**COMP4521 Mobile Application Development**

**Final Report**

**Project topic:** Job-Seeking Application with Similarity Learning in Natural Language Processing

**Group name:** Group A – JMatch

**Group members:**

SZE Kai Tik (20496011)

LIANG Houdong (20676855)

PENG Yiqing (20676934)

Word count: **3143**

Date: **15/05/2023**

Table of Contents

[1 Introduction 3](#_Toc134917183)

[2 Methodology 4](#_Toc134917184)

[2.1 Design 4](#_Toc134917185)

[2.1.1 The Overall System Architecture 4](#_Toc134917186)

[2.1.2 System App Flow and User Journey Design 5](#_Toc134917187)

[2.1.3 User Interface Design 6](#_Toc134917188)

[2.1.4 Database Design 7](#_Toc134917189)

[2.2 Implementation 8](#_Toc134917190)

[2.2.1 Android Application Implementation 8](#_Toc134917191)

[2.2.2 Data Storage and Backend Service 12](#_Toc134917192)

[2.2.3 Natural Language Processing Algorithm 14](#_Toc134917193)

[2.2.4 Backend APIs 15](#_Toc134917194)

[2.3 Testing 18](#_Toc134917195)

[3 Discussion 20](#_Toc134917196)

[3.1 Challenges in NLP Algorithm Design 20](#_Toc134917197)

[3.2 Challenges in Android App Development 20](#_Toc134917198)

[3.3 Limitation 21](#_Toc134917199)

[4 Conclusions 23](#_Toc134917200)

[5 Reference 24](#_Toc134917201)

[6 Appendix A: Testing Documents 25](#_Toc134917202)

1 Introduction

The rapid advancement of human resources (HR) technologies has led to the integration of machine learning techniques into recruiting sourcing systems, which has significantly reduced errors and saved time for recruiters and job seekers [1]. However, the digitalization of the recruitment process has brought challenges to talent management as the recruiting process needs to happen in shorter time frames.

To address this challenge, we have developed a job-seeking mobile application for candidates and recruiters that uses Natural Language Processing (NLP) to match job descriptions with cover letters. The application also offers a folder structure to store the matching history and quantitative analysis for job-seeking history management. Our project aims to create an ecosystem for recruiters and applicants to streamline the recruitment process in a competitive job market.

This report will demonstrate the application implementation of this job-seeking application project from the backend development, frontend Andriod features, and database development, to the algorithms we designed. The results and challenges will also be discussed in this report.

1. Methodology
   1. Design
      1. The Overall System Architecture

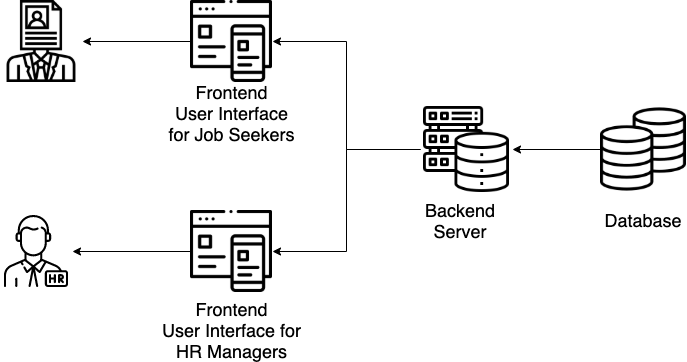


Figure 1 – System Architecture

Our project built a job-seeking mobile application for both candidates and recruiters. The overall system requires a backend server with a designed NLP algorithm for resume-JD matching. With the support of a database, the backend server can learn the information provided by the user and give matching feedback to the two UIs through APIs. One UI is designed for HR managers and the other for job seekers to achieve resume analysis.

* + 1. System App Flow and User Journey Design

一張含有 圖表 的圖片

自動產生的描述

Figure 2 – System App Flow

As shown in *Figure 2*, our project aims to build a mobile app to conduct the use cases demonstrated above. After the welcome page, the user will be guided to a login page. If the user does not have an account yet, they can click the Sign-Up button, which will link to the sign-up page for registration. Otherwise, they can directly type in their email and password to log in to their account and access the upload page. As candidates, they can upload one cover letter for matching, while an enterprise account allows the user to upload one job description at a time. After clicking the Get Matching Result button, the user can view the matching analysis calculated by our NLP algorithm, which will show the final score plus a keyword contribution chart for the candidate and a ranking table for the company. The data for matching and resume analysis will be retrieved from the database.

For the users’ convenience to manage the account, our app aims to build a user center as well to store the matching history. The users could view the previous matching records as a table in their user center. Clicking a specific matching record will redirect the user to that matching history. If the user would like to log out of the account, he can click the button Log Out to go back to the Login Page.

* + 1. User Interface *Design*

The prototype of the JMatch mobile app is initially designed by Figma [2]. For the design details, feel free to refer to *Figure 3*.

一張含有 螢幕擷取畫面, 文字, 設計 的圖片

自動產生的描述 一張含有 螢幕擷取畫面, 文字 的圖片

自動產生的描述

Figure 3 – User Interface Design

* + 1. Database Design

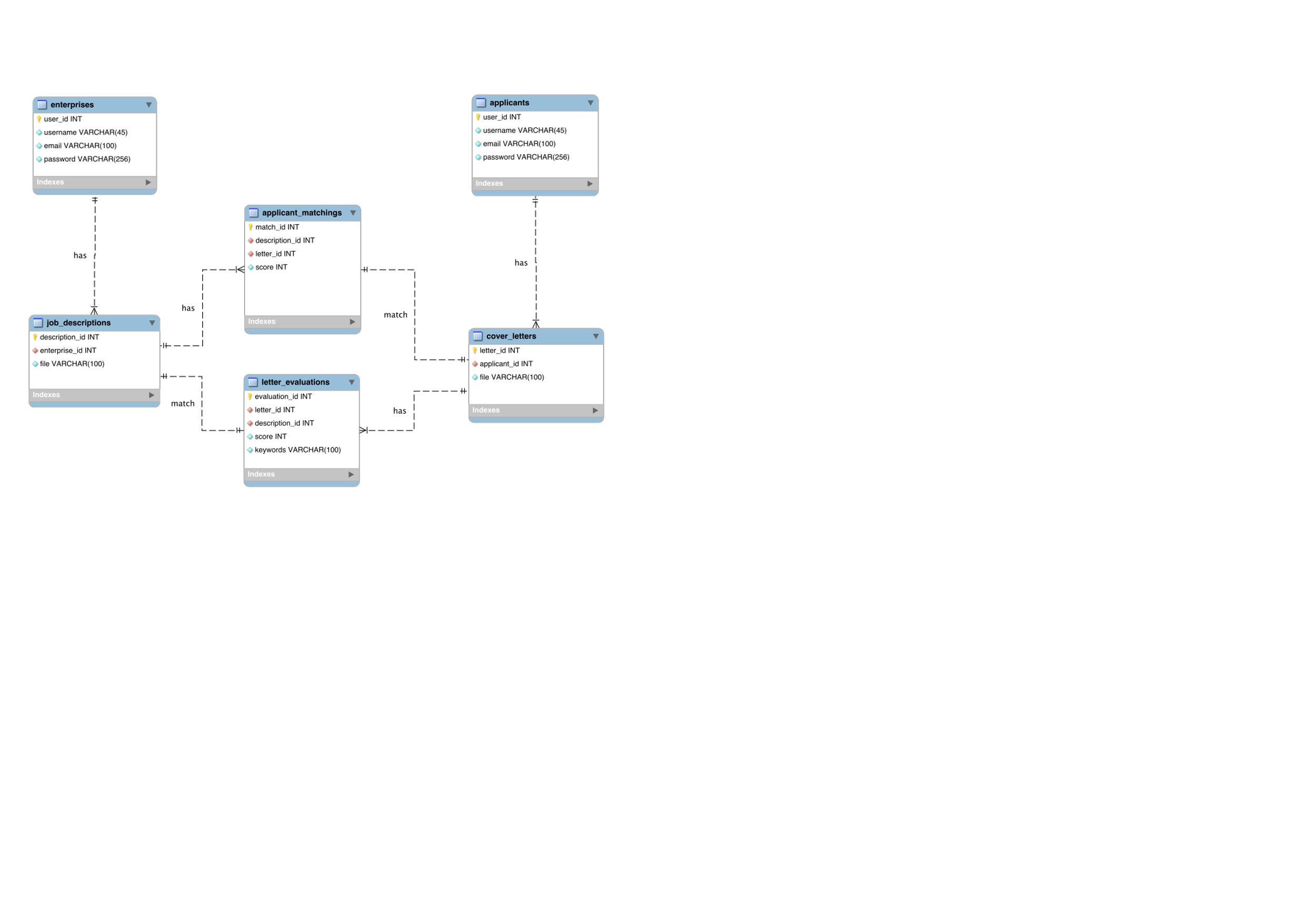


Figure 4 – Entity Relational Diagram

To be more specific, the data model includes the following data attributes for analysis:

1. User information: user\_id, username, email, (encrypted) password.
2. Document information: document\_id, cover letter/job description content, submission time, username.
3. Matching record data (Not completely stored in the database): title of the job description/cover letter, extracted keywords, matching scores with other documents
   1. Implementation
      1. Android Application Implementation

There are three main pages of our mobile app for both the enterprise side and the candidate side: the login page, the cover letters-job descriptions (CL-JDs) matching page, and the user center page.

A picture containing text, screenshot, font, graphic design

Description automatically generatedA screenshot of a login form

Description automatically generated with medium confidenceA screenshot of a login form

Description automatically generated with medium confidence

Figure 5 – Start, Login, and Signup

The first activity displayed on the screen is the start activity. Users can click the screen to use the application. On the login page, users can choose to either click the Log In button to log in with their existing account or click the Sign-Up button to create a new account with their email address. Click the Candidate button or the Enterprise button to choose whether they would like to log in or create a candidate account or an enterprise account. These two buttons are mutually restricted, which could not allow the user to choose both identities in registration.

A picture containing text, font, screenshot

Description automatically generated

Figure 6 – Verification Email Notification

After filling in the email address, new users are required to submit a verification code that is sent to the email they provided. There is an OnClick() activity linked to the Sign-Up button which will lead the users back to the login page after successful registration.

A screenshot of a login form

Description automatically generated with medium confidenceA screenshot of a test

Description automatically generated with medium confidence

Figure 7 – Login and Main Page

The main function for the login page is the login function. By choosing the account type, the user could type their registered email and password, and then there is an Intent activity linked to the OnClick() activity of the Log In button to lead the user to the main page. Users can click the “Start Matching” button to start a new match. Matching results are stored in the database and can be accessed again by clicking the corresponding record.

A screenshot of a phone

Description automatically generated with low confidenceA screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a document

Description automatically generated with low confidence

Figure 8 – CL-JDs Matching Page

The upload page allows the users to upload the cover letter and the job description by uploading files or pasting them on PlainText. If the user accidentally does both operations, only the uploaded file will be considered. The OnClick() activity of the Upload File button would allow the user to browse the local storage or their google drive (if there is a google drive account loaded) to find the PDF file of their documents for uploading. The difference between the candidate upload page and the enterprise upload page is the enterprise upload page allows the user to upload multiple cover letters for one job description. By clicking the “Get Matching Result” button, there is an Intent connected to its OnClick() activity to run the NLP algorithm and output the matching analysis results for the user.

A picture containing text, screenshot, software, operating system

Description automatically generatedA screenshot of a phone

Description automatically generated with low confidence

Figure 9 – CL-JDs Matching Result Page

After matching, the results are demonstrated to the user. For the candidate, the first page shows the keyword distribution bar chart in the uploaded PDF file or text. One thing that is worth mentioning is that the bar chart is implemented without using any packages or outside resources. It is a linear layout with many adaptations. Each bar in the bar chart is a row in the linear layout with different background colors and text. The second page displays the matching scores of the top 10 companies in a table. For the enterprise user, the only difference is that the list of companies on the second page is replaced with the list of candidates. Users can click the buttons at the bottom to switch between different pages. If the user wants to go back, the  button can lead him to the previous main page (their user center).

A screenshot of a phone

Description automatically generated with medium confidence

Figure 10 – Main Page After Matching

The updated main page will show the most current record on top of the records list with the title defined by the user when starting the match and the time when the match happened. Should the user log out of the account, clicking the Log Out button  could lead the user back to the login page.

* + 1. Data Storage and Backend Service

Most of the data required for our application is stored in the Weixin Cloud Base, a cloud database powered by Tencent. This database service is known for its lightweight efficiency, stability, and accessibility, and it fully implements the ACID properties of atomicity, consistency, isolation, and durability. The Weixin Cloud Base provides three major functionalities - cloud database, cloud storage, and cloud function - two of which have been integrated into our application. Textual information and properties of candidates (jobseekers), enterprises, cover letters, and job descriptions are stored as records in the database. A cloud function using *nodemailer* in JavaScript has been embedded into the application to enable the transmission of account verification emails. The services provided by the Weixin Cloud Base greatly support the functionalities of our application.

There are four tables created in the database, candidates, enterprises, cover letters, and job descriptions. The relational schema of the four tables is described below.

candidates (\_id, username, email, password)

primary key(username)

enterprises (\_id, username, email, password)

primary key(username)

cvs (\_id, document\_id, username, content, time)

primary key(document\_id)

foreign key(username) references candidate(username)

jds (\_id, document\_id, username, content, time)

primary key(document\_id)

foreign key(username) references enterprises(username)

The \_id attribute of users is a unique identifier generated internally. It could be treated as the primary key in any case. However, in our implementation, we simply use the username and our self-generated document\_id as the primary key. With the tables defined above, each candidate, enterprise, cover letter, and job description are uniquely identified. Some information needs to be shared across the entire session and is mostly user-specific, such as whether the user is logged in, whether the user is a candidate or an enterprise, and their username. This information does not change frequently once initialized and should be updated per session. It can be collected and accessed globally. The Android UI thread will throw an exception if blocked for more than 16ms. As a network-intensive application, it is necessary to perform all network actions in alternative threads. However, this can lead to challenges when data required in the main thread may not be prepared or retrieved in the background thread. It is necessary to prevent this situation.

* + 1. Natural Language Processing Algorithm

The first step for natural language processing is data processing. The input data is initially converted to lowercase, while the dummy strings are removed as follows.

With two sets of cover letter strings and job description strings, the relevance score is calculated based on the cosine similarity, measuring the similarity of angles between two vectors of an inner product space [3]. By doing so, the score could be used as a reference to the matching level between cover letters and job descriptions for comparison.

The second part of the algorithm is to extract keywords in the job description as a reference to the candidate for self-improvement. In this part, Term Frequency-Inverse Document Frequency (TF-IDF) Learning is applied, which is a traditional NLP approach in matching systems [4]. Since the importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus, we applied the TF-IDF approach and shows the top 10 keywords in the job descriptions in a bar chart format for the user.

* + 1. Backend APIs

First, to use the WeChat Cloud Service, we need to retrieve an access token. The AccessToken class is designed for this use. It issues a GET request to the cloud server to retrieve an access token that must be issued with any subsequent request to the cloud server. It is called at the beginning of StartActivity and runs every 2 hours because the access token is valid for two hours upon being requested according to the rule of WeChat Cloud Service.

A screenshot of a computer

Description automatically generated with low confidence

Figure 11 – WeChat Cloud Service Access Token API

To insert a record or query record in the WeChat Cloud Database, WeChat Cloud Service provides some useful APIs already. The details are shown below:

A screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidence

Figure 12 – WeChat Cloud Service Insert and Query API

One thing that deserves to be noticed is that the email verification function needs the help of the WeChat Cloud Function, which is much more complex than the previous database operations. To complete this feature, we implemented our own WeChat Cloud Function in the WeChat Small Application, using JavaScript (More precisely, NodeJS). And then in our Android application, we use HTTP POST requests to invoke the cloud function and fire the email verification. The following shows the details of the implementation of the cloud function and the API to invoke the cloud function:

A screen shot of a computer program

Description automatically generated with low confidenceA screen shot of a computer program

Description automatically generated with low confidence

Figure 13 – WeChat Cloud Function Implemented Using NodeJS

A screenshot of a computer

Description automatically generated with low confidence

Figure 14 – WeChat Cloud Service Invoke Cloud Function API

* 1. Testing

After implementing our system, we conducted testing on different parts of the system, ensuring the correctness of the program running. The testing was divided into three main parts: the backend testing, which ensured the backend server can successfully send verification emails, and store the user information such as name, email, and encrypted password, etc; the user interface testing, which consists of the enterprise UI and the candidate UI; the system testing that ensures the application is efficient, stable, and compatible.

To test the validity of our model, we proposed quantitative evaluations. As for quantitative evaluation, we experimented with randomly feeding a cover letter and a JD to the corresponding algorithm. This test can evaluate whether our matching method is valid and reasonable.

For the frontend testing, we test both the enterprise UI and candidate UI under the consideration that it provides excellent iteration speed combined with powerful features like mocking modules and timers, letting users control how the code executes. We conducted several kinds of testing, including Unit Testing, Visual Regression Testing, Accessibility Testing, Performance Testing, End-to-End Testing, Integration Testing, and Cross-Browser Compatibility Testing. These tests ensured the correctness of Android application switching, the display, and functionality of UI components including drop-box, checkbox, and buttons, and data transmission between pages. By testing, we ensure that the API executes correctly.

To test our app, we run the project and test every function and ensure that the jump from one activity to another takes no longer than 5 seconds and does not crash while using. The codes were written with no version dependency and various versions of Android virtual machines were also tested to ensure compatibility.

3 Discussion

* 1. Challenges in NLP Algorithm Design

The biggest challenges in the NLP algorithm design are to find an approach that is easily applied and compatible with the Java system since most of the existing artificial intelligence algorithms are built on Python and its powerful libraries. Therefore, this project hopes to design an algorithm that could be built in the Java system aiming to use fewer external libraries to avoid any possible technical issues. To achieve this, for some algorithms, we could not use the existing API directly but needed to write the basic code by ourselves manually. Therefore, we surveyed and attempted several NLP traditional approaches to examine their performance in our system. After several approaches, we finally chose TF-IDF and cosine similarity techniques for similarity learning since it is easy to replicate with the minimum installation of external libraries in Java.

* 1. Challenges in Android App Development

Developing the JMatch presents several challenges. One of these challenges is backend development using HTTP GET and POST requests and processing the resulting data. These requests are used to retrieve and send data to and from a server and handling the responses can be complex. For example, in our project, we needed to implement an email verification function. To accomplish this, we had to create our own WeChat cloud function using the nodemailer package. This required a deep understanding of the package and its capabilities, and it was difficult to implement correctly.

Another challenge is building visual UI components from scratch. While many existing packages provide pre-built components, these may not always meet the specific needs of the project. For instance, we wanted to include a bar chart in our app but using an existing package would have introduced too many complex and unnecessary functions and made it difficult to achieve the desired visual effect. Instead, we used a Linear Layout to create the bar chart, with each bar implemented as a row in the layout. This required a detailed understanding of Android’s layout system and careful planning to ensure that the chart was displayed correctly.

Performance is also a major concern during our development process. Mobile devices have limited resources, and apps must be carefully optimized to run smoothly. To ensure that our app runs smoothly and avoids ANR (Application Not Responding) errors, we focused on programming efficiency and performed all database operations and other background tasks in background threads. We kept operations in the UI thread, which is the main thread, to a minimum so that the app could achieve 60 FPS (frames per second). This required careful planning and testing to ensure that all tasks were performed efficiently and did not impact the app’s performance.

* 1. Limitation

In this project, we aim to examine and realize the heated artificial intelligence technologies in our mobile application design. Therefore, we designed a job-matching app to use the NLP algorithm in the recruitment process. However, the algorithm design of this project is still in a traditional approach. Due to the time limit, we could not research and apply some complicated black-box learning approaches in our matching system, such as BERT for deep learning. In future development, we may attempt more possible uses of deep learning in career apps to examine the speed and performance of deep learning algorithms in mobile app development.

Another limitation of our project is the external API used in our mobile app. For example, the text input in our mobile app is still based on plain text and a simple PDF conversion library. Due to the business use limitation, we may not apply other useful techniques such as Optical Character Recognition (OCR) in our system for more meaningful string cleaning and input. In the future development, if we could have more resources to use, more advanced technologies and APIs could be attempted for more functions to our system to enhance our user experiences, such as text recognition technologies, recommendation engines, and aspect-based deep learning methods.

4 Conclusions

In this project, we finally built a cover letter-JD matching mobile app with an NLP similarity learning algorithm. With the application of artificial intelligence, our mobile app could offer a reliable reference to both HR managers and job seekers with data visualization on the mobile user interface, which creates a job-seeking ecosystem for the talent acquisition process.

This project also has some problems that remain unsolved. Due to the limited development time, we could not build a solid deep-learning algorithm to filter job descriptions and cover letters. In addition, the upload process of resumes and JD could also be further improved with more advanced functions and possible OCR use. It is hoped that with further survey and development, the system could be further enhanced to be more powerful.

The ultimate goal our team would like to pursue is building a job-seeking ecosystem for both candidates and recruiters. In the future, it is hoped that we could improve the app design and the training of our matching models. By offering more options and visualized functions on the Android mobile app development, we believe that we could create deep bonds between job seekers and HR managers.

5 Reference

1. D. Card, "What's New in HR Tech this year?" [Online]. Available: <https://isg-one.com/articles/what-s-new-in-hr-tech-this-year> [Accessed Feb. 20, 2023].
2. “Figma: The Collaborative Interface Design Tool,” Figma. [Online]. Available: <https://www.figma.com/proto/pCOwCq6fR4gGcdH3wtkD0u/JMatch?node-id=1-2> [Accessed Apr. 21, 2023].
3. P. K. Roy, S. S. Chowdhary, and R. Bhatia, "A Machine Learning Approach for Automation of Resume Recommendation System," Procedia Computer Science, vol. 167, pp. 2318–2327, 2020, DOI: 10.1016/j.procs.2020.03.284.
4. N. Sethi, “TF-IDF for Similarity Scores,” Data-Driven Investor, Oct. 5, 2019. [Online]. Available: <https://medium.datadriveninvestor.com/tf-idf-for-similarity-scores-391c3c8788e8> [Accessed: Apr. 21, 2023].

6 Appendix A: Testing Documents

All the testing documents and the documents stored in the database can be found at <https://drive.google.com/drive/folders/1uzAIo6OUytz5fkqYh7pBsY0zIvDcJgQw>.

**Self-testing Account（All Candidate Account）**

|  |  |  |
| --- | --- | --- |
| **Account** | **Password** | **Testing File** |
| dongdong | 000000 | sample |
| ethan | 00000000 | sample |

**Documents to Populate the Database**

**Candidates**

|  |  |  |
| --- | --- | --- |
| **Account** | **Password** | **Testing File** |
| Jack Chan | 0 | CL-1 |
| Amy Leung | 0 | CL-2 |
| John Ip | 0 | CL-3 |
| Alan Wu | 0 | CL-4 |
| Mike Sze | 0 | CL-5 |
| Coco | 0 | CL-6 |
| Lucy Wong | 0 | CL-7 |
| Joshua Hu | 0 | CL-8 |
| Winson Co | 0 | CL-9 |
| July Pak | 0 | CL-10 |
| Lily Xue | 0 | CL-11 |

**Enterprises**

|  |  |  |
| --- | --- | --- |
| **Account** | **Password** | **Testing File** |
| AMERIA | 0 | JD-1 |
| Caucasus Environmental NGO Network | 0 | JD-2 |
| Manoff Group | 0 | JD-3 |
| FRCS | 0 | JD-4 |
| Teleplus LLC | 0 | JD-5 |
| NetCall Communications | 0 | JD-6 |
| SOC.Stockholm | 0 | JD-7 |
| UN Development Programme | 0 | JD-8 |
| Counterpart International, Inc. | 0 | JD-9 |
| CUTS Centre | 0 | JD-10 |
| Yerevan Brandy Company | 0 | JD-11 |

**Presentation Demo Use**

**Candidates**

|  |  |  |
| --- | --- | --- |
| **Account** | **Password** | **Testing File** |
| Louis | TBD | CL-12 |
| TBD | TBD | CL-13 |

**Enterprises**

|  |  |  |
| --- | --- | --- |
| **Account** | **Password** | **Testing File** |
| ACDI/VOCA | TBD | JD-12 |
| Armenia TV | TBD | JD-13 |