**High Level Design (HLD)**

**Job Recommendation System**

| Written By | Rakshit Khajuria, Prikshit Sharma |
| --- | --- |
| Last Revised Date | 11-March-2023 |

**Contents**

Abstract —------------------------------------------------------------------------------------------------------ 2

1 Introduction —----------------------------------------------------------------------------------------------- 3

1.1 Why this High-Level Design Document? —----------------------------------------------------- 3

1.2 Scope —------------------------------------------------------------------------------------------------- 3

2 General Description —--------------------------------------------------------------------------- 4

2.1 Problem statement —--------------------------------------------------------------------------------- 4

2.2 PROPOSED SOLUTION —------------------------------------------------------------------------- 4

2.3 FURTHER IMPROVEMENTS —------------------------------------------------------------------- 4

2.4 Technical Requirements —-------------------------------------------------------------------------- 5

2.5 Data Requirements —-------------------------------------------------------------------------------- 5

2.6 Tools used —-------------------------------------------------------------------------------------------- 6

2.7 Constraints —------------------------------------------------------------------------------------------- 7

3 Design Details —-------------------------------------------------------------------------------------------- 8

3.1 Process Flow —---------------------------------------------------------------------------------------- 8

3.1.1 For Candidate —--------------------------------------------------------------------------------- 8

3.1.2 Resume Analyzer —----------------------------------------------------------------------------- 8

3.1.3 For Recruiter —----------------------------------------------------------------------------------- 9

3.2 Text preprocessing —--------------------------------------------------------------------------------- 9

3.3 Model Training and Evaluation —------------------------------------------------------------------ 10

3.4 Database ------------------------------------------------------------------------------------------------- 13

3.5 Error Handling —--------------------------------------------------------------------------------------- 13

3.5.1 Type of Errors —----------------------------------------------------------------------------- 13

3.5.2 Error Handling Process —---------------------------------------------------------------- 14

3.6 Reusability —------------------------------------------------------------------------------------------- 14

3.7 Resource Utilization —------------------------------------------------------------------------------ 14

3.8 Deployment —----------------------------------------------------------------------------------------- 14

4 Conclusion —-------------------------------------------------------------------------------------------------15

**Abstract**

Recent trends are to build tall buildings in big cities as a way out of the current housing overpopulation problem. These new structures unveil problems that if not addressed in time could cause catastrophes of unimaginable impact. Some of those problems is the incidence of a fire threat happening upstairs in one of those buildings, medical emergencies due to any road accidents or mob that may cause threat to the human kind. This work discusses the implementation of the unmanned ground vehicles to spot the real location of the medical emergencies due to road mishap, mob or illegal activities such as hooliganism, snatching, robbery and the fire emergency and accordingly channelize or route them to the concerned helpline for quick mitigation and avoid disaster.

|  |  |
| --- | --- |

1. **Introduction**

**1\_1 Why this High-Level Design Document?**

The purpose of this High-Level Design (HLD) Document is to add the necessary detail to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions prior to coding, and can be used as a reference manual for how the modules interact at a high level.

The HLD will:

* Present all of the design aspects and define them in detail
* Describe the user interface being implemented
* Describe the hardware and software interfaces
* Describe the performance requirements
* Include design features and the architecture of the project
* List and describe the non-functional attributes like:
* Security
* Reliability
* Maintainability
* Portability
* Reusability
* Application compatibility
* Resource utilization
* Serviceability

**1.2 Scope**

The HLD documentation presents the structure of the system, such as the database architecture, application architecture (layers), application flow (Navigation), and technology architecture. The HLD uses non-technical to mildly-technical terms which should be understandable to the administrators of the system.

1. **General Description**

**2.1 PROBLEM STATEMENT**

Job seekers and recruiters struggle to find the right match for open job positions, leading to a time-consuming and inefficient recruitment process. TalentHive offers a solution to this problem with its advanced technologies that provide personalized job and candidate recommendations based on qualifications and experience.

**2.2 PROPOSED SOLUTION**

This app is designed to assist applicants in searching for potential jobs and to help recruiters find talented candidates. The app offers a user-friendly interface that allows applicants to easily browse and search for job opportunities based on their preferences and qualifications. Users can create a profile, upload their resumes, and set up job alerts to receive notifications about new job postings that match their criteria. The app also provides helpful tips and resources for applicants, such as resume writing guides and interview preparation tips.

Recruiters can use this app to post job openings, search for candidates based on their qualifications, and view applicant profiles and resumes. The app's advanced search features allow recruiters to narrow down their search to find the most qualified candidates quickly. The app also includes features for scheduling interviews and communicating with applicants directly through the app.

Overall, this app is an excellent tool for both job seekers and recruiters, making the job search and hiring process more efficient and effective.

**2.3 FURTHER IMPROVEMENTS**

1. Enhance the recommendation system to deliver precise and pertinent outcomes: Improve the recommendation system through advanced algorithms and data analysis techniques like using different deep learning Word Embeddings.
2. Establish a relationship with job posting firms to procure database/API access: Engage with job posting companies to gain access to their databases or APIs, expanding the range of job listings available on our platform and make it real time.
3. Encourage candidates to become part of the CV database: Increase the number and quality of CVs in our database by inviting the applicants.

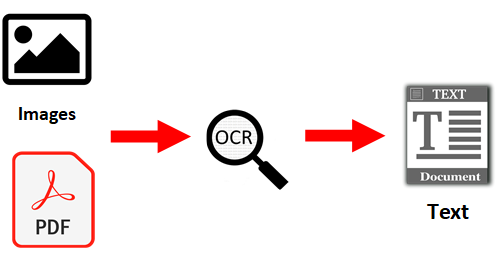
**2.4 TECHNICAL REQUIREMENTS**

* Cloning the Repository
* Creating Conda Environment
* Installing Required Packages
* Streamlit

**2.5 Data Requirements**

****

Due to web scraping restrictions imposed by job search engines such as Indeed, we sought out alternative means to collect pertinent job-related information. We turned to Apify, an advanced web scraping and automation tool, to gather the data we needed. With Apify's powerful capabilities, we were able to extract vast amounts of information in an efficient and accurate manner. This enabled us to provide up-to-date and precise job postings and candidate recommendations to both job seekers and recruiters, ensuring that they have access to the most relevant information available.

****

We were able to convert a PDF document containing textual data into a machine-encoded text format by utilizing Optical Character Recognition (OCR) technology. This allowed us to further process and analyze the information in a Jupyter Notebook. OCR technology is a highly advanced tool that automatically converts scanned documents and images into machine-encoded text, which can be easily manipulated and analyzed. By leveraging OCR technology, we were able to extract valuable insights from the textual data, providing us with a deeper understanding of the information contained within the document. This technology allowed us to work with the text data in a more efficient and effective way, enabling us to make the most of the information available to us.

**2.6 Tools used**

Python programing language and frameworks such as Numpy, Pandas. Scikit-Learn, Streamlit, plotly, MongDB, Pytesseract etc are used to build the whole model and the UI.

****

**2.7 Constraints**

1. For recommending the jobs to the candidate :-

We are currently exploring several job domains that have promising opportunities and potential for growth. These fields include:

1. Human Resources
2. Blockchain Developer
3. Cloud Engineer
4. Cyber Security
5. Data Analyst/Business Analyst
6. Data Scientist
7. Database Engineer
8. Frontend Developer
9. Machine Learning Engineer
10. Product Manager
11. Software Engineer

2. For resume analyzer we are recommending skills and courses belonging to following domains :-

1. Data Science
2. Web Development
3. Android Development
4. IOS Development
5. UI-UX Development

3. For Resume recommender , as we are having very limited number of resumes , and most of them are from data science domain , so by increasing number of resumes we can improve the resume recommendation

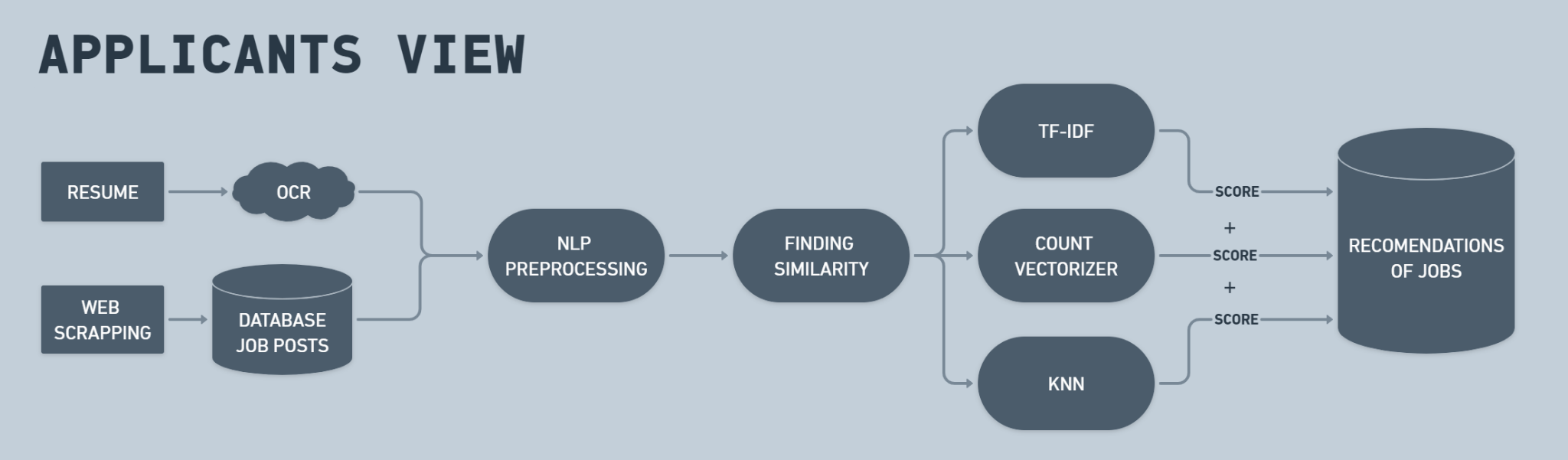
**3 Design Details**

**3.1 PROCESS FLOW**

For recommending the types of job based on the resume and shortlisting the candidates based on the Job Description we utilized CountVectorizer, TFIDF, KNN and Cosine Similarity. Below is the Process flow diagram.

**3.1.1 FOR CANDIDATE**

To assist job seekers, the process begins with uploading their CV to TalentHive. The CV is then processed by Optical Character Recognition (OCR) technology and undergoes Natural Language Processing (NLP). The NLP process compares the uploaded CV with various job postings to determine the best match based on similarity. Finally, the system provides a list of recommended jobs that match the user's qualifications and experience.

****

**3.1.2 RESUME ANALYZER**

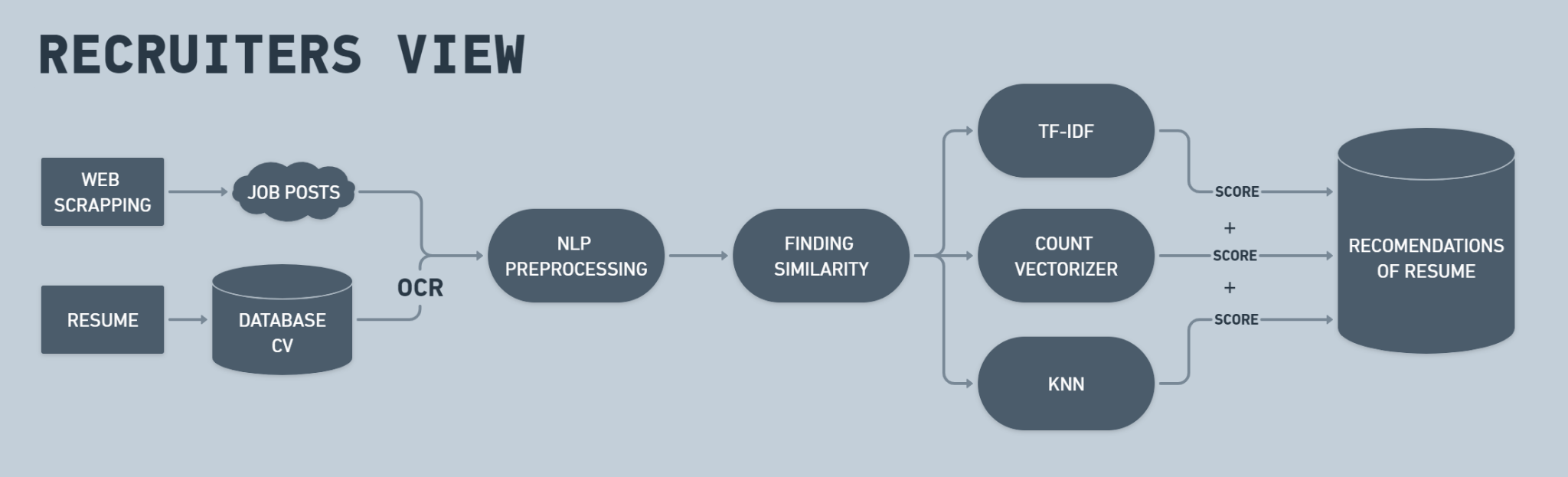
Extracting user's information from the Resume,by using PyResparser then Extracting Resume PDF into Text, by using PDFMiner.

Features

* Resume Score
* Career Recommendations
* Resume writing Tips suggestions
* Courses Recommendations
* Skills Recommendations

**3.1.3 FOR RECRUITERS**

For recruiters, the process begins with inputting the job post into TalentHive. The job post undergoes NLP analysis alongside the CVs in the database. Using various comparison methods, TalentHive identifies the best-matching candidates for the job posting. The system then provides a list of recommended candidates to the recruiter based on their qualifications and experience.



For recruiters, the process begins with inputting the job post into TalentHive. The job post undergoes NLP analysis alongside the CVs in the database. Using various comparison methods, TalentHive identifies the best-matching candidates for the job posting. The system then provides a list of recommended candidates to the recruiter based on their qualifications and experience.

**3.2 Text preprocessing**

Data preprocessing is an essential step in building a Machine Learning model and depending on how well the data has been preprocessed; the results are seen.

In NLP, text preprocessing is the first step in the process of building a model.

Some text preprocessing steps used are:

Word tokenization:

Stop words removal

Lemmatization

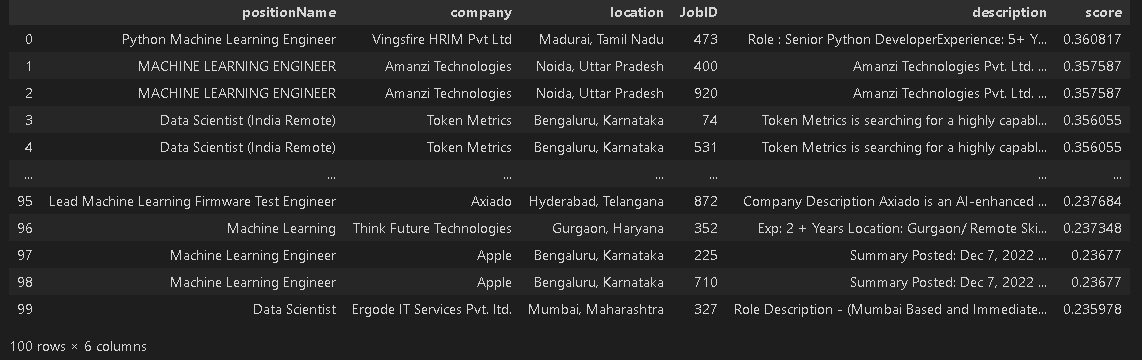
Bigram Collection Finder

**3.3 Model Training and Evaluation**

**Recommendation Strategy:**

| 1 | 2 | 3 |
| --- | --- | --- |
| count vectorizer | TF - IDF | TF - IDF |
| cosine similarity | cosine similarity | KNN |

**Countvectorizer + Cosine Similarity = Score 1**



**TF - IDF + Cosine Similarity = Score 2**

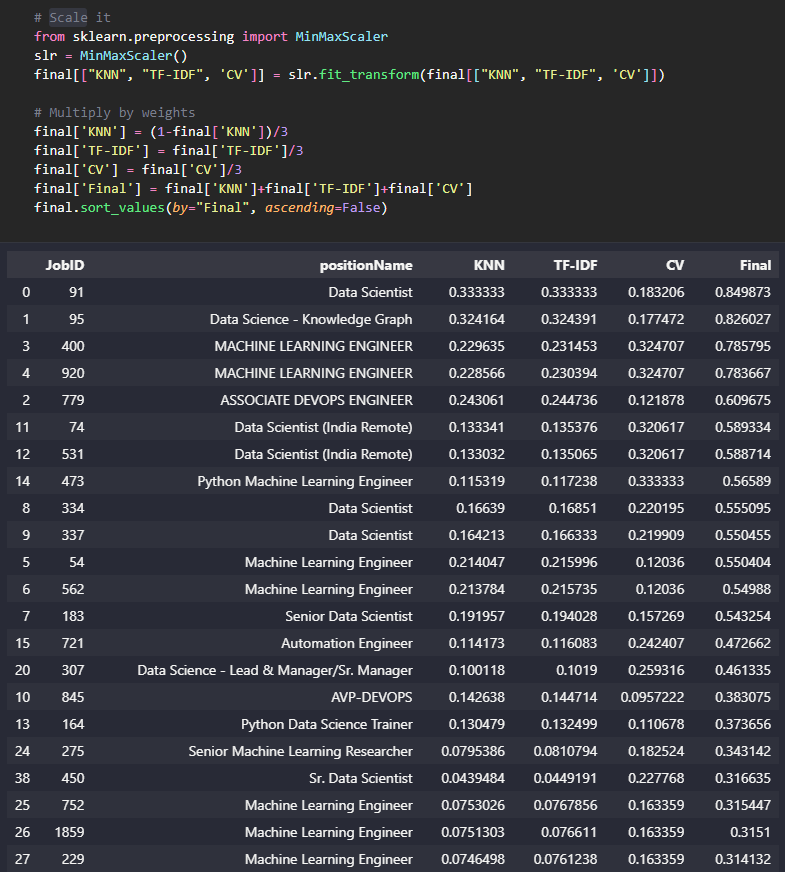


**TF - IDF + KNN = Score 3**



**We will scale and add all the three scores to make recommendations.**

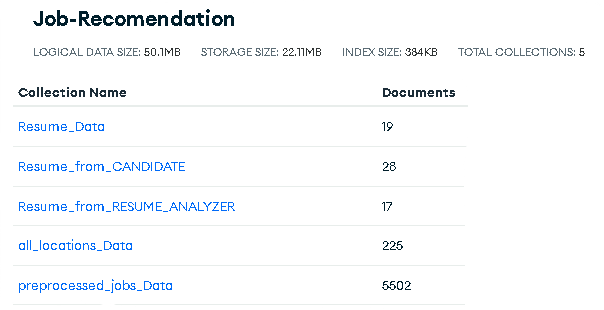
**FINAL PREDICTIONS AFTER SCALING THE SCORES**



**3.4 Database**

**MongoDB**

MongoDB is a cross-platform document-oriented database program that uses JSON-like documents with optional schemas. It is a NoSQL database that stores data in flexible, JSON-like documents, which makes it easier to store and process complex data types, such as nested arrays and objects. MongoDB is designed for scalability, high availability, and high performance.



**3.5 Error Handling**

Error handling is used to handle errors and exceptions in the system. The goal is to ensure that the system can identify and respond to unexpected errors in a consistent and user-friendly way, while also providing effective feedback to developers and system administrators.

**3.5.1 Types of Errors**

The system may encounter various types of errors during operation, including syntax errors, runtime errors, logic errors, and external errors such as network or database errors. To ensure effective error handling, the system will identify and categorize errors based on their type, severity, and impact on system functionality.

**3.5.2 Error Handling Process**

The error handling process will follow a set of guidelines to ensure consistency and effectiveness. When an error occurs, the system will log the error details, display an error message to the user if appropriate, and perform appropriate error handling and recovery actions based on the type and severity of the error. In some cases, errors may trigger automatic recovery actions, while in others, manual intervention may be required.

**3.6 Reusability**

Our code was designed to be modular, flexible, and easily adaptable to different use cases. By doing so, we ensured that it was reusable and could be applied to other projects or contexts without issue. This saved time and resources, maintained consistency, and improved the quality and reliability of the system. To ensure code reusability, we followed best practices for software design and development, including using established design patterns and libraries, adhering to coding standards and conventions, and thoroughly documenting our code.

**3.7 Resource Utilization**

When any task is performed, it will likely use all the processing power available until that function is finished based on the computer specifications.

**3.8 Deployment**

****

We deployed our system on Streamlit, an open-source framework for creating and sharing data apps quickly and easily. Streamlit was chosen for its ease of use, ability to handle different types of data visualizations, and compatibility with different programming languages. The deployment process involved setting up a Streamlit account, developing the necessary code for the app, and testing it locally to ensure it functioned properly. Once the code was ready, we deployed it on Streamlit for others to access and use.

**4 Conclusion**

In summary, TalentHive utilizes advanced technologies such as OCR and NLP to provide job seekers and recruiters with personalized job and candidate recommendations. Additionally, the use of Apify allows for efficient and accurate data collection, leading to up-to-date and relevant job postings. TalentHive streamlines the recruitment process, saving time and resources for both job seekers and recruiters.