TECHNOLOGICAL UNIVERSITY DUBLIN

Literature Review of Just Apply

b00150058 – Daniel Aigbe

Supervisor: Luke Raeside

November 13, 2025

Literature Review

in the

School of Computing

Department of Informatics and Cyber Security

November 2025

Contents

Declaration of Authorship

I, Daniel Aigbe, declare that this thesis titled *"Literature Review of Just Apply"* and the work presented in it are my own. I confirm that:

- This work was completed wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this has been clearly attributed.
- Where I have quoted from the work of others, the source has always been given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done jointly with others, I have clearly indicated what was done by others and what I have contributed myself.

Signed: Daniel Aigbe

Date: 14/10/2025

1 Abstract

This literature review investigates how Natural Language Processing, cloud computing, and data analytics support the development of the intelligent job recommendation system of the Just Apply project. It assesses how NLP processes unstructured text from CVs and job descriptions in order to extract skills, experience, and qualifications. It reviews the cloud technologies that enhance scalability, storage, and system reliability for recruitment platforms. Additionally, it analyzes how data analytics identifies labour market trends, skill gaps, and personalized recommendation patterns. The review has noted current challenges in automation, accuracy, and system integration within the paradigm of recruitment technologies. It points out several knowledge gaps in the existing literature, such as restricted use of lightweight NLP models, inconsistent adoption of cloud technologies in the field of recruitment system management, and minimum analytics-driven integration into real-time recommendation workflows. In summary, findings illustrate that integrating NLP, cloud infrastructure, and analytics ensures greater matching accuracy, system performance, and user experience for both job seekers and administrators. These insights thus endorse the design of Just Apply and present ground for future work on efficient, scalable, and fair recruitment solutions.

2 Introduction

2.1 Background and Motivation

The increasing utilization of digital technologies in employment platforms has revolutionized the way job seekers and employers get connected. Traditional job application systems are usually based on static keyword searches and manual filtering, normally incapable of capturing richer semantic relationships between candidates' skills and job requirements. This leads to poor matching accuracy and inefficient user experiences. With the ever-increasing volumes of online job data and applicant information, there is a need for intelligent systems that would understand context, process unstructured text, and generate personalized job recommendations.

Recent developments in NLP, Cloud Computing, and Data Analytics have enabled the creation of intelligent, scalable, and data-driven platforms for the automation of processes

related to job–candidate matching. NLP enables text data analysis, like résumés and job descriptions, information extraction, and determination of semantic similarities between candidate profiles and job listings. Cloud Computing provides infrastructure for hosting such large-scale systems, ensuring scalability, real-time data access, and cross-platform integration. Meanwhile, Data Analytics allows pattern and insight extraction from user interactions and labor market data, which enhances the system's recommendation accuracy and adaptability over time.

The proposed project, *Just Apply*, seeks to combine all these technologies into a cloud-based intelligent job recommendation system. It therefore seeks to simplify the process through the analysis of user input, job descriptions, and requirements using NLP while leveraging cloud infrastructure for scalability and storage. This is further integrated with Data Analytics to perform real-time analyses of trends, thus making recommendations that closely relate to the preferences, experience, and qualifications of each user. This project responds to an increased need for intelligent recruitment tools that go beyond simple keyword-matching approaches by leveraging sophisticated algorithms combined with scalable cloud resources to provide more relevant and user-specific results. This review derives its motivation from an understanding of how these three technological domains intersect to improve the accuracy, scalability, and overall performance of job recommendation.

2.2 Aim and Objectives of the Review

This literature review has been developed to analyze and synthesize the existing research related to the application of NLP, Cloud Computing, and Data Analytics in developing intelligent job recommendation systems. Rather than organizational or HR perspectives, the emphasis of this review is on software engineering, computational, and system architecture aspects relevant to the design and implementation of
textitJust Apply.

The specific objectives of this review are:

- To examine the use of NLP algorithms in extracting, classifying, and matching skills, experiences, and job requirements.
- To investigate how cloud-based infrastructures support scalable, reliable, and accessible recommendation systems.

- To explore the application of data analytics techniques for improving personalization, performance, and predictive accuracy in job recommendation models.
- To identify existing gaps and limitations in current systems that justify the development of an integrated platform like *Just Apply*.

These goals provide the basis to assess in what ways existing studies establish or constrain the development of a next-generation job recommendation platform. The review will further establish how the integration of NLP, cloud computing, and analytics can be used to provide a seamless and efficient user experience for both job seekers and platform administrators.

2.3 Scope and Structure

This review focuses on scholarly and peer-reviewed research published within the period of 2012-2025 in the domains of NLP, cloud computing, and data analytics, with particular emphasis on their application in job matching, information retrieval, and intelligent recommendation systems. Non-academic sources and materials unrelated to computational methods or system implementation are beyond the scope of this review.

This literature review is structured thematically, based on the core technologies underpinning the
textitJust Apply system. Section 2 delves into how NLP techniques are applied in job recommendation systems, describing semantic analysis, text classification, and skill extraction. Section 3 covers cloud computing technologies that enable system scalability, storage, and deployment. Section 4 discusses data analytics methods applied to user profiling, trend detection, and recommendation optimization. Section 5 brings together insights from all three domains, focusing on the gaps, challenges, and opportunities presented, and the way these inform the architecture of
textitJust Apply. The final section, Section 6, outlines key findings and implications for future system development.

By centering the review around these technological pillars, the paper ensures coherence with the objectives of the
textitJust Apply project and maintains the focus on computational and engineering contributions for intelligent, cloud-based job recommendation systems.

3 Application of Natural Language Processing (NLP) in Recruitment Systems

3.1 Overview of NLP in Recruitment

Application of natural language processing is central to the
textitJust Apply web application. The system employs NLP for reading and comprehension of uploaded CVs for the identification of skills, experience, and qualifications. Traditional job search systems rely on manual filtering or simple keyword searches that often fail to retrieve relevant results. They cannot understand language variations or contextual expressions. NLP solves this by processing unstructured text and identifying useful information automatically (Otani et al., Devaraju, Ertuğrul, Bitirim, 2025).

The Flask backend works as an intermediary between the NLP module and the web interface made in React. Once the user has uploaded his or her CV, the Flask server processes the file by extracting the text that would need to be analyzed. This is cleaned and processed using Python libraries such as NLTK or spaCy. These libraries help remove stop words, punctuation, and redundant symbols. The essence of doing so is to prepare clean text which can be compared efficiently with job descriptions kept in the system database. Such a simple approach works for lightweight web applications since it requires minimal resources and gives fast results (Otani et al., 2025).

NLP processing in
textitJust Apply does not rely on sophisticated deep learning models. Instead, it focuses on keyword-based and rule-based methods that are easy to maintain. This design is in line with the objectives of the project, which seek to provide an effortless web app that serves the needs of job seekers. The analytics module utilizes the extracted data on skills and experience for purposes of recommending the best jobs. The recommendations are then fed into the user interface through the React frontend. This ensures end-to-end connection between the user and the backend Flask to the database (Ertuğrul Bitirim, 2025).

3.2 CV Parsing and Skill Extraction Techniques

CV Parsing and Skill Extraction Techniques CV parsing is the first technical step in textitJust Apply. It converts a CV into structured data that the system can store in its SQLite or MySQL database. Python's text-processing libraries extract text from PDF or DOCX files and split it into sentences and tokens. Each token is analyzed to identify skills, education, and experience. Techniques such as part-of-speech tagging and keyword matching are used to classify each term correctly. For example, when the system finds words such as "Python," "React," or "data analysis," they are recorded as skills. Dates and company names are categorized under experience (Devaraju, 2022).

Another important data used in this parsing process is a set of common skill keywords that is kept in the database. It helps the NLP engine in recognizing both technical and soft skills of the candidate during analysis. This list is loaded in the Flask backend during processing, which maps tokens from the CV against this. If there is a match, that skill is added to the candidate's profile. This kind of keyword-based matching is precise for structured lists of skills and quick enough to be deployed on a Flask server for real-time applications without adding noticeable latency to the user experience (Otani et al., 2025).

After extracting the data, the results are stored in structured format. It enables the analytics module to query the database and compare candidate profiles against job descriptions. The parsed data also helps in displaying various insights to users, such as missing skills or preferred job categories. Combining the simple NLP with structured storage allows

textitJust Apply to create a clear bridge between raw text and actionable job recommendations, (Ertuğrul Bitirim, 2025).

3.3 Semantic Similarity and Matching Approaches

In

textitJust Apply, the matchmaking relies solely on the textual content of a candidate's CV, which is matched with job descriptions stored in the database by shared keywords and term frequency. When a candidate uploads a CV to the system, the system extracts skill keywords and counts how many of them appear in each job description. The more overlap between them, the higher the match score. This method is simple yet effective for applications relying on structured text. The comparison logic is run from the Flask

backend, which also calculates the similarity scores of all the available job entries. After sorting these results, they are sent to the React frontend and presented to the user. Every job listing includes a relevance percentage along with the list of matching skills. Techniques such as term frequency analysis and cosine similarity have been used to determine the closeness between a CV and the job description. These are some basic, yet reliable NLP techniques that can be used in the case of web-based systems developed through Python (Otani et al., Devaraju, Ertuğrul, Bitirim, 2025).

The Flask backend runs the comparison logic and calculates similarity scores for all available job entries. These results are sorted and sent to the React frontend, where they are displayed to the user. Each job listing shows a relevance percentage and a list of matching skills. The approach uses techniques such as term frequency analysis and cosine similarity to determine how closely a CV matches a job description. These are basic yet reliable NLP methods suitable for web-based systems developed in Python (Devaraju, 2022).

To further enhance precision, the system also filters out common or irrelevant words, such as "team," "project," or "experience." This process ensures that only important technical or professional terms will be included in the final score. In addition, this analysis is conducted in small batches through Flask routes to keep performance at its best, whereas the SQL database efficiently handles the search results. This approach keeps the computation fast and lightweight, suitable for deployment on Azure cloud infrastructure (Ertuğrul Bitirim, 2025).

Future versions of

textitJust Apply can further expand the matching logic through including support for synonyms. This would let the system understand that words like "developer" and "programmer" refer to similar jobs. Python's integrated synonym detection libraries, such as WordNet, can be embedded in the Flask NLP module without adding any extra complexity. Such future enhancements are recommended in consistency with the long-term vision to keep improving the system's performance while sustaining simplicity in nature. (Otani et al., 2025).

3.4 Challenges and Limitations

There are a couple of challenges in conducting NLP for job matching within a lightweight web application. First, data inconsistency poses the first challenge since CVs generally come in many different formats and job descriptions have varying styles. This affects the accuracy of text extraction. Converting PDF and DOCX into readable text without errors is critical. Python libraries can handle most cases, but from time to time differences in format cause several words to be either missing or repeated (Devaraju, 2022).

Another challenge is the limited context of keyword-based matching. Simple NLP models cannot fully understand sentence meaning. For example, a candidate may mention "led a project using Java," but the system may only detect "Java" as a skill and ignore the leadership aspect. While deeper language models could fix this, they are too complex for a Flask-based project running on small-scale cloud resources. The system, therefore, focuses on consistency and speed instead of deep semantic understanding as would be advanced (Otani et al., 2025).

Another consideration is performance. With a number of users, the application will have multiple requests, and the handling of NLP has to remain efficient. This can be computationally heavy, especially in the processing of large-sized CV files, or even worse, the uploading of many files simultaneously. Optimized tokenization used in this application minimizes such negative impact, as well as repeated computations because of caching. This balances real-time performance and accuracy of the system (Ertuğrul Bitirim, 2025).

Data privacy and security are critical as well. CVs contain personal information regarding contact details or work experience. The
textitJust Apply system uses Azure cloud storage and encrypted SQL databases to protect this information. Access is restricted through authentication routes in Flask, and sensitive data is removed prior to processing. This approach aligns with secure design practices found in similar research regarding cloud-hosted recruitment tools (Otani et al., 2025).

Finally, the effectiveness of NLP depends on the quality of the job dataset. This means that the system should always keep a clean and updated list of job postings in order to bring appropriate results. Outdated or inconsistent data reduces recommendation accuracy. Continuous updates and validation of job entries ensure users get current and accurate job suggestions at all times (Devaraju, 2022). Summarizing, the NLP part of textitJust Apply is implemented using practical and effective techniques that appeal to

the purpose of the project. The approach has avoided over-complication while yielding useful results. The application generates relevant job recommendations using keyword extraction, CV parsing, and matching based on word frequency in real time. Such methods are realistic for a Flask-based Python web application hosted on Azure integrated with a SQL database (Otani et al., Devaraju, Ertuğrul, Bitirim, 2025).

4 Cloud Computing in Recruitment Platforms

4.1 Cloud-Based HR and Recruitment Systems

Cloud computing forms the basis of the deployment and operationalization of textitJust Apply. The system relies on cloud services for data storage of users, hosting applications, and request processing. Cloud platforms allow web applications to operate without the limitation of local servers. Thus, job recommendations are efficiently delivered to the users anywhere with access to the internet (Otani et al., Devaraju, Ertuğrul, Bitirim, 2025).

Traditional recruitment platforms needed large on-premise systems. These were expensive to buy and difficult to maintain. Cloud computing made many things easier by offering flexible, pay-as-you-go resources. For lightweight web applications such as textitJust Apply, hosting in the cloud made deployment and scaling much easier. The application was implemented using a Flask backend and a React frontend connected to a cloud database. The Flask API processes user requests, while the database stores parsed CV data, job descriptions, and analytics. The cloud makes sure these components communicate smoothly and remain available for the users, (Ertuğrul Bitirim, 2025).

They are also easier to upgrade and monitor. The developers can push upgrades directly to the server instead of reinstalling the software locally. This is important for the student-built system
textitJust Apply, because iterative development and testing continue. Azure's dashboard allows performance monitoring, tracking of storage used, and the management of traffic. In-built tools like these reduce maintenance work and help detect problems in the system at an early stage. Other studies indicate that this allows cloud platforms to enhance the reliability and minimize operating costs in small web systems (Devaraju 2022)

4.2 Scalability, Security, and Data Management

One of the key benefits of cloud computing is scalability. The number of users uploading their CVs or browsing for jobs can increase over time. A cloud-based setup ensures that *Just Apply* will handle such growth. Azure supports both horizontal and vertical scaling. This means that the system can add more computing power or duplicate services in case traffic goes up. Complementing this, Flask is lightweight in design and handles requests in an efficient manner with low resource usage (Otani et al., 2025).

Of course, another important factor is data security. CVs and job postings contain personal data that is sometimes sensitive. The
textitJust Apply system will be using Azure's secure storage/ database services to keep this data secure. All the files that a user uploads are kept in encrypted format. Authentication routes in Flask ensure that only registered users can access their data. Access permissions for each component are limited with Azure's access control. This structure prevents unauthorized access or data leaks (Devaraju, 2022).

Effective data administration is highly important for proper job recommendations. The application will store the cloud database of CV data, extracted skills, and job postings. It uses Azure SQL Database to store structured data, like the parsed skills and matching results. The Flask backend uses SQLAlchemy to interact with the database. This arrangement, therefore, means efficient data retrieval and reduces the need for local storage of data. In addition, it makes it easier to back up and restore data if needed. Centralized data management in cloud databases enhances the reliability and decreases redundancy, as seen in various studies (Ertuğrul Bitirim, 2025).

The cloud environment also provides automated backups and recovery options. This ensures that user data can stay safe even in the case of unexpected system failure. The redundancy incorporated within Azure shields against hardware or networking failures. This is quite vital for Web applications needing high uptimes and assurance of data integrity. The application of such features in
textitJust Apply helps to assure reliability and security in its services to users (Otani et al., 2025).

4.3 Microsoft Azure and Cloud Integration in Recruitment

Just Apply uses Microsoft Azure as its primary cloud service provider. Azure offers a number of services that support the project's architecture. The Flask backend is deployed using Azure App Service, which allows autoscaling and continuous deployment. This allows updates to the system to be pushed directly from development tools without any downtime. The React frontend can be hosted using Azure Static Web Apps, offering rapid access by users through a global content delivery network (Otani et al., Devaraju, Ertuğrul, Bitirim, 2025).

This system supports Azure Blob Storage for managing files. It stores uploaded CVs securely and provides direct access through API calls. Flask interacts with Blob Storage through Azure SDK for Python. It helps to retrieve the files fast when it is to be used for NLP processing. Use of cloud storage reduces dependency on local servers and makes the system more flexible (Devaraju, 2022).

The *Just Apply* database layer runs on Azure SQL Database. This service enables structured data storage related to user accounts, job listings, and skill data. Furthermore, Flask has an integration with SQLAlchemy, which seamlessly enables access to data. Finally, the Azure portal offers different insights into performance that help in maintaining the efficiency of the database so the system is able to withstand workloads both presently and in the future.(Ertuğrul Bitirim, 2025).

The analytics tools on Azure are also in agreement with data analytics goals of the project. The cloud environment supports the dashboards that track the end-user activities, search behaviors, and system performance. These give insight into improving the recommendation and maintaining responsiveness. Further enhancements can be done using Azure Monitor and Application Insights to log error, performance, and usage pattern, for further system optimization (Otani et al., 2025).

Azure's security framework has built-in protection for web applications through firewalls, data encryption, and DDoS protection. Flask routes and database connections use secure protocols to meet these standards. This assures that data traveling between the user interface and server and database is always encrypted. Such measures also lead to the project objective of providing a secure and scalable environment to job seekers. (Devaraju, 2022).

4.4 Identified Challenges and Research Opportunities

While cloud integration provides strong advantages, it also introduces challenges. Cost management is one of the main concerns. Azure's pay-per-use model may be expensive if resource usage increases unexpectedly. The instance sizes, database calls, and storage traffic need to be managed in order to control costs. For student projects like *Just Apply*, free or low-tier service plans are enough to manage their budgets effectively (Otani et al., 2025).

The other challenge is the dependency on the cloud environment. It leads to limited access to the application when Azure services are down. The risk can be reduced by building fallback mechanisms, such as local caching or mirrored backups of data. In such cases, Flask will handle cached responses while the cloud service recover. This design keeps users connected even during brief outages (Ertuğrul Bitirim, 2025).

Security management also requires continuous attention. Although Azure provides strong default protections, misconfiguration can expose sensitive data. Developers must monitor access permissions, database credentials, and API keys regularly. Using environment variables and secret management tools is recommended to avoid leaks. These practices maintain data privacy and comply with security standards (Devaraju, 2022).

Security management also requires ongoing attention. Even though Azure has very strong default protections against data exposure caused by misconfigurations, it is the developer's job to keep an eye on access permissions, database credentials, and API keys. This is where environment variables are encouraged and secret management tools should be used in order to avoid leaks. Such practices will maintain data privacy and follow security standards accordingly. (Otani et al., 2025).

Other opportunities involve data analytics integration. Azure is able to provide the analytics and AI tools to drive detailed insight into job trends and user activity. While *Just Apply* focuses on skill matching today, predictive analytics can add a new aspect of finding out future skill demands or career patterns. These enhancements will make the platform more valuable to job seekers and recruiters alike (Ertuğrul Bitirim, 2025).

In conclusion, the proposed integration of Azure Cloud addresses the flexibility, scalability, and security concerns to support the *Just Apply*. There are challenges, especially with regard to cost control, dependency, and configuration, but its advantages in cloud deployment outweigh the disadvantages. Thus, cloud technologies make it possible for

even lightweight Flask-based applications to deliver consistent, secure, and efficient job recommendations. (Otani et al., Devaraju, Ertuğrul, Bitirim, 2025).

5 Data Analytics and Labour Market Insights

5.1 Role of Analytics in Decision-Making

Data analytics are used to support the decision-making features of *Just Apply*. It converts raw CV and job data into useful insights that guide users toward suitable opportunities. For each CV uploaded, there is a set of extracted skills and experience values. This data, matched against stored job descriptions, computes which roles are best suited for a candidate. The underlying process for this would be the structured analysis of text-based data using Python and SQL queries (Otani et al., Devaraju, Ertuğrul, Bitirim, 2025).

Flask acts to connect the analytics logic to the web interface. Once NLP is processed, the backend calculates basic matches of percentages and skill frequency counts. These results then determine which jobs are shown first. It aims to help users identify jobs that align with their abilities quickly and clearly. This approach can also be useful to developers in monitoring how users interact with the system, which improves performance and accuracy in succeeding updates.(Devaraju, 2022).

Analytics in *Just Apply* go even beyond job matching. It highlights gaps between user skills and job requirements. Whenever some skills appear often in relevant vacancy descriptions but do not figure in the user's CV, it gets tagged by the system as lacking. That way, it points out how one should work on their weak points. Such insights are simple yet practical, showing how data-driven feedback improves the job search experience (Ertuğrul Bitirim, 2025).

Data-driven decision-making also fosters platform performance evaluation. Usage data measured by Flask at the platform would show how long users browse, what jobs they select, and how frequently they return. Azure visualizes the data on dashboards to further analyze it. Usability is maintained by tracking these metrics and ensuring that the platform meets its goal of simplifying job searching (Otani et al., 2025).

5.2 Predictive and Prescriptive Analytics in Recruitment

Predictive analytics mainly concerns the focus of how much existing data is used to estimate future outcomes. In *Just Apply*, this may involve observing which skills or job categories a user is most likely to explore. Flask gathers data from every user interaction and rests it within the SQL database. Python scripts process the data to work out probabilities and trends that could be used to prioritize recommendations for similar users in the future (Otani et al., 2025).

Prescriptive analytics goes one step further and advises on what to do. In *Just Apply*, this means recommending what new skills or certifications to pursue based on recurring patterns within job postings. Using the example above, if most jobs that match a user's background require "data visualization," the system should suggest that skill as an improvement target. This feature strongly supports continuous learning and helps users plan career development based on data, rather than guesswork (Devaraju, 2022).

Predictive and prescriptive analytics depend on simple algorithms in Python, such as frequency counts, ratios, or filtering conditionally. This is efficiently handled by Flask through SQL queries. The result is then shown to the users with the visual elements coming inbuilt in the React frontend to present users with personalized insights without any delay. Studies show that this kind of lightweight analytical model improves system transparency by maintaining high speed in small-scale web applications until the end (Ertuğrul Bitirim, 2025).

Azure enhances these analytics functions with its set of tools. As the application grows, the platform analytics services can process larger datasets. In future versions, the integration of Azure Machine Learning or Power BI could enable the application to perform more complex analyses, such as trend forecasting or career clustering. By doing so, these upgrades extend the system's functionality while maintaining a stable core architecture. (Otani et al., 2025).

5.3 Career Guidance through Data Visualization

Data visualization makes the analytics results clear and accessible to the users. Data was presented in simple, visual form through charts and indicators in *Just Apply*. Match percentages, missing skills, and job categories are visualized with React components for an easy-to-read layout. This direct feedback supports quicker and less confusing decision-

making processes by the user (Otani et al., Devaraju, Ertuğrul, Bitirim, 2025).

In the backend, Flask summarizes the analytics results into data, formats it for visualization, and sends it using RESTful APIs to a React frontend. The summarized results include top skills matching each job, missing skills in count, and job category rankings. Each visualization is generated using basic JavaScript chart libraries integrated into React. This design is efficient and avoids the overhead of complex visualization tools (Devaraju, 2022).

These are hosted and supported by Azure, ensuring that cloud services support instant updating of data across sessions. New data is processed and shown in real time as users interact with the platform; this creates an immersive experience that retains user interest. As argued by (Ertuğrul Bitirim 2025), real-time visualization in a cloud-based web system offers greater user retention and satisfaction.

The visualization process facilitates developers analyzing system accuracy. Whenever a user is shown irrelevant job recommendations, it would come forth from the analytics dashboard as to why. It could range from overrepresentation of certain keywords in the text data to adjustment thresholds in skill matching. These insights make system tuning more practical and data-driven (Otani et al., 2025).

5.4 Limitations and Research Gaps

There are various limitations on the use of analytics for recruitment systems. The first is in regard to the dataset size. Small and outdated datasets result in less accuracy in analytics results. The
textitJust Apply system relies on updated, varied job postings. If this does not happen regularly, the predictive results will lose relevance. A clear-cut plan of data collection would help in sustaining accuracy for a longer period of time (Devaraju, 2022).

Another critical limitation involves data diversity. There are different structures and qualities of job postings, where some contain exhaustive lists of requirements and others are vaguely worded. This inconsistency reduces the precision of predictive and prescriptive models. There is a need for enhancement in data quality by way of pre-filtering or manual review for higher reliability (Otani et al., 2025).

Scalability of the system itself affects analytics performance. While the amount of data grows, simple Python or SQL-based analytics may slow down. Moving these to

Azure's analytical services or distributed computing models can resolve this challenge but involves advanced configuration and cost management. Keeping performance balanced with a budget is one more critical research opportunity for lightweight cloud systems (Ertuğrul Bitirim, 2025).

Also, there are privacy and security issues. Analytics needs the storage of user behavior and CV data that can keep for trend analysis. This data will have to be anonymous to protect the users' identity. This is handled through encrypted communication in Flask, and access control policies in Azure. Up-scaling analytics functions while maintaining these standards is another current challenge (Otani et al., 2025).

Finally, there is an opportunity to explore more interactive career guidance tools. The basic features of the system in use today provide recommendations and missing-skill insights. Future research can investigate integrating external APIs such as LinkedIn Learning or Coursera to enable users to act on these recommendations with directly linked skill courses. Simple analytics and guided learning could make *Just Apply* more effective as a personal career-planning tool (Ertuğrul Bitirim, 2025).

In all, data analytics support operational and decision-making functions at *Just Apply*. It converts user and job data into succinct insights that improve accuracy and usability. Although the current system uses simple analytical methods, future research can definitely scale its predictive and visualization capabilities with advanced services provided by Azure and by using external integrations. (Otani et al., Devaraju, Ertuğrul, Bitirim, 2025).

6 Integration of Technologies and Research Gaps

6.1 Synergy Between NLP, Cloud Computing, and Analytics

The effectiveness of *Just Apply* depends on the combined use of Natural Language Processing, Cloud Computing, and Data Analytics. These three technologies work together to deliver fast, accurate, and scalable job recommendations. Each technology supports a specific layer of the application: NLP handles reading and interpretation of CVs, Cloud Computing provides for performance, availability, and data security, while Data Analytics turns processed text into insights and recommendations (Otani et al., Devaraju, Ertuğrul, Bitirim, 2025).

Flask integrates these components into a single system. Upon the upload of a CV by a user, the backend extracts the skills and experience by using NLP. The results are securely stored in the Azure SQL Database. The data is then matched against job postings by the analytics module, which will compute the match score. After computation, results are sent to the React frontend for display. Each step depends on cloud infrastructure for speed and storage efficiency (Devaraju, 2022).

The synergy between these technologies provides for continuous learning. The analytics layer records user behavior, feeding new insights into the NLP module. This feedback positively influences the accuracy as time goes by. The process is supported by Azure's cloud resources: it handles larger volumes of work without decreasing system speed. This collaboration between the modules secures that *Just Apply* runs well even when data volume and user activity grow high (Ertuğrul Bitirim, 2025).

This integration allows for modular development. Each component in a system can be updated or replaced without interfering with the others. For example, the NLP module could be upgraded to include new keyword libraries without making any changes to the analytics or cloud setup. Flask's blueprint structure and RESTful APIs make these modules communicate seamlessly with each other. This modular design keeps maintenance simple and allows for future expansions on the system. (Otani et al., 2025).

6.2 Identified Gaps in Existing Studies

Most of the literature in the domain of job recommendation systems discusses separate technologies rather than integrated solutions. So many discuss either an NLP-based resume parsing system or an analytics-driven approach to matching. Few papers discuss how these technologies might function collectively on a single, lightweight web application. This limits practical use for smaller or academic projects which cannot support complex architectures as discussed (Otani et al., Devaraju, Ertuğrul, Bitirim, 2025).

Other gaps include scalability in academic implementations. Many research works have presented prototypes that may be tested on small datasets and very limited deployment. There is little focus on integration with the cloud or how the system performs when exposed to real web traffic. *Just Apply* addresses this by deploying its modules through Azure's cloud platform, allowing dynamic scaling and continuous operation (Ertuğrul Bitirim, 2025).

Data privacy and ethics are also minimally discussed in related works. Most of the systems analyze personal information without discussing storage options or how the information is secured. These raise privacy concerns for the individuals involved. In contrast, *Just Apply* utilizes encrypted Azure storage along with Flask authentication routes to maintain data protection. The design of the system counts privacy and compliance as features rather optional features (Devaraju, 2022).

Another gap involves the combination of technical and user-oriented analytics. Some studies assess the accuracy of a system without considering user experiences. A recommendation system, in realistic practice, should monitor user interactions with results and adjust future suggestions based on those interactions. User behavior tracking is also developed as part of the *Just Apply* analytics module to refine job recommendations. This feature bridges the gap between technical performance and real user needs (Otani et al., 2025).

Few projects target missing skill identification. Most of the systems stop at matching candidates for jobs without necessarily showing what they lack. *Just Apply* analytics integrate to identify missing skills and highlight areas for improvement. This function really guides people rather than passively matching them. It reflects the goal of the whole project: helping users understand their strengths and areas for growth.

6.3 Relevance to the Just Apply Project

The combination of NLP, Cloud Computing, and Data Analytics directly contributes to the objectives of the

textitJust Apply project. It is envisioned as a fast, web-based engine for CV processing, skill analysis, and job recommendations with high accuracy. Each of these technologies serves to meet these needs. NLP extracts structured data from text. The cloud provides storage, security, and accessibility. Analytics transform data into meaningful feedback and visual insights (Otani et al., Devaraju, Ertuğrul, Bitirim, 2025).

The combined power of Flask, Azure, and Python libraries enables all the modules to work in a single environment: Flask manages API routes between the NLP engine, analytics logic, and SQL database; Azure ensures uptime and integrity of data; while React presents the results visually for better user engagement. This architecture reflects the practical focus of the system and avoids unnecessary complexity (Devaraju, 2022).

The research reviewed shows that integration improves both accuracy and performance. Systems using these technologies together deliver faster processing with more relevant recommendations. The modular structure of *Just Apply* supports this conclusion, showing that even the simplest of tools can be connected to achieve high functionality. This fits the educational and practical goals of the project with good scalability for future development (Ertuğrul Bitirim, 2025).

The work of *Just Apply* contributes to academic understanding by showing how an integrated cloud-based recruitment system can be built using accessible tools that provide an example of how an opensource framework such as Flask and React can be combined with commercial cloud services to offer a secure, data-driven solution. Future research can easily extend this approach by testing advanced analytics and larger datasets while retaining the same lightweight structure (Otani et al., 2025).

7 Conclusion

7.1 Summary of Key Findings

From the reviewed literature, it is evident that NLP, cloud computing, and data analytics are pivotal technologies behind any modern job recommendation system. Each technology plays the role of enabling a specific function in
textitJust Apply. NLP reads and processes user CVs. Cloud Computing provides hosting, scalability, and data security. Data Analytics delivers insight through skill comparisons and job recommendations. Combined, these technologies create a structured and efficient workflow that improves accuracy and user experience. (Otani et al., Devaraju, Ertuğrul, Bitirim, 2025).

For instance, NLP for *Just Apply* allows it to extract important skills and experiences of applicants from their uploaded CVs. Techniques for cleaning text and matching keywords will give good results without much computational overhead. Such techniques align with the project focus of remaining light and accessible in design. This application integrates Cloud Computing via Microsoft Azure as the glue that combines these parts-Flask and React-appropriately. Azure facilitates real-time processing with secure storage and maintenance. Azure SQL Database is used for fast and efficient data access. Data Analytics for all these components puts them together: matched jobs, missing skills, and

career trends. All of these put together make *Just Apply* a functional and scalable web application (Devaraju, 2022).

The review further supports the fact that integrated systems outperform isolated modules. Applications featuring combined NLP, cloud, and analytics technologies indeed report speedier and more accurate results. The reviewed research justifies the project structure and further consolidates its design choices. It is also underscored that practical systems do not require complex models of artificial intelligence, but rather effective integration and data management drive success (Ertuğrul Bitirim, 2025).

7.2 Implications for Future Research

The review identifies several areas for future development. First, data quality and diversity remain key challenges. Future work should focus on building larger, verified datasets of job postings to improve matching precision. Continuous updates will ensure that users receive relevant results. Second, there is potential to expand analytics functions. Predictive models can be added to track job trends or forecast future skill demands. These enhancements would strengthen the decision-support capabilities of *Just Apply* (Otani et al., 2025).

Another opportunity is the integration of external APIs. Connectivity with professional networks or learning platforms would enable users to directly apply for jobs or improve missing skills. These would make the application more interactive and useful. Further research could also investigate other cloud configurations to compare the cost and performance across service tiers. This would guide developers in optimizing deployments for academic or commercial use. (Devaraju, 2022).

Security and privacy will continue to be paramount. Systems such as *Just Apply* deal with sensitive user data, so future works should consider advanced encryption, access control, and compliance standards. User trust and transparency should be investigated by researchers, who also need to assure fairness and non-bias in analytics. Finally, there is space for the investigation of collaborative or community-driven datasets, allowing users to contribute feedback and enhance the accuracy of job recommendations over time (Ertuğrul Bitirim, 2025).

The overall review, therefore, supports the design of *Just Apply* as a lightweight, scalable, data-driven job recommendation platform. The findings confirm that with simple

NLP, cloud integration, and analytics, the developed foundation will be strong for any future research and system development (Otani et al., Devaraju, Ertuğrul, Bitirim, 2025).

References

Devaraju, S. (2022). Natural Language Processing (NLP) in AI-driven recruitment systems. International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 8(3), 555–566.

Ertuğrul, D. Ç., Bitirim, S. (2025). Job recommender systems: A systematic literature review, applications, open issues, and challenges. Journal of Big Data, 12(140). https://doi.org/10.1186/s40537-025-01173-y

Joshi, M. (2024). Reinvigorating human resource management through cloud computing: A systematic review of literature and bibliometric perspective. Journal of Scientific and Industrial Research, 83(5), 490–499.

Madanchian, M. (2024). From recruitment to retention: AI tools for human resource decision-making. Applied Sciences, 14(24), 11750. https://doi.org/10.3390/app142411750

Maqueira Marín, J. M., De Oliveira Dias, D., Jafari Navimipour, N., Gardas, B. (2022). Cloud computing and human resource management: Systematic literature review and future research agenda. Kybernetes, 51(6), 2172–2191. https://doi.org/10.1108/K-05-2021-0420

Otani, N., Bhutani, N., Hruschka, E. (2025). Natural language processing for human resources: A survey. In Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Industry Track), 583–597.

Sharma, D., Salehi, W., Bhardwaj, B. (2024). Dovetailing the human resource management with the cloud computing in the era of Industry 4.0: A review. Frontiers in Management and Business, 4(2), 340–351. https://doi.org/10.25082/FMB.2023.02.004