Job Vacancy Recommendation System Using Jaccard Method on Graph Database

Saiful Riza[1]*, Wahyu Fuadi[2], Yesy Afrillia[3]
Universitas Malikussaleh[1], [2], [3]
Aceh Utara
Saiful.180170101@mhs.unimal.ac.id[1], wahyu.fuadi@unimal.ac.id[2], yesyafrillia@unimal.ac.id[3]

***Abstract*— In the rapidly evolving digital era, recommendation systems play a crucial role in helping users discover relevant information aligned with their preferences. PT Nirmala Satya Development, a company engaged in psychology and human resource development, faces challenges in utilizing big data consisting of 500 applicants, 500 job postings, and 500 job applications to generate accurate and relevant job recommendations. This study develops a job recommendation system using the *Jaccard Coefficient* method to measure similarity between users based on their job application history, implemented within a Neo4j graph database. The system models the relationships between entities through nodes and edges, allowing dynamic analysis using the *Cypher Query Language*. Testing on 237 users demonstrated that the majority received at least one relevant recommendation, with recall values often reaching 1.0, especially among users who had a single job target. The system achieved precision values ranging from 10% to 20%, which is considered acceptable given that ten recommendations are generated per user. The highest F1-score reached 0.33, although some users received F1 = 0 due to limited application history or unique preferences. Overall, the system effectively delivers personalized and efficient job recommendations, particularly for active users. This research also proves that combining the Jaccard Coefficient with a graph database structure is a powerful approach to representing and analyzing complex relationships between users and job postings in a modern recruitment platform.**

Keywords*— Graph Database, Jaccard Coefficient, Neo4j, Recommendation System

I. INTRODUCTION

In the rapidly evolving digital era, recommendation systems have become a crucial component in helping users find information that aligns with their needs and preferences. PT Nirmala Satya Development, a company specializing in psychology and human resource development, faces significant challenges in leveraging applicant, job, and application data to create an effective job recommendation system. Although large amounts of data are already available, the existing system has not yet fully optimized its potential to deliver relevant job recommendations.

One commonly adopted approach in recommendation systems is collaborative filtering, which is divided into two main types: user-based and item-based. In the user-based approach, recommendations are made based on behavioral similarities between users, while the item-based approach relies on the similarity between items that users have interacted with, such as job postings they have applied to [1]. To enhance relevance in the context of recruitment, the item-based collaborative filtering method is deemed more suitable, as it focuses on user behavior toward job postings.

In traditional relational databases, modeling and traversing relationships between entities such as users, job applications, and companies can be inefficient, especially in large-scale systems. Graph databases like Neo4j, with their node–relationship structures and optimized traversal mechanisms, offer an alternative approach better suited to recommendation systems that rely on identifying complex user–item interaction patterns[2] [3]. With the aid of the Cypher query language, Neo4j enables dynamic and efficient exploration of data relationships [4], making it ideal for recommendation tasks similar to those used by Facebook for friend suggestions or by Amazon for product recommendations[5].

To address the challenge of matching job seekers with relevant postings, this study develops a recommendation system using the Jaccard Coefficient within a graph database structure. From a dataset of 500 applicants, job postings, and applications, 237 users were selected for evaluation to ensure sufficient interaction data. While Euclidean Distance has been used in prior studies—such as a textbook translation system that achieved 86.72% accuracy [6]. it is less efficient for large-scale or sparse data due to higher computational complexity. In contrast, the Jaccard Coefficient is better suited for such data as it relies on simple set-based comparisons [7], making it ideal for modeling historical job applications in recruitment platforms.

This study hypothesizes that using Jaccard similarity on a graph-based model will yield high recall, especially for users with consistent application histories. The system applies item-based collaborative filtering, with similarity measured by the Jaccard Coefficient. Data is stored in Neo4j and queried using Cypher. Jaccard compares set intersection over union and has proven effective in handling imbalanced data. For instance, a study on Covid-19 hoax classification using Jaccard at k = 4 achieved 0.696 accuracy, 0.710 precision, and 0.599 F1-score[8].

A recent study by Siregar, Pratama, and Himawan (2024) further confirmed the effectiveness of the Jaccard method in

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v14i3.2387, Copyright ©2025
Submitted : Mey 20, 2025, Revised : June 17, 2025, Accepted : June 20, 2025, Published : July 28, 2025

recruitment contexts. By using applicant profiles and competencies as the basis for recommendation, the Jaccard-based model successfully predicted potential candidates with an accuracy of 75%, precision of 71%, recall of 62%, and an F1-score of 67%. These findings indicate that the Jaccard Coefficient is not only efficient but also accurate in filtering suitable applicants based on their profiles and historical interactions [9].

Based on this background, the present study proposes the development of a Job Recommendation System using the Jaccard Coefficient Method implemented on a Graph Database. The system is designed to maximize historical data from 500 applicants, 500 job postings, and 500 applications collected by PT Nirmala Satya Development during the 2020–2024 period. This system is expected to provide more accurate and efficient job recommendations, helping job seekers find positions that match their qualifications.

II. Literature Review

A. Previous research

This section reviews key studies on recommendation systems, graph databases, and Jaccard similarity.

Jennifer Florentina & Kurniawan (2023) compared relational vs graph databases in a movie recommendation system using Jaccard-based content filtering. Their Neo4j implementation performed far better in latency (6–7 s) and memory (~42 MB) compared to PostgreSQL (120–150 s, ~120 MB)[10].

Ristias et al. (2023) evaluated Cosine vs Jaccard similarity in a content-based recommendation within an Indonesian event–vendor platform. Cosine outperformed Jaccard (66% vs 49% accuracy), highlighting Jaccard's limitations in certain content-based tasks [11], particularly with attribute-rich item descriptions.

This study builds upon these insights by integrating Jaccard Coefficient with Neo4j for user–job recommendations using real application data. Unlike movie or content-based systems, our work focuses on user–item collaborative filtering in a recruitment setting, modeling data interactions through graph structure for efficient similarity computation. We also complement Jaccard-based recommendations with popularity-based fallbacks to handle cold-start users, offering a more robust solution in practical settings.

B. Recommendation System

A recommendation system personalizes content by analyzing user preferences and behaviors to suggest relevant items [12]. It uses data analysis to rank items likely to interest users [13]. This study applies item-based collaborative filtering using the Jaccard Coefficient to recommend similar job listings based on user application history.

C. Job Vacancy

A job vacancy provides company-published information about available positions, required qualifications, and application procedures. In the job-seeking process, efficiency is crucial. Recommendation systems can assist job seekers in finding more relevant job postings faster and more accurately.

D. Jaccard Method

The Jaccard Coefficient measures the similarity between two sets by dividing the number of shared elements (intersection) by the total number of unique elements (union) [14]. It is simple, effective for sparse data, and independent of element frequency. However, it does not account for the order or quantity of elements, making it less ideal for quantitative or large identical datasets. Here is the jaccard formula:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \qquad (1)$$

The sequence of the jaccard algorithm is as follows:

1. Start with two sets of data to compare.
2. Calculate the intersection by finding the elements that are in set A and in set B.
3. Calculate union by combining all elements from both sets, without duplication.
4. Calculate the Jaccard coefficient by dividing the total intersection by the total union.

E. Graph Database

A graph database stores data in the form of nodes, edges, and properties, using index-free adjacency to access related data directly. This structure is ideal for modeling complex inter-data relationships [15].

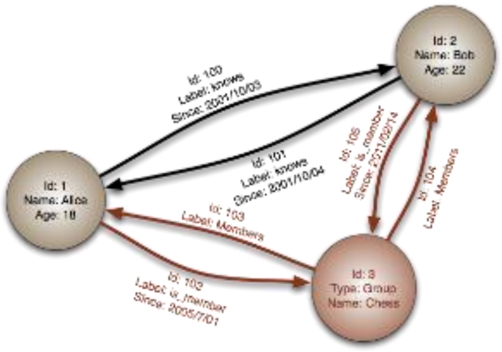


Fig. 1. Structure Database Graph

The overall structure of the graph is shown in Figure 1, where data is modeled as nodes (entities) and relationships (edges), enabling efficient traversal and representation of complex connections.

F. Neo4j

Neo4j is a widely used graph database system that supports the property graph model and Cypher query language. It efficiently stores and traverses connected data, making it suitable for recommendation systems, and is available in both open-source and enterprise editions [16].

G. Cypher Query Language

Cypher is a declarative query language designed for Neo4j, allowing expressive queries to retrieve data from a graph. With a visual syntax based on the property graph model, Cypher uses keywords such as MATCH, WHERE, and RETURN to describe patterns, and supports complex data types and graph structures.

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v14i3.2387, Copyright ©2025
Submitted : Mey 20, 2025, Revised : June 17, 2025, Accepted : June 20, 2025, Published : July 28, 2025

H. *Application Programming Interface (API)*

API (Application Programming Interface) is a mechanism that allows two or more systems, applications, or components to connect and communicate with each other using certain definitions or protocols [17]. An API enables communication between applications. REST API, used in this system, operates over HTTP with methods like GET, POST, PUT, and DELETE to facilitate data exchange. Responses are typically in JSON format, making REST efficient and developer-friendly.

III. RESEARCH METHODOLOGY

A. *Place and Time of Research*

This research was conducted at PT Nirmala Satya Development from August 2024 to September 2024. The data needed are applicant biodata, vacancy data and applicant application data which are used to find job vacancy recommendations.

B. *Research Flow*

The steps undertaken in this study follow the flow illustrated in Figure 2, which outlines the sequence from data collection to system implementation and evaluation.

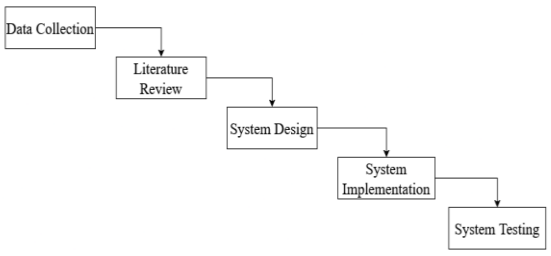


Fig. 2. Research Flow

- Data collection consists of primary data in the form of user data, job vacancies, and applications from the PT Nirmala Satya Development website, as well as secondary data obtained from literature such as books, journals, and the internet related to the Jaccard Coefficient method and graph data structures.
- Literature Review, Before conducting research, a literature review of related research on the Jaccard Coefficient method on graph databases was conducted.
- System design is carried out starting from designing the overall system architecture which produces a context diagram then a database design is carried out which produces a graph model and the last is an interface design that will produce a wireframe. The next stage is system implementation, the system design that has been made will be implemented in the coding stage using a programming language.
- System implementation is the process of coding applications using programming languages. The system implementation uses Javascript (NodeJS) as the programming language on the backend side, Cypher Query Language as a query language to the database and on the frontend side using the VueJS framework.
- System testing is carried out by performing the stages of testing and debugging the program to ensure that it runs well in accordance with the design that has been made before.

C. *Method*

This research method is carried out through user data testing at PT Nirmala Satya Development with several approaches, namely: a focused literature review to collect references from books and journals related to recommendation systems, Jaccard Coefficient, and graph databases; interviews with company parties; analysis of historical application data to understand user behavior.

D. *System scheme*

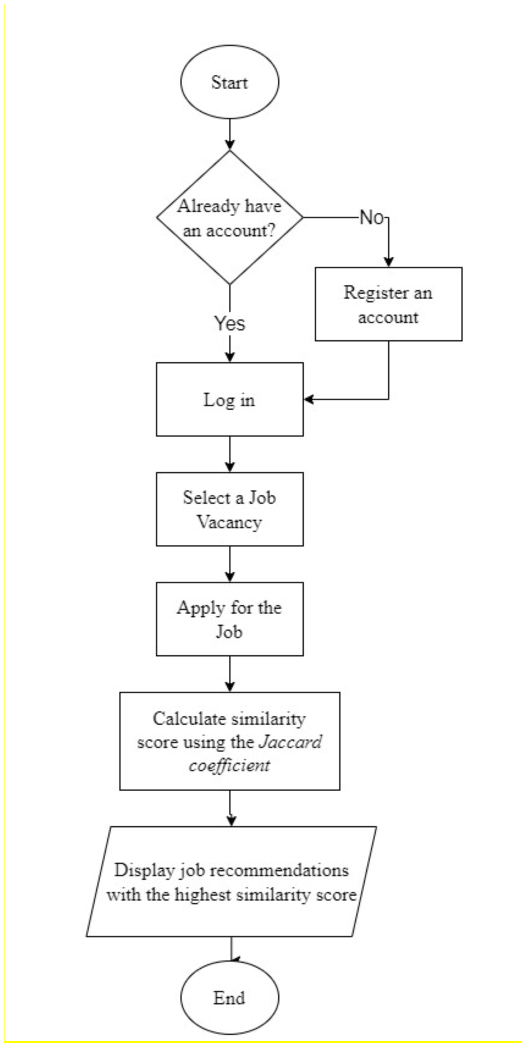

Start
Already have an account?
No
Register an account
Yes
Log in
Select a Job Vacancy
Apply for the Job
Calculate similarity score using the *Jaccard coefficient*
Display job recommendations with the highest similarity score
End

Fig. 3. System scheme

As illustrated in Figure 3, the system starts by checking whether the user already has an account; if not, the user is directed to register, if so, the user can log in. After logging in, the user selects and applies for a vacancy of interest. The system then calculates the similarity between the vacancy and other vacancies applied by other users using the Jaccard Coefficient, then displays vacancy recommendations with the highest similarity value.

E. Calculation Scheme of Jaccard Coefficient Method

The Jaccard similarity computation process is visualized in Figure 4, highlighting how application histories are compared between users to generate recommendations.

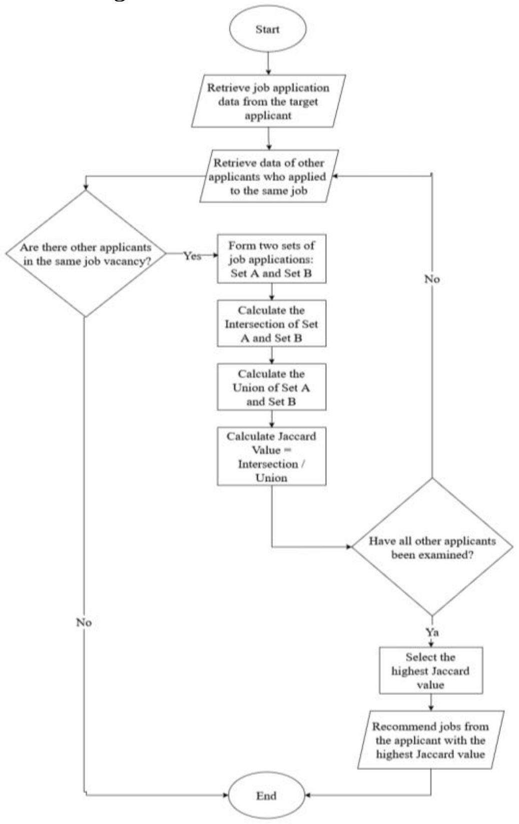

Start
Retrieve job application data from the target applicant
Retrieve data of other applicants who applied to the same job
Are there other applicants in the same job vacancy?
Yes
Form two sets of job applications: Set A and Set B
Calculate the Intersection of Set A and Set B
Calculate the Union of Set A and Set B
Calculate Jaccard Value = Intersection / Union
No
Have all other applicants been examined?
No
Ya
Select the highest Jaccard value
Recommend jobs from the applicant with the highest Jaccard value
End

Fig. 4. Diagram Calculation Scheme of Jaccard Coefficient Method

The process starts when a user applies for a job vacancy; the system then retrieves the user's application data and other applicants who applied for the same vacancy. If there are no other applicants, the process stops. If there are, the system forms two sets of applications (user and other applicants), then calculates the Jaccard value based on the intersection and union comparison of the vacancies applied for. This process is repeated until all relevant applicants are calculated, then the system selects the highest Jaccard value and displays recommendations for vacancies that the user has not applied for but have high similarity.

IV. Analysis And Discussion

A. System Overview

This recommendation system uses a graph database and the Jaccard Coefficient method to calculate the similarity of applications and present relevant job vacancies to users..

1) System Architecture

The flow of data communication in the system in general is as follows:

- The user accesses the application through the VueJS interface.
- Requests are sent by the frontend to the backend through the API.
- The server processes the request and passes the query to Neo4j.
- The results obtained from Neo4j are returned to the server, and then displayed back to the user through the interface.

The communication between system components is depicted in Figure 5, which outlines the interactions between frontend, backend, and the Neo4j graph database.

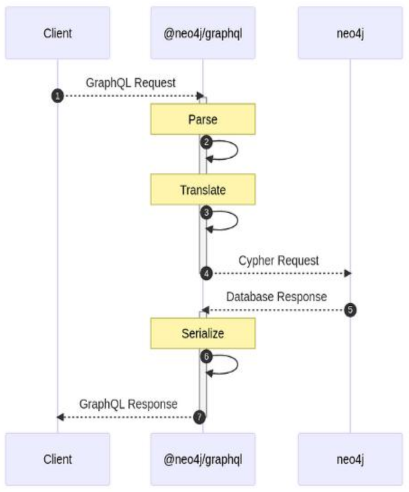
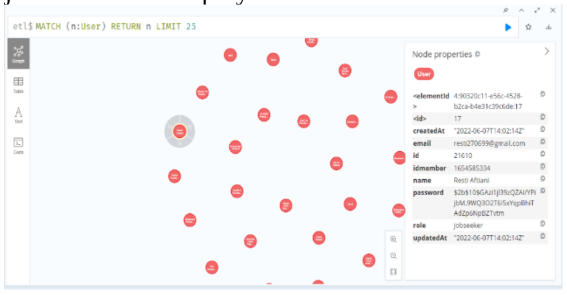


Fig. 5. System Architecture

2) Graph Database Structure

The database structure is built using a graph model, consisting of three main types of nodes, namely:

- User, Nodes that represent users of the system, both as jobseekers and employers.



```
etl$ MATCH (n:User) RETURN n LIMIT 25
```

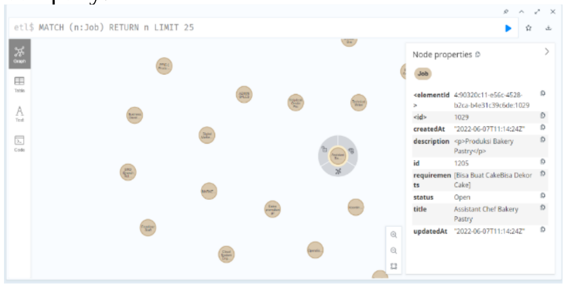
Node properties

User

<elementId>	4:90320c11-e56c-4528-b2ca-b4e31c39c6de:17
<id>	17
createdAt	"2022-06-07T14:02:14Z"
email	resti270699@gmail.com
id	21610
idmember	1654585334
name	Resti Aftiani
password	$2b$10$GAzI1jl39zQZAI/YPijbM.9WQ3O2T6I5xYqpBhiTAdZp6NpBZTvtm
role	jobseeker
updatedAt	"2022-06-07T14:02:14Z"

Fig. 6. Node User

- Job, Node that represents job vacancy data posted by the company.



```
etl$ MATCH (n:Job) RETURN n LIMIT 25
```

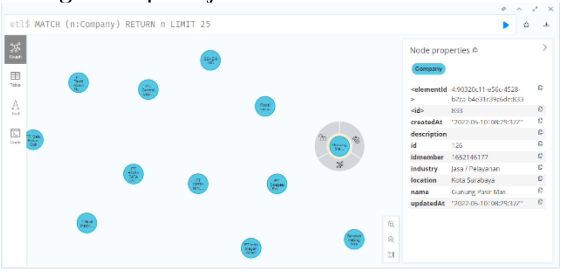
Node properties

Job

<elementId>	4:90320c11-e56c-4528-b2ca-b4e31c39c6de:1029
<id>	1029
createdAt	"2022-06-07T11:14:24Z"
description	<p>Produksi Bakery Pastry</p>
id	1205
requirements	[Bisa Buat CakeBisa Dekor Cake]
status	Open
title	Assistant Chef Bakery Pastry
updatedAt	"2022-06-07T11:14:24Z"

Fig. 7. Node Job

- Company, Company nodes represent company data that manages and posts job vacancies.



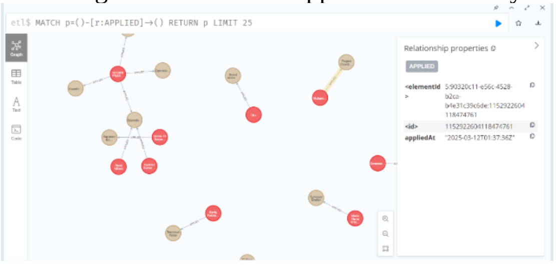
```
etl$ MATCH (n:Company) RETURN n LIMIT 25
```

```
Node properties
Company
<elementId>  4:90320c11-e56c-4528-b2ca-b4e31c39c6de:833
<id>         833
createdAt    "2022-06-10T08:29:37Z"
description
id           126
idmember     1652146177
industry     Jasa / Pelayanan
location     Kota Surabaya
name         Gunung Pasir Mas
updatedAt    "2022-06-10T08:29:37Z"
```

Fig. 8. Company Node

The relationship between nodes consists of:

- APPLIED, A relation from the User node to the Job node, indicating that the user has applied for a vacancy.

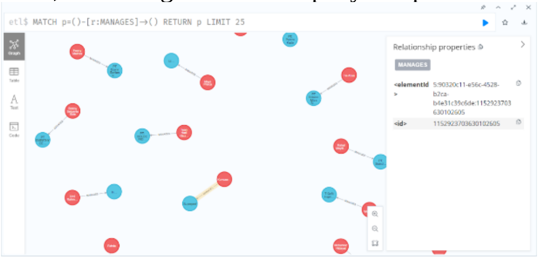


```
etl$ MATCH p=()-[r:APPLIED]→() RETURN p LIMIT 25
```

```
Relationship properties
APPLIED
<elementId>  5:90320c11-e56c-4528-b2ca-b4e31c39c6de:1152922604118474761
<id>         1152922604118474761
appliedAt    "2025-03-12T01:37:36Z"
```

Fig. 9. Applied Relationship

- POSTED, A relation from the Company node to the Job node, indicating that the company has posted the vacancy.

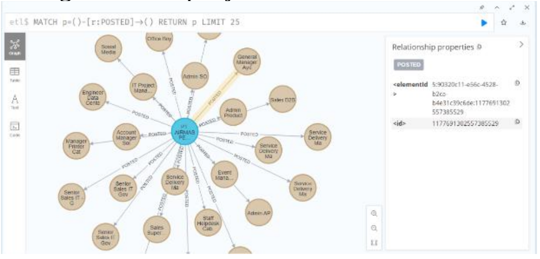


```
etl$ MATCH p=()-[r:MANAGES]→() RETURN p LIMIT 25
```

```
Relationship properties
MANAGES
<elementId>  5:90320c11-e56c-4528-b2ca-b4e31c39c6de:1152923703630102605
<id>         1152923703630102605
```

Fig. 10. Posted Relationship

- MANAGES, A relation from the User node (with the role employer) to the Company node, indicating that the user manages the company.



```
etl$ MATCH p=()-[r:POSTED]→() RETURN p LIMIT 25
```

```
Relationship properties
POSTED
<elementId>  5:90320c11-e56c-4528-b2ca-b4e31c39c6de:1177691302557385529
<id>         1177691302557385529
```

Fig. 11. Manages Relationship

3) *Recommendation System Workflow*

- The user logs in to the system and selects the job vacancy of interest.
- The user applies for one of the vacancies.
- The system then searches for other users who have also applied for the same vacancy.
- From those users, the system collects their entire application history.
- Next, the system calculates the level of similarity between the current user and other users using the Jaccard Coefficient method.
- After the similarity value is obtained, the system selects vacancies that are often applied for by users with a high degree of similarity, but have never been applied for by the current user
- The vacancies are then organized and displayed as recommendations to the user.
- With this approach, the system is able to provide recommendations that are more personalized and in accordance with the interests of users based on the behavior patterns of other users who have similar preferences.

4) *Concept of Jaccard Calculation in the System*

The Jaccard Coefficient method is used to measure the similarity between two sets, namely vacancies applied for by users (Set A) and other users who also applied for the same vacancy (Set B). The system calculates the Jaccard value to recommend other vacancies that have been applied for by similar users, based on the similarity of applications after the user has applied for Job X.

TABLE I. USER DATA

User	Job Applied for
1	Job X
2	Job X, Job Y, Job Z
3	Job X, Job Y, Job W
4	Job X, Job Y, Job Z, Job W

Jaccard Calculation Steps:

Set A = {Job X}

Calculate the Jaccard value between A and every other user (who also applied for Job X)

a. User 2
   - Set B = {Job X, Job Y, Job Z}
   - Intersection = {Job X} → 1
   - Union = {Job X, Job Y, Job Z} → 3
   - Jaccard(A, B) = 1 / 3 = 0.33

b. User 3
   - Set C = {Job X, Job Y, Job W}
   - Intersection = {Job X} → 1
   - Union = {Job X, Job Y, Job W} → 3
   - Jaccard(A, C) = 1 / 3 = 0.33

c. User 4
   - Set D = {Job X, Job Y, Job Z, Job W}
   - Intersection = {Job X} → 1
   - Union = {Job X, Job Y, Job Z, Job W} → 4
   - Jaccard(A, D) = 1 / 4 = 0.25

TABLE II. JACCARD SCORE RESULT

User	Jaccard Score
User 2	0.33
User 3	0.33
User 4	0.25

Combine all vacancies applied for by other users (except Job X), then sort by highest score:

- From User 2: Job Y, Job Z
- From User 3: Job Y, Job W

- From User 4: Job Y, Job Z, Job W

Calculate the frequency of occurrence:

TABLE III. RECOMMENDATION RESULTS

Job	Frequency
Job Y	3
Job Z	2
Job W	2

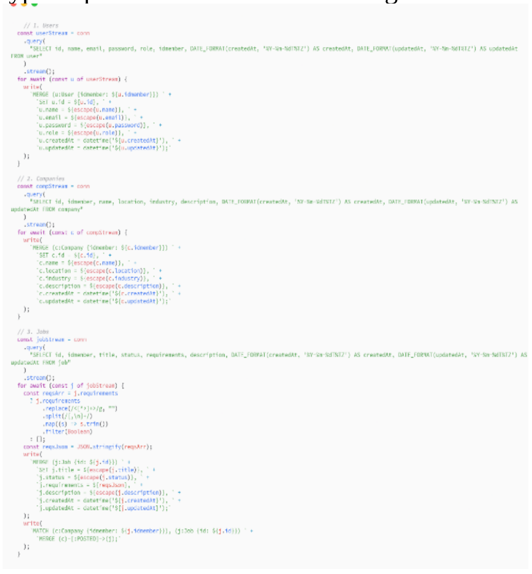
B. *System Implementation*

1. *Data Migration to Neo4j Graph Database*

Data from PT NSD's MySQL (500 vacancies, companies, and applicants) is migrated to Neo4j to build a graph-based recommendation system.

- Extracting data from MySQL.
- Convert the data into Cypher Query format.
- Running the query in Neo4j to form nodes and relations.

The process of migrating tabular data into graph format using Cypher queries is demonstrated in Figure 12.



```
  // 1. Users
  const userStream = conn
    .query(
      "SELECT id, name, email, password, role, idmember, DATE_FORMAT(createdAt, '%Y-%m-%dT%TZ') AS createdAt, DATE_FORMAT(updatedAt, '%Y-%m-%dT%TZ') AS updatedAt
FROM user"
    )
    .stream();
  for await (const u of userStream) {
    write(
      `MERGE (u:User {idmember: ${u.idmember}}) ` +
        `SET u.id = ${u.id}, ` +
        `u.name = ${escape(u.name)}, ` +
        `u.email = ${escape(u.email)}, ` +
        `u.password = ${escape(u.password)}, ` +
        `u.role = ${escape(u.role)}, ` +
        `u.createdAt = datetime('${u.createdAt}'), ` +
        `u.updatedAt = datetime('${u.updatedAt}');`
    );
  }

  // 2. Companies
  const compStream = conn
    .query(
      "SELECT id, idmember, name, location, industry, description, DATE_FORMAT(createdAt, '%Y-%m-%dT%TZ') AS createdAt, DATE_FORMAT(updatedAt, '%Y-%m-%dT%TZ') AS
updatedAt FROM company"
    )
    .stream();
  for await (const c of compStream) {
    write(
      `MERGE (c:Company {idmember: ${c.idmember}}) ` +
        `SET c.id = ${c.id}, ` +
        `c.name = ${escape(c.name)}, ` +
        `c.location = ${escape(c.location)}, ` +
        `c.industry = ${escape(c.industry)}, ` +
        `c.description = ${escape(c.description)}, ` +
        `c.createdAt = datetime('${c.createdAt}'), ` +
        `c.updatedAt = datetime('${c.updatedAt}');`
    );
  }

  // 3. Jobs
  const jobStream = conn
    .query(
      "SELECT id, idmember, title, status, requirements, description, DATE_FORMAT(createdAt, '%Y-%m-%dT%TZ') AS createdAt, DATE_FORMAT(updatedAt, '%Y-%m-%dT%TZ') AS
updatedAt FROM job"
    )
    .stream();
  for await (const j of jobStream) {
    const reqsArr = j.requirements
      ? j.requirements
          .replace(/<[^>]+>/g, "")
          .split(/[,\n]+/)
          .map((s) => s.trim())
          .filter(Boolean)
      : [];
    const reqsJson = JSON.stringify(reqsArr);
    write(
      `MERGE (j:Job {id: ${j.id}}) ` +
        `SET j.title = ${escape(j.title)}, ` +
        `j.status = ${escape(j.status)}, ` +
        `j.requirements = ${reqsJson}, ` +
        `j.description = ${escape(j.description)}, ` +
        `j.createdAt = datetime('${j.createdAt}'), ` +
        `j.updatedAt = datetime('${j.updatedAt}');`
    );
    write(
      `MATCH (c:Company {idmember: ${j.idmember}}), (j:Job {id: ${j.id}}) ` +
        `MERGE (c)-[:POSTED]->(j);`
    );
  }
```

Fig. 12. Table Migration in Cypher

2. *User Interface*

The system interface is designed with responsive and intuitive VueJS, supporting two user roles: jobseeker and employer. Users can register, login and access pages according to their respective roles. The vacancy list is available to all users, but after login, the system adjusts the display: users without application history will see popular vacancies, while those who have applied will get recommendations based on application similarity through Jaccard method and Neo4j graph data. The vacancy details page displays complete information along with an “Apply Now” option for logged-in jobseekers. Employers have additional access to manage company profiles and post vacancies, while all users can view company details and a list of available vacancies.

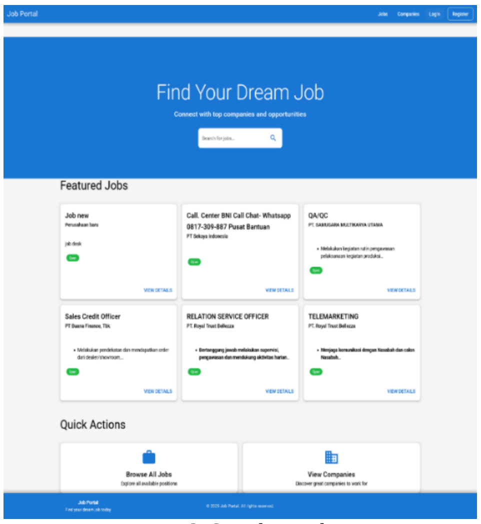


Fig. 13. Sample Display

3. *Implementation of Neo4j and Recommendation Algorithm*

The backend system is built with Node.js and uses Neo4j as a graph database to represent relationships between entities such as users, vacancies, and companies. Data communication is done through API, and queries are written with Cypher.

The graph structure consists of User, Job, and Company nodes, with relations such as [:APPLIED], [:POSTED], and [:MANAGES].

The recommendation algorithm is divided into two:

- New users: are recommended the most popular vacancies based on the number of applicants.
- Existing users: using Jaccard Similarity to recommend vacancies applied for by other users with similar preferences.

C. *System Testing*

1. *Functional Testing*

TABLE IV. TEST THE SYSTEM WITH BLACK-BOX TESTING

No	Feature Tested	Test Scenario	Expected Result	Status
1	Registration	User fills out the form and clicks "Register"	Account is successfully created and redirected to login page	Passed
2	Login	User enters valid email and password	User is redirected to the main page	Passed
3	Apply for a job	Jobseeker clicks "Apply Now" on the job detail page	[:APPLIED] relationship is created in Neo4j	Passed
4	Display popular jobs	New jobseeker accesses the recommendation page	System displays 10 most-applied jobs	Passed

No	Feature Tested	Test Scenario	Expected Result	Status
5	Display recommendati ons (Jaccard)	Returning jobseeker accesses the recommendation page	System displays 10 relevant jobs based on similarity	Passed
6	Post a job (Employer)	Employer fills out the job form and clicks "Post Job"	New job appears in the list and is linked to the company	Passed
7	Manage company (Employer)	Employer opens the company management page	Can view and edit company profile	Passed

2. System Evaluation

Evaluation of the recommendation system was carried out on a dataset of 237 users, each with varying levels of job application history. The evaluation employed standard metrics: precision, recall, and F1-score. Results showed that approximately 80% of users received at least one relevant recommendation, indicating the system's ability to capture relevant matches for the majority of users. The average precision ranged from 10% to 20%, which is reasonable considering that the system generates 10 recommendations per user. Meanwhile, recall was consistently high, often reaching 1.0, particularly for users with a single or clearly defined job target. The F1-score, which reflects the balance between precision and recall, peaked at 0.33, indicating strong performance in several cases. On the other hand, F1-scores of 0 occurred in users with sparse historical data or unique application patterns, where the system lacked sufficient information to compute meaningful similarities.

D. Discussion

The evaluation results indicate that the developed recommendation system is generally effective in delivering relevant job suggestions. Most of the 237 tested users received at least one appropriate recommendation from the ten provided, with recall values often reaching 1.0, especially among users who had applied to a small number of jobs. This shows the system's ability to accurately retrieve relevant items. However, the average precision ranged between 0.10 and 0.20, which, although significantly higher than the 1.8% obtained through a random recommendation baseline, still means that several of the recommended jobs may be irrelevant to the user. This level of precision may affect user satisfaction and suggests that future improvements are needed to enhance recommendation quality.

A small portion of users experienced an F1-score of 0. These cases occurred primarily due to either a lack of overlapping job application history—where users applied to jobs that no one else had—or due to extremely sparse interaction data, such as applying to only a single job. In such situations, the system was unable to calculate similarity scores, resulting in no relevant recommendations. To address this, future versions of the system could integrate content-based profiling or demographic matching to provide better support for cold-start users.

Overall, the system performed reliably and consistently for users with moderate to rich application histories. The Jaccard Coefficient proved effective for measuring preference similarity among users, while the use of Neo4j as a graph database allowed for efficient modeling and traversal of user-job relationships. To further improve system accuracy and user satisfaction, hybrid filtering techniques, enhanced user profiling, and result ranking strategies could be implemented. With these improvements, the system has strong potential to support behavior-aware job recommendation on a broader scale.

V. Conclusion

The results of this study show that the developed job recommendation system is capable of providing relevant suggestions, particularly for users with active application histories. High recall (often reaching 1.0) was achieved in most cases, and while average precision ranged from 10% to 20%, it was significantly better than a random baseline. The use of the Jaccard Coefficient successfully captured similarities in user behavior, and the integration with Neo4j facilitated efficient relationship modeling between users and jobs.

From a practical perspective, this system can be adopted by small- to medium-scale recruitment platforms, especially those with limited datasets, to enhance their matching processes without requiring complex machine learning infrastructure. Its lightweight, graph-based design is suitable for environments where computing resources are constrained.

Future work includes testing the system's scalability on larger datasets, as well as improving recommendation accuracy by enhancing user profiling, for example through CV parsing or behavioral tracking. Incorporating hybrid filtering methods and personalized ranking mechanisms also holds promise for increasing precision and user satisfaction.

References

[1] A. A. P. Devi and D. B. Tonara, "Rancang Bangun Recommender System dengan Menggunakan Metode Collaborative Filtering untuk Studi Kasus Tempat Kuliner di Surabaya," *Jurnal Informatika Dan Sistem Informasi*, pp. 102–110, 2015.

[2] F. A. Ajipradana, "Sistem Rekomendasi Film Menggunakan Algoritma Itesm-Based Collaborative Filtering Dan Basis Data Graph," *Jurnal Univsitas Diponegoro*, 2017.

[3] Y. Liang, "Research on Personalized Recommendation System for Graph Database," in *2018 International Conference on Network, Communication, Computer Engineering (NCCE 2018)*, Atlantis Press, 2018, pp. 1019–1023.

[4] D. Fernandes and J. Bernardino, "Graph Databases Comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB.," *Proceedings of the 7th International Conference on Data Science*, vol. 18, pp. 373–380, 2018.

[5] A. Ilic and M. Kabiljo, "Recommending items to more than a billion people." Accessed: Oct. 13, 2024. [Online]. Available: https://engineering.fb.com/2015/06/02/core-infra/recommending-items-to-more-than-a-billion-people/

[6] W. Fuadi, M. Maryana, and U. Zahara, "Sistem penerjemahan kitab pelajaran akhlak ke dalam bahasa Indonesia menggunakan metode Euclidean Distance," *TECHSI-Jurnal Teknik Informatika*, vol. 11, no. 1, pp. 92–103, 2019.

[7] S. Pawestri and Y. Suyanto, "Analisis Perbandingan Metode Similarity untuk Kemiripan Dokumen Bahasa Indonesia pada Deteksi Kemiripan Teks Bahasa Indonesia," *JURNAL MEDIA INFORMATIKA BUDIDARMA*, vol. 3, no. 8, pp. 1440–1450, 2024.

[8] W. Hidayat, E. Utami, A. F. Iskandar, A. D. Hartanto, and A. B. Prasetio, "Perbandingan Performansi Model pada Algoritma K-NN terhadap

Klasifikasi Berita Fakta Hoaks Tentang Covid-19," *Edumatic: Jurnal Pendidikan Informatika*, vol. 5, no. 2, pp. 167–176, 2021.

[9] I. M. Siregar, D. Pratama, and C. Himawan, "Penggunaan Jaccard Similarity Coefficient dalam Optimasi Proses Rekrutmen Karyawan Berbasis Profil dan Kompetensi," *SINTECH (Science and Information Technology) Journal*, vol. 7, no. 2, pp. 101–111, 2024.

[10] J. Florentina and H. C. Kurniawan, "Perbandingan Penerapan Relational Database Dan Graph Database Dalam Sistem Rekomendasi Film," *Jurnal Telematika*, vol. 18, no. 2, pp. 94–103, 2023.

[11] A. A. Ristias, E. D. Wahyuni, and S. F. A. Wati, "Komparasi Kinerja Metode Cosine dan Jaccard Similarity dalam Content-Based Recommendation Systems (CBRS) pada Aplikasi Eventhings," *Jurnal Informatika dan Teknik Elektro Terapan*, vol. 12, no. 3, 2024.

[12] L. Sebastia, I. Garcia, E. Onaindia, and C. Guzman, "e-Tourism: a tourist recommendation and planning application," *International Journal on Artificial Intelligence Tools*, vol. 18, no. 05, pp. 717–738, 2009.

[13] 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.

[14] S. Sunardi, A. Yudhana, and I. A. Mukaromah, "Implementasi Deteksi Plagiarisme Menggunakan Metode N-Gram Dan Jaccard Similarity Terhadap Algoritma Winnowing," *Transmisi: Jurnal Ilmiah Teknik Elektro*, vol. 20, no. 3, pp. 105–110, 2018.

[15] Wikipedia, "Graph database," wikipedia.org. Accessed: May 15, 2025. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/3/3a/GraphDatabase_PropertyGraph.png.

[16] N. Francis *et al.*, "Cypher: An evolving query language for property graphs," in *Proceedings of the 2018 international conference on management of data*, 2018, pp. 1433–1445.

[17] C. L. Rujiani, E. R. Syahputra, and S. D. Andriana, "Implementation Of Application Programming Interface (API) Using Representational State Transfer (REST) Architecture For Development E-Learning Unhar Medan," *INTERNATIONAL JOURNAL OF DATA SCIENCE AND VISUALIZATION (IJDSV)*, vol. 1, no. 1, 2023.

p-ISSN 2301-7988, e-ISSN 2581-0588
DOI : 10.32736/sisfokom.v14i3.2387, Copyright ©2025
Submitted : Mey 20, 2025, Revised : June 17, 2025, Accepted : June 20, 2025, Published : July 28, 2025