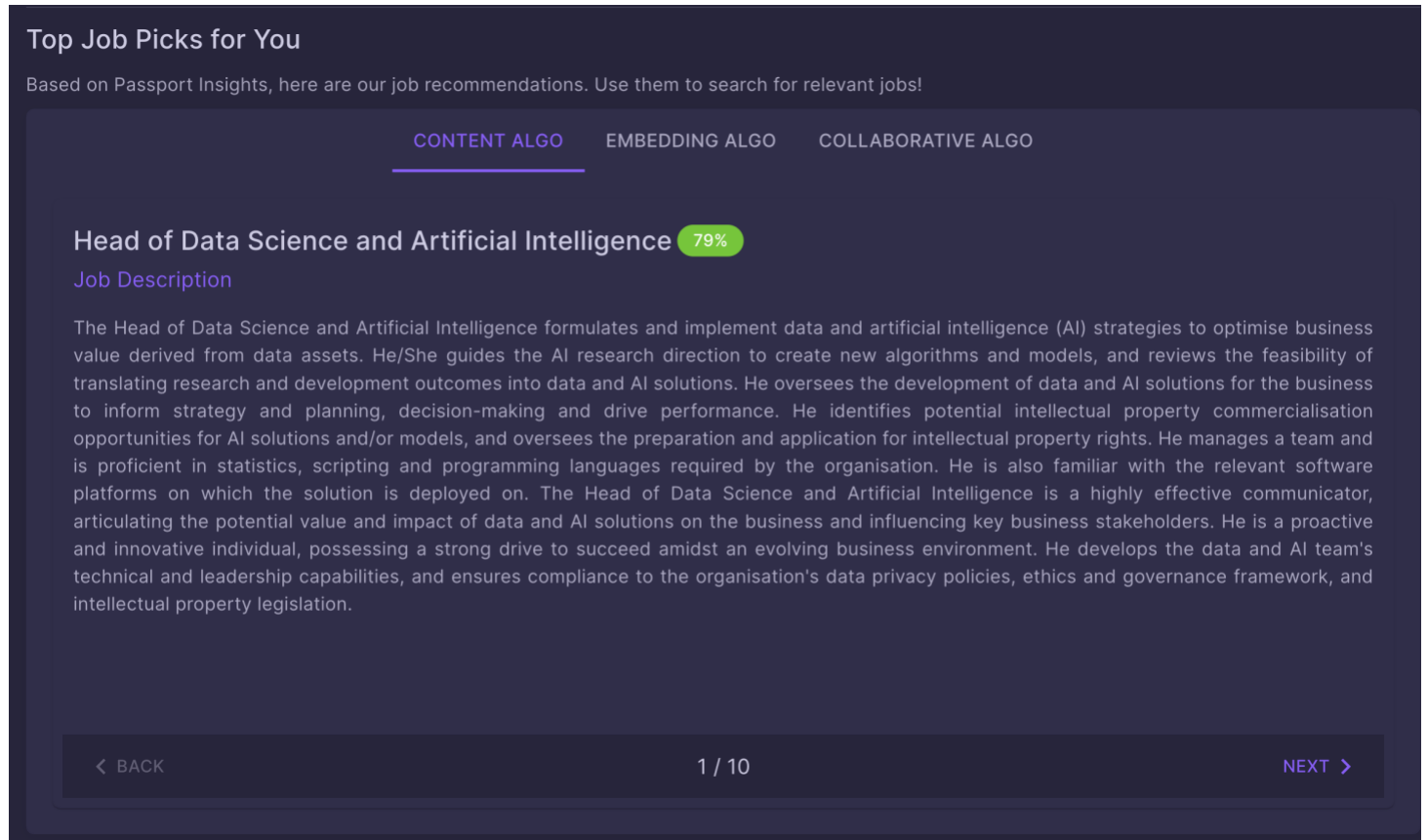


Figure 26 Recommendation Page

When users select different algorithm tabs, the system fetches recommendations tailored to the user's active learner profile. This is achieved by invoking the **GET /recommendations** endpoint and specifying the desired strategy type; in this context, it would be "content".

4.2.2.5.2. Recommendation Module



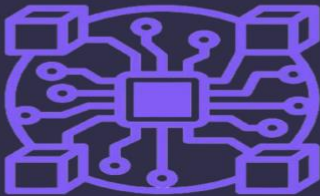

Recommendation Help Modal	
<div><div><div><div>Tutorial</div><div>Getting Started:</div><div></div><div><p>Start by choosing your preferred recommendation algorithm to receive job suggestions. Our system understands that you know your needs, preferences, and career aspirations best. Align the recommendations with your individual career goals and professional development pathway.</p></div><div><div>< BACK</div><div>1 / 4</div><div>NEXT ></div></div></div><div><div>Tutorial</div><div>Understanding Content-Based Algorithm</div><div></div><div><p>The Content-Based Algorithm recommends jobs based on your profile and past behavior. It analyzes the content of your profile, including your skills, experience, and preferences, and matches it with job descriptions to find the most suitable job.</p><p>Ideal For: Users seeking jobs closely related to their profiles and past applications.</p></div><div><div>< BACK</div><div>2 / 4</div><div>NEXT ></div></div></div></div><div><div><div>Tutorial</div><div>Exploring Embedding-Based Algorithm</div><div></div><div><p>Think of Embedding Methods like a smart matching game. Both users and jobs are assigned special tags that describe them in various ways. By comparing these tags, the system identifies how well a user and a job match. This enables it to discover hidden connections and similarities, leading to recommendations that are a good fit in multiple ways.</p><p>Ideal For: Users looking for jobs closely related to their profiles and past applications.</p></div></div><div><div>Tutorial</div><div>Discovering Collaborative-Based Algorithm</div><div></div><div><p>Collaborative filtering recommends jobs based on the behavior of similar users. If users with similar preferences and behavior liked a particular job, this algorithm will recommend that job to you.</p><p>Ideal For: Exploring diverse job opportunities liked by peers with similar tastes and preferences.</p></div><div><div>< BACK</div><div>4 / 4</div><div>NEXT ></div></div></div></div></div>	<p>A recommendation help modal provides the users with assistance, explanations, and additional information about the different recommendation algorithms available in a layman way.</p>

Figure 27 Recommendation Help Modal

4.3. System Testing

4.3.1. Black Box Testing

Black box testing is an evaluation method that assesses software functionality without inspecting its internal code. Testers interact with the software as typical users, ensuring built functionality aligns with specified requirements. This approach effectively detects usability problems, functional anomalies, and verifies the software's alignment with user expectations. In our project, black box testing was executed to verify that all system functionalities adhere to the design specifications.

4.3.2. Unit Testing

Jest, a widely adopted JavaScript testing framework, was utilized for testing in the project. In this project, test cases were written to assess API requests.



```
You, 3 minutes ago | 1 author (You)
jest.mock('../db/neo4j', () => ({
  read: jest.fn(),
}));

// Mocking middleware to skip JWT verification for the tests.
jest.mock('../middlewares/verifyJWT', () => (req, res, next) => next());

import { read } from '../db/neo4j';
import handler from '../pages/api/jobs/getAllJobs';
```

Figure 28 Setup Testing Environment

In the setup phase, we configure the testing environment by establishing essential mocks. We employ `jest.mock` to simulate database and middleware dependencies. Specifically, we mock the `read` function from `'../db/neo4j'` using `jest.fn()`. This approach intercepts calls to the database functions and directs them to the mock, ensuring controlled testing. Additionally, we mock the JWT verification middleware to bypass it during the testing process, enabling focused API testing.

```

it('should return a 500 status code on database error', async () => {
  // Arrange
  read.mockRejectedValueOnce(new Error('Database error'));

  const mockReq = {};

  const mockRes = {
    status: jest.fn().mockReturnThis(),
    send: jest.fn()
  };

  // Act
  await handler(mockReq, mockRes);

  // Assert
  expect(mockRes.status).toHaveBeenCalledWith(500);
  expect(mockRes.send).toHaveBeenCalledWith('Internal server error');
});

```

Figure 29 Using AAA Pattern for Testing

For all unit testing, the AAA - Arrange, Act, Assert testing pattern is being followed [26]. The AAA pattern, which stands for "Arrange, Act, Assert," is a widely used testing methodology that separates a unit test into three distinct phases:

1. **Arrange:** Setup the test environment. E.g. preparing test data, initialising variables
2. **Act:** Execute the functionality being tested.
3. **Assert:** Verify behaviour of the code and make assertions based on the results.

Refer to [Figure 29](#) for an example.

```

PASS  src/__tests__/getAllJobs.test.js
Job API Handler
  ✓ should fetch and format all job data successfully (9 ms)
  ✓ should return a 500 status code on database error (36 ms)

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        0.433 s, estimated 3 s

```

Figure 30 Output of Passed Testcases

By executing the npm test command, we can automatically run all unit tests whenever changes are pushed to the codebase. This continuous testing process helps ensure that no breaking changes are introduced. Refer to [Figure 30](#) for a sample output of a passed unit test case.

Chapter 5

DevOps

DevOps has gained traction in software engineering due to its ability to enhance the pace at which development groups roll out new features while bolstering software quality and reliability. The name "DevOps" is a fusion of "Development" and "Operations", highlighting the emphasis on automation to accelerate development and enhance infrastructure scalability [17]. For this project, we have employed CI/CD tools to standardize and simplify the development workflow. CI/CD which stands for continuous integration and continuous delivery, is a software development practice where code changes are automatically integrated, tested, and delivered to production, enabling faster and more consistent software releases [18].

5.1. Tools Used for CI/CD

In this section, we will discuss tools used for deployment and ensuring continuous integration and continuous delivery.

5.1.1. Vercel

Vercel was chosen as the deployment tool for the next.js application.

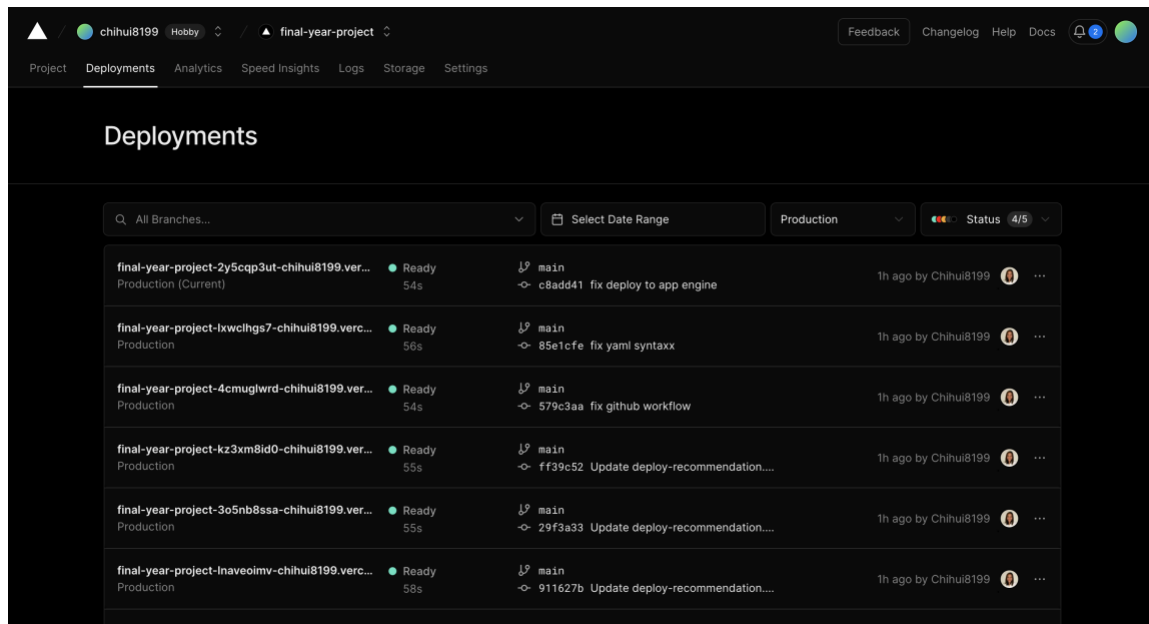


Figure 31 Automatic Deployment on Vercel

Vercel is a cloud-based Platform as a Service (PaaS) specialized in the hosting and deployment of web applications and websites. It abstracts away most of the infrastructure management tasks, such as server provisioning and scaling. This means that Vercel offers a fully managed hosting environment for your web applications, simplifying much of the underlying infrastructure management [16]. Users can link their code repositories, such as those hosted on GitHub, and Vercel handles the deployment process, which includes tasks like server setup, scaling, and content delivery. Vercel has the distinction of being both the creator and maintainer of Next.js, which results in seamless integration and optimization when deploying applications, as any new features are introduced first to Next.js.

Furthermore, Vercel automatically provisions SSL certificates for deployed domains, ensuring that websites and applications benefit from HTTPS encryption, enhancing security and trust for end users.

By integrating with our version control tool, GitHub, and configuring the CI/CD deployment settings, such as triggering deployments from the main branch and specifying build and testing configurations, Vercel automates the CI/CD deployment process. At present, given the relatively small scale of the application, Vercel is exclusively used for provisioning the production environment. There isn't a separate staging environment, and development work primarily occurs on local development setups. Refer to [Figure 31](#) for an example of the deployment process.

5.1.2. Google Cloud and Google App Engine

Google Cloud was chosen as the platform to host the recommendation engine alongside Vercel's deployment for a user-centric Next.js application. While Vercel is generally capable of deploying Flask applications, our specific use case presented challenges. Due to the machine learning nature of the serverless function for the recommender, which relies on packages like TensorFlow, we encountered limitations related to package size when deploying it on Vercel.

To tackle these challenges, we opted for Google App Engine, a managed environment purpose-built for efficient serverless function deployment. Google App Engine adeptly resolved the limitations we encountered within Vercel's framework, enabling us to smoothly deploy and execute the Flask serverless function, which includes TensorFlow.

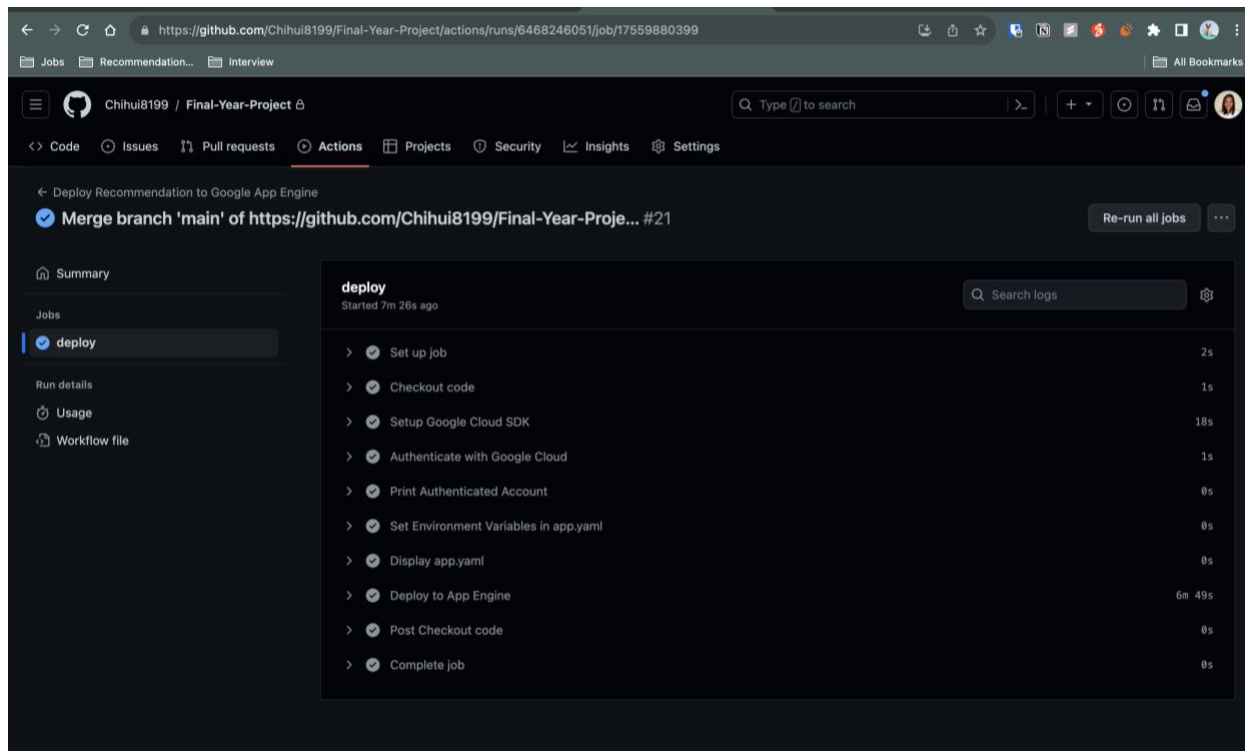


Figure 32 GitHub Action Workflow

Additionally, GitHub Actions played a pivotal role in our CI/CD pipeline, as automatic deployment wasn't available for Google Cloud. GitHub Actions seamlessly integrates into the GitHub platform, serving as a versatile CI/CD (Continuous Integration/Continuous Deployment) tool. It empowers users to automate various software workflows, encompassing tasks such as building, testing, and deploying their applications directly from their GitHub repositories.

In our specific use case, we configured GitHub Actions to automatically redeploy the recommender engine whenever new changes were pushed to the main branch, ensuring a streamlined and responsive development process. This can be seen from [Figure 32](#)

```

name: Deploy Recommendation to Google App Engine

on:
  push:
    branches:
      - main
    paths:
      - 'FYP Website/recommendation_system/**'
  workflow_dispatch:

jobs:
  deploy:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v2

      - name: Setup Google Cloud SDK
        uses: google-github-actions/setup-gcloud@main
        with:
          project_id: recommendation-ml-401105
          service_account_key: ${ secrets.GCLOUD_AUTH }}
          export_default_credentials: true

      - name: Authenticate with Google Cloud
        run: |
          echo "$GCLOUD_AUTH_JSON" > tmp_key.json
          gcloud auth activate-service-account --key-file=tmp_key.json
        env:
          GCLOUD_AUTH_JSON: ${ secrets.GCLOUD_AUTH }}

      - name: Print Authenticated Account
        run: gcloud auth list

      - name: Set Environment Variables in app.yaml
        working-directory: FYP Website/recommendation_system
        run: |
          echo "env_variables:" >> app.yaml
          echo "  AURA_CONNECTION_URI: '$AURA_CONNECTION_URI'" >> app.yaml
          echo "  AURA_USERNAME: '$AURA_USERNAME'" >> app.yaml
          echo "  AURA_PASSWORD: '$AURA_PASSWORD'" >> app.yaml
          echo "  JWT_SECRET: '$JWT_SECRET'" >> app.yaml
        env:
          AURA_CONNECTION_URI: ${ secrets.AURA_CONNECTION_URI }}
          AURA_USERNAME: ${ secrets.AURA_USERNAME }}
          AURA_PASSWORD: ${ secrets.AURA_PASSWORD }}
          JWT_SECRET: ${ secrets.JWT_SECRET }}

      - name: Display app.yaml
        working-directory: FYP Website/recommendation_system
        run: cat app.yaml

      - name: Deploy to App Engine
        working-directory: FYP Website/recommendation_system
        run: gcloud app deploy app.yaml --project=recommendation-ml-401105

```

Figure 33 GitHub Actions Workflow Script

Figure 33 shows the configured YAML file that defines a GitHub Actions workflow to automatically re-deploy the recommendation system to Google App Engine whenever changes are pushed to the main branch. It sets up the Google Cloud SDK, authenticates using a service account, sets environment variables, and deploys the application using gcloud app deploy.

This harnesses the strengths of Google Cloud for machine learning and serverless functions, while Vercel excels in frontend deployment and user-centric features.

Chapter 6

Conclusion and Future Works

6.1. Conclusion

This Final Year Project delves into the objectives, design considerations, and development process of "Skills Passport" web application. The project centres on a web application tool designed to offer job seekers (such as those transitioning between jobs or recent graduates) and career coaches a clear understanding of their skill gaps and their proximity to their desired roles. Additionally, based on the user's chosen active profile aligned with their target job in a specific sector, the application includes job recommendations as an auxiliary feature. This feature can be particularly instrumental in nudging job seekers and career coaches to consider and apply for roles they hadn't previously contemplated but are, in fact, a close match to their profiles.

We have effectively utilized the modern cloud database, Neo4J, to develop a Singapore-centric Job Knowledge Graph. Additionally, the project incorporates features ensuring security and maintainability by harnessing advanced cloud technologies such as Vercel and Google Cloud for deployment and CI/CD processes. With these achievements, the project fulfils the objectives detailed in [Section 1.3](#) – the project scope.

6.2. Future Works

Grasping the intricacies of jobs and skills in the job market is a complex task, and crafting a proficient recommender is challenging. This section will detail potential improvements in data gathering and synthesis, recommendation mechanisms, and the user interface.

6.2.1. Improvements Knowledge Graph Construction

6.2.1.1. Enrichment of Data Representation

Currently, data retrieval is conducted in a static manner from the SkillsFuture website. To enrich the existing data and enhance the representation of the knowledge graph's relevance to the present labour market dynamics, more efforts could be directed towards incorporating insights from noisy, unstructured job postings. This augmentation ensures that the knowledge graph captures the real-time state of the labour market, fostering a more comprehensive and current understanding of the job landscape. This could be possible to done by data mining and scraping the internet for the latest data source.

To achieve a broader perspective and relevance, the knowledge graph can be integrated with other established skills frameworks, such as European Skills/Competencies qualifications and Occupations (ESCO). This interconnectedness amplifies the graph's utility by offering cross-referenced insights into skills, qualifications, and occupations, fostering a holistic comprehension of the job ecosystem.

6.1.1.2. Improvements to Recommendations Algorithms and its Implementation

The recommendation engine acts as an auxiliary feature, enabling users to discover uncharted career paths. While it wasn't the primary focus of the project, we employed a basic model for the recommendation system to help users identify alternative job titles they might consider searching and applying for. A natural extension of this feature will be to couple this job recommendation to actual job posting that closely matches the user current skill level.

For a more tailored and precise model, advanced recommendation methods, as discussed in [Section 2.3.2.](#) of the literature review, can be employed. Significantly, the constructed knowledge graph can be delved into deeper, utilizing graph-based algorithms to enhance the recommendation engine – a method that has gained significant momentum recently. As more user data becomes available, challenges like the cold start problem and data sparsity will diminish. Consequently, A/B testing could be conducted to determine which model best captures user preferences and accuracy. This can help us offer practicality and relevance of the recommendations allowing for a more user-centric approach. If integrated with actual job posting, more advanced metrics like Mean Average Precision at k (MAP@k), Mean Reciprocal Rank (MRR), click-through rates can be evaluated on the relevancy of the job recommendations [27]

Additionally, the recommender engine uses a live inference model, which means it processes real-time data for generating recommendations [15]. While this offers up-to-date and relevant suggestions, it further compounds the response time, especially without a caching mechanism. Every user request triggers the engine and the live inference model, leading to notable delays. To enhance both accuracy from the live inference and the system's responsiveness, introducing a caching strategy can dramatically boost the recommendation engine's performance.

6.2.2. Future Enhancement: Bridging Skills Gap with Actionable Steps

While the application currently offers users a clear view of their progress and roadmap, it lacks specific calls to action regarding the next steps after identifying their skill gaps. A crucial feature to prioritize next would be the introduction of courses tailored to address the specific areas users are deficient in, helping them convert insights into actionable steps.

6.2.3. Enhancing Testing and Deployment Pipeline

Given the time constraints faced during the development of this web application, only manual testing and unit testing was incorporated and Devops was mainly used to automatically deploying new changes to the application. For enhancing the robustness of our CI/CD deployment pipeline, it is imperative in future iterations to prioritize the integration of automated end-to-end tests. Such additions will not only streamline the deployment process but also ensure higher reliability and code quality.

Bibliography

- [1] Young NTUC, "NTUC Youth Taskforce Report 2023," 2023. [Online]. Available: <https://youthtaskforce.sg>.
- [2] M. de Groot, J. Schutte, and D. Graus, "Job Posting-Enriched Knowledge Graph for Skills-based Matching," arXiv:2109.02554v1 [cs.IR], Sep. 6, 2021.
- [3] A. Hogan, E. Blomqvist, M. Cochez, C. D'amato, G. De Melo, C. Gutierrez, S. Kirrane, J. E. Labra Gayo, R. Navigli, S. Neumaier, et al., "Knowledge Graphs," *ACM Computing Surveys*, vol. 54, no. 4, pp. 1–37, 2021, doi: 10.1145/3447772.
- [4] Hao, Xuejie & Ji, Zheng & li, Xiuhong & Yin, Lizeyan & Liu, Lu & Sun, Meiying & Liu, Qiang & Yang, Rongjin. (2021). Construction and Application of a Knowledge Graph. Remote Sensing. 13. 2511. 10.3390/rs13132511.
- [5] Y. Fettach, M. Ghogho and B. Benatallah, "Knowledge Graphs in Education and Employability: A Survey on Applications and Techniques," IEEE, 2022.
- [6] L. Pasquale, de G. Marco, and S. Giovanni, "Content-based Recommender Systems: State of the Art and Trends," in *Recommender Systems Handbook*, 2011, pp. [pages, if provided], doi: 10.1007/978-0-387-85820-3_3.
- [7] X. N. Lam, T. Vu, T. D. Le, and A. D. Duong, "Addressing cold-start problem in recommendation systems," in Proceedings of the 2nd international conference on Ubiquitous information management and communication, 2008, pp. 208-211.
- [8] B. Lika, K. Kolomvatsos, and S. Hadjiefthymiades, "Facing the cold start problem in recommender systems,"
- [9] R. Pradhan, J. Varshney, K. Goyal, and L. Kumari, "Job Recommendation System Using Content and Collaborative-Based Filtering," in *International Conference on Innovative Computing and Communications*, A. Khanna, D. Gupta, S. Bhattacharyya, A. E. Hassanien, S. Anand, and A. Jaiswal, Eds. Singapore: Springer, 2022, vol. 1387, Advances in Intelligent Systems and Computing. [Online]. Available: https://doi.org/10.1007/978-981-16-2594-7_47

- [10] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in Proceedings of the 10th international conference on World Wide Web, 2001, pp. 285-295.
- [11] Q. Guo, F. Zhuang, C. Qin, H. Zhu, X. Xie, H. Xiong, and Q. He, "A Survey on Knowledge Graph-Based Recommender Systems," arXiv:2003.00911 [cs.IR], Feb. 28, 2020. [Online]. Available: <https://doi.org/10.48550/arXiv.2003.00911>
- [12] S. D. Sondur, A. P. Chigadani, and S. Nayak, "Similarity Measures for Recommender Systems: A Comparative Study," Journal for Research, vol. 02, no. 03, pp., May 2016, ISSN: 2395-7549.
- [13] J. Kanjilal, "A Deep Dive into the Back-End for Front-End Pattern," CODE Magazine, March/April 2022. [Online]. Available: <https://www.codemag.com/Article/2203081/A-Deep-Dive-into-the-Back-End-for-Front-End-Pattern>. [Accessed: October 12, 2023].
- [14] "Strategy," Refactoring.Guru, [Online]. Available: <https://refactoring.guru/design-patterns/strategy>. [Accessed: October 11, 2023].
- [15] J. Foo, "Deploying Job Recommender Systems in Production," DSAID GovTech, Dec. 8, 2021. [Online]. Available: <https://medium.com/dsaid-govtech/deploying-job-recommender-systems-in-production-9aaed4c7ffc7>.
- [16] "What is Vercel?," Vercel Blog, [Online]. Available: <https://vercel.com/blog/what-is-vercel>. [Accessed: October 11, 2023].
- [17] "What is DevOps?," Amazon Web Services, Inc., [Online]. Available: <https://aws.amazon.com/devops/what-is-devops/>. [Accessed: October 11, 2023].
- [18] "What is CI/CD?," Red Hat, Inc., [Online]. Available: <https://www.redhat.com/en/topics/devops/what-is-ci-cd>. [Accessed: October 11, 2023].
- [19] M. del Carmen Rodríguez-Hernández, R. del-Hoyo-Alonso, S. Ilarri, R. M. Montañés-Salas, and S. Sabroso-Lasa, "An Experimental Evaluation of Content-based Recommendation Systems: Can Linked Data and BERT Help?," in *2020 IEEE/ACS 17th International Conference on Computer Systems and Applications (AICCSA)*, Antalya, Turkey, 2020, pp. 1-8.
- [20] Z. Hvarlingov, "bert-for-recommendation," GitHub, Jul. 3, 2021. [Online]. Available: <https://github.com/ZdravkoHvarlingov/bert-for-recommendation>

- [21] D. Lavi, V. Medentsiy, and D. Graus, "conSultantBERT: Fine-tuned Siamese Sentence-BERT for Matching Jobs and Job Seekers," arXiv preprint arXiv:2109.06501, 2021.
- [22] C.S. Perone, "Machine Learning :: Cosine Similarity for Vector Space Models (Part III)", Dec. 09, 2013, [Online], Available: <http://blog.christianperone.com/2013/09/machine-learning-cosine-similarity-for-vectorspace-models-part-iii/>
- [23] G. A. Ekainu, "Next.js vs React: The Difference and Which Framework to Choose," Ninetailed, April 1, 2023. [Online]. Available: <https://ninetailed.io/blog/next-js-vs-react/>. [Accessed: October 11, 2023].
- [24] "Neo4j Aura Documentation," Neo4j, [Online]. Available: <https://neo4j.com/docs/aura/>. [Accessed: October 11, 2023].
- [25] "Relational to Graph Data Modeling," Neo4j, [Online]. Available: <https://neo4j.com/developer/relational-to-graph-modeling/>. [Accessed: October 11, 2023].
- [26] Paulo Gomes. Unit Testing and the Arrange, Act and Assert (AAA) Pattern. url: <https://medium.com/@pjbgrf/title-testing-code-ocd-and-the-aaa-pattern-df453975ab80>. [Accessed: October 13, 2023].
- [27] Y.-M. Tamm, R. Damdinov, and A. Vasilev, "Quality Metrics in Recommender Systems: Do We Calculate Metrics Consistently?" arXiv preprint arXiv:2206.12858, Jun. 2022. [Online]. Available: <https://doi.org/10.48550/arXiv.2206.12858>