



Tribhuvan University

Institute of Science and Technology

A Final Year Project Report

On

PERSONALITY EVALUATION AND CV ANALYSIS

Submitted To

Department of Information Technology

Kathford International College of Engineering and Management

**In partial fulfillment of the requirement for the Bachelor Degree in Computer
Science and Information Technology**

Submitted By

Alok Dugar (25868/077)

Ashim Khadka (25873/077)

Priyanjan Subba (25892/077)

Under the Supervision of

Mukesh Prasad Chaudhary

January, 2025

ACKNOWLEDGEMENT

It is a matter of great pleasure to present this project “PERSONALITY EVALUATION AND CV ANALYSIS” as our Major Project. We are grateful to Institute of Science and Technology for including major project in the syllabus of B.Sc. CSIT IV year. We are also thankful to the college management for providing us this great opportunity and managing the resources and specialists to assist our project.

Our heartfelt thanks go to **Mr. Mukesh Prasad Chaudhary**, our esteemed Project Supervisor, for his unwavering support, guidance, and expert insights throughout the project's lifecycle. His mentorship and encouragement have been instrumental in shaping the direction and success of our work. We would like to express our genuine appreciation to our Project Coordinator, **Mrs. Sarita Neupane** for her invaluable guidance and encouragement throughout the project development.

We would also like to express our gratitude to **Mr. Hari Prasad Pokhrel, Mr. Ishwor Dhungana, Mr. Suwas Karki** for their intuitive suggestions and advices.

We are also deeply grateful to our friends and family for their continuous support, understanding, and encouragement. Their patience and belief in us gave us the strength to overcome challenges and stay motivated throughout this journey.

ABSTRACT

This project introduces an automated CV ranking system designed to streamline the recruitment process, making it more efficient, fair, and consistent. Recruitment often involves subjective biases and time-consuming manual evaluations, which can hinder the selection of the most suitable candidates. To address these challenges, the system combines CV analysis with a personality assessment using the Big Five Personality Test. Candidates upload their CVs, which are analyzed by a Named Entity Recognition (NER) model trained on manually annotated job descriptions and resumes to extract key information such as qualifications, skills, and experience. Synonyms are standardized using the WordNet dictionary, and the TF-IDF algorithm with cosine similarity is applied to match CVs with job requirements. An optional aptitude test further refines the evaluation process. By integrating advanced AI techniques, this system simplifies and enhances shortlisting, ensuring fair, unbiased, and data-driven decisions that assist HR departments in identifying the best candidates.

Keywords: workforce, recruitment, WordNet, NER, NLTK, TF-IDF

TABLE OF CONTENTS

ACKNOWLEDGEMENT	i
ABSTRACT.....	ii
LIST OF FIGURES	v
LIST OF TABLES	vi
LIST OF ABBREVIATIONS.....	vii
CHAPTER 1: INTRODUCTION	1
1.1 INTRODUCTION.....	1
1.2 PROBLEM STATEMENT.....	2
1.3 OBJECTIVES	2
1.4 SCOPE AND LIMITATIONS	3
1.4.1 SCOPE	3
1.4.2 LIMITATIONS	3
1.5 DEVELOPMENT METHODOLOGY	4
1.6 REPORT ORGANIZATION.....	4
CHAPTER 2: BACKGROUND STUDY AND LITERATURE REVIEW	6
2.1 BACKGROUND STUDY	6
2.2 LITERATURE REVIEW	7
CHAPTER 3: SYSTEM ANALYSIS.....	9
3.1 REQUIREMENT ANALYSIS.....	9
3.1.1 FUNCTIONAL REQUIREMENTS	9
3.2. FEASIBILITY ANALYSIS	11
3.2.1 TECHNICAL FEASIBILITY	11
3.2.2 OPERATIONAL FEASIBILITY	11
3.2.3 ECONOMIC FEASIBILITY	11
3.2.4 SCHEDULE FEASIBILITY	12
3.3 ANALYSIS.....	12
3.3.1 DATA MODELING WITH ENTITY RELATIONSHIP DIAGRAM	12
3.3.2 PROCESS MODELING WITH DATA FLOW DIAGRAM.....	13
CHAPTER 4: SYSTEM DESIGN.....	20
4.1 DESIGN.....	20
4.1.1 SYSTEM FLOW DIAGRAM	20

4.1.2 SCHEMA DIAGRAM	21
4.1.3 FORM AND REPORT DESIGN	22
4.1.4 INTERFACE DESIGN	25
4.2 ALGORITHM DETAILS.....	26
4.2.1 TF-IDF ALGORITHM.....	26
4.2.2 COSINE SIMILARITY.....	27
CHAPTER 5: IMPLEMENTATION AND TESTING	28
5.1 IMPLEMENTATION	28
5.1.2 TOOLS USED	28
5.1.2 IMPLEMENTATION DETAILS OF MODULES.....	29
5.2 TESTING.....	40
5.2.1 UNIT TESTING.....	40
5.2.2 INTEGRATION TESTING	41
5.2.3 SYSTEM TESTING	43
5.3 RESULT ANALYSIS	45
CHAPTER 6: CONCLUSION AND FUTURE RECOMMENDATIONS.....	46
6.1 CONCLUSION	46
6.2 FUTURE RECOMMENDATIONS.....	46
REFERENCES	47
APPENDICES	48

LIST OF FIGURES

Figure 1.5.1: Agile Development Process	4
Figure 3.2.1.1: Use Case Diagram of Personality Evaluation & CV Analysis System	10
Figure 3.3.4.1: Gantt chart	12
Figure 3.4.1.1: ER Diagram of Personality Evaluation & CV Analysis System	13
Figure 4.1.1.1: System Flow Diagram of Personality Evaluation & CV Analysis System	20
Figure 4.1.2.1: Schema Diagram of Personality Evaluation & CV Analysis System	21
Figure 4.1.3.1: Employee Sign Up	22
Figure 4.1.4.2: Company Sign Up	23
Figure 4.1.5.3: Employee Login	23
Figure 4.1.6.4: Company Login	23
Figure 4.1.7.5: Job Application Form	24
Figure 4.1.8.6: Aptitude Test Form	24
Figure 4.1.9.7: Report Design	24
Figure 4.2.1.10: TF-IDF Algorithm	26
Figure 5.2.2.1: CV Ranking System Flow Diagram	29
Figure 5.3.2.2: NER Training	31
Figure 5.4.2.3: WordNet Synonym Replacement Example	33
Figure 5.5.2.4: Synonym Replacement using WordNet	34
Figure 5.6.2.5: Big 5 Personality Test	39
APPENDICES	48

LIST OF TABLES

Table 5.2.1.1: Unit Testing	41
Table 5.2.2.1: Integration Testing.....	43
Table 5.2.3.1: System Testing.....	44
Table 5.3.1: Accuracy for Job Description Annotation	45
Table 5.3.2: Accuracy for CV Annotation	45

LIST OF ABBREVIATIONS

ATS – Applicant Tracking System

CV – Curriculum Vitae

DFD – Data Flow Diagram

ER – Entity Relationship

HR – Human Resources

JSON – Java Script Object Notation

LLC – Limited Liability Company

ML – Machine Learning

NER – Named Entity Recognition

NLP – Natural Language Processing

NLTK – Natural Language Tool Kit

SQL – Structured Query Language

NoSQL – Not Only SQL

TF-IDF – Term Frequency – Inverse Document Frequency

CHAPTER 1: INTRODUCTION

1.1 INTRODUCTION

In today's competitive job market, finding the right candidate for a role has become an increasingly challenging task for recruiters. With an overwhelming number of applicants vying for every position, the recruitment process is often a lengthy and complex ordeal. Evaluating and shortlisting candidates based solely on their CVs is not only time-consuming but also prone to human error and inconsistencies. Recruiters face the daunting challenge of thoroughly analyzing multiple aspects of a candidate's profile, such as their educational qualifications, professional experience, skills, and personality traits. Traditional methods of manually assessing resumes, conducting aptitude tests, and evaluating personality traits are no longer sufficient to meet the fast-paced and highly demanding requirements of modern recruitment. The lack of an efficient and unified system to evaluate candidates often results in delays, misjudgments, and missed opportunities for both employers and job seekers.

The job application process is a critical phase in an individual's career journey. After completing education, securing the right job becomes one of the most significant milestones. However, representing oneself effectively through a Curriculum Vitae (CV) or Resume can be a daunting challenge, particularly in a world where employers receive hundreds, if not thousands, of applications for every open position. Despite the widespread adoption of online job portals and technological advancements in recruitment, HR departments often find themselves overwhelmed by the volume of applications they need to evaluate. Even with predefined CV formats and guidelines provided by companies, the manual process of reviewing resumes remains tedious, repetitive, and monotonous, leading to inefficiencies in identifying the most qualified candidates.

To address these challenges, this project introduces an intelligent, automated recruitment system that integrates CV analysis, aptitude evaluation, and personality assessment into a unified online platform. By leveraging advanced machine learning techniques, data analytics, and recommendation algorithms, the system aims to

streamline the recruitment process, enabling organizations to identify the most suitable candidates for a given role with greater speed and accuracy. This innovative platform will not only analyze CVs for completeness and relevance but will also incorporate aptitude test results and personality assessments to provide a holistic evaluation of candidates.

The benefits of such an approach extend to both employers and job seekers. Employers can make data-driven decisions, ensuring transparency and fairness in the hiring process, while job seekers benefit from an objective and efficient evaluation system that increases their chances of being considered for roles that align with their skills and potential. Ultimately, this project envisions creating an intelligent recruitment platform that fosters efficiency, transparency, and reliability, transforming the way organizations and candidates interact in the job market.

1.2 PROBLEM STATEMENT

Finding the right candidate for a job has become increasingly challenging for companies, as their success often hinges on the quality of their workforce. The rising mobility of job seekers has led to an explosion in the number of applications for many positions, forcing recruiters to sift through hundreds or even thousands of CVs to identify the ideal match. This time-intensive and error-prone process underscores the need for automation, particularly in ranking and scoring CVs based on their relevance to job requirements. Our system addresses this challenge by advancing traditional resume ranking methodologies, incorporating enhanced Natural Language Toolkit (NLTK) processing and introducing a personality evaluation feature to provide a comprehensive assessment of candidates. By offering deeper insights and greater flexibility for both employers and job seekers, this system aims to optimize recruitment processes, ensuring fair, efficient, and accurate matching of talent to opportunities.

1.3 OBJECTIVES

- To implement a CV grading web application accompanied by personality evaluation with the use of NER model and TF-IDF algorithm.
- To simplify the hiring process by grading CVs.

1.4 SCOPE AND LIMITATIONS

The scope and limitations, which form the foundation of this project's framework, are outlined comprehensively in the following sections. By delving into both the extent of the project's capabilities and the inherent constraints it faces, a clearer understanding of its operational boundaries emerges. These details are presented meticulously, ensuring a nuanced examination of its functionality and potential challenges.

1.4.1 SCOPE

The project is designed to enhance the recruitment process for organizations. The scope includes:

1. **CV Analysis:** The system evaluates and ranks candidates' CVs based on job requirements using machine learning techniques, ensuring accurate and efficient shortlisting.
2. **Integrated Personality and Aptitude Tests:** Incorporates personality assessments and aptitude evaluations to provide a comprehensive analysis of candidates' suitability for the role.
3. **Web-Based Platform:** Offers an online portal for recruiters and applicants, providing seamless access to candidate evaluations, test results, and rankings.

1.4.2 LIMITATIONS

The project, while robust, has certain limitations, including:

1. **Reliance on Input Quality:** The accuracy of CV evaluation and recommendations depends on the quality and completeness of the input data provided by candidates. Poorly formatted or incomplete CVs may lead to suboptimal results.
2. **Algorithm Bias:** Machine learning models may inadvertently inherit biases from the training data, potentially affecting fairness in candidate evaluation.

1.5 DEVELOPMENT METHODOLOGY

The Agile methodology was selected for the project due to its flexibility and adaptability, which were essential for managing dynamic and evolving requirements. Development was carried out in a repeatable, iterative manner, enabling continuous improvement throughout the project lifecycle. Regular interactions with mentors and supervisors ensured that feedback was consistently incorporated into the system's design and functionality. Additionally, small, functional increments were delivered frequently, allowing for early evaluation and refinement of components. This approach proved beneficial, particularly for addressing complex requirements through iterative refinement and active collaboration.

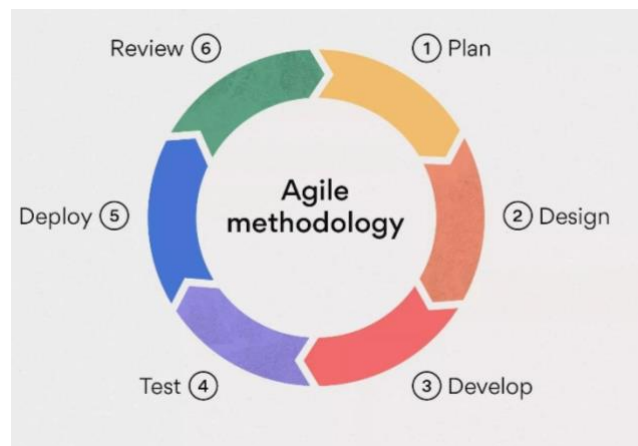


Figure 1.5.1: Agile Development Process

1.6 REPORT ORGANIZATION

The comprehensive documentation of the project adheres to a structured format for clarity and coherence:

Chapter 1: Introduction

This chapter introduces the project, highlighting the challenges faced by traditional recruitment methods and the need for automation in candidate evaluation. It outlines the project objectives, scope and limitations, emphasizing the system's role in improving recruitment efficiency and accuracy.

Chapter 2: Background Study and Literature Review

This chapter explores the theoretical foundations and related works underpinning the project. It highlights machine learning techniques and recommendation algorithms relevant to CV analysis, providing insights into their applications, strengths, and limitations.

Chapter 3: System Analysis

This section analyzes the architecture of the system, presenting diagrams such as Use-Case, ER Diagram, DFD. The functional and non-functional requirements are presented, ensuring the system aligns with organizational needs. It also encompasses feasibility analysis.

Chapter 4: System Design

This chapter focuses on the design framework, presenting diagrams such as System Architecture, Schema Diagram, etc. It also highlights the use of algorithms in the system.

Chapter 5: Implementation and Testing

This chapter explains the technical implementation of the system, including the tools, programming languages, and frameworks utilized. The development process is outlined, followed by the testing methodologies applied to ensure system performance and reliability.

Chapter 6: Conclusion and Future Recommendations

The final chapter summarizes the project outcomes, highlighting its contributions to streamlining the recruitment process. Recommendations for future improvements, such as enhancing algorithm fairness and expanding customization options for specific industries, are proposed to guide further advancements.

CHAPTER 2: BACKGROUND STUDY AND LITERATURE REVIEW

2.1 BACKGROUND STUDY

Recruitment systems have undergone significant transformations over the years, evolving from manual processes to advanced, technology-driven solutions. Initially, hiring processes relied heavily on human judgment to review resumes, conduct interviews, and assess candidates' suitability. These traditional methods, though effective to some extent, were time-consuming, prone to biases, and limited in their scalability.

The advent of Applicant Tracking Systems (ATS) in the 1990s marked a significant milestone in recruitment technology. ATS allowed organizations to manage job applications electronically, enabling keyword-based filtering to shortlist candidates. While this was an improvement, it often resulted in overlooking qualified candidates due to over-reliance on specific keywords.

By the 2010s, machine learning and artificial intelligence began transforming recruitment systems, offering smarter and more efficient tools for candidate evaluation. Recommendation systems, particularly content-based filtering and collaborative filtering, gained prominence for their ability to analyze and match candidate profiles with job requirements. Tools like "Career Mapper" emerged, leveraging algorithms to assess the completeness and relevance of candidates' resumes. These systems provided tailored recommendations based on content features such as skills, experiences, and education.

Another notable innovation was the integration of personality assessments and aptitude tests into recruitment workflows. Traditional methods of gauging personality relied on standardized questionnaires, such as the Big Five Inventory, while aptitude tests evaluated cognitive abilities. These tests became more efficient and accessible through online platforms, allowing recruiters to assess candidates remotely and integrate results into automated systems.

The rise of tools like "Jobscan" further refined the recruitment process by optimizing resumes for ATS compatibility. Jobscan analyzed resumes against specific job descriptions, providing insights on alignment and offering suggestions for improvement. This not only helped candidates tailor their applications but also reduced the burden on recruiters by ensuring higher-quality resumes.

Incorporating machine learning into these systems enabled predictive analytics, where algorithms could forecast a candidate's success in a role based on historical data. For instance, deep learning models now analyze not only textual data from resumes but also patterns in responses to assessments, offering a comprehensive evaluation of a candidate's potential.

Today, recruitment systems have advanced to include AI-driven platforms that perform end-to-end automation of hiring processes. These systems analyze resumes, conduct virtual interviews, and even predict cultural fit based on personality traits. While these innovations have streamlined recruitment, challenges such as algorithmic biases, data quality dependency, and the need for ATS compatibility continue to shape ongoing research and development in the field.

The evolution of recruitment technology reflects a broader trend toward automation and data-driven decision-making. With machine learning and recommendation algorithms at its core, modern systems like the one proposed in this project aim to provide fair, efficient, and scalable solutions for organizations, redefining how talent is identified and hired.

2.2 LITERATURE REVIEW

A CV, short for the Latin term *curriculum vitae* meaning "course of life," is a comprehensive document that details an individual's professional and academic history. CVs are indispensable during job applications as they significantly influence whether a candidate progresses to the interview stage [1].

Alongside CVs, job descriptions are crucial documents that outline a position's main duties and responsibilities. These descriptions help job seekers determine if a role aligns with their skills and interests while aiding recruiters in attracting suitable candidates.

Well-crafted job descriptions streamline the hiring process by reducing mismatches, ensuring candidates closely align with role requirements [2].

Modern recruitment processes have evolved significantly due to the influx of applications and technological advancements. Automated resume parsing systems have emerged as essential tools for recruiters. These systems analyze resumes to extract relevant information, such as education, skills, and work experience, and classify them into structured databases. This automation saves hours of manual work, particularly for organizations managing high application volumes, by quickly identifying suitable candidates based on predefined criteria [3].

Resume writing websites like 'resumeprofessionalwriters' provide ATS-friendly interview-winning professional-looking and keyword-optimized resumes [4]. Also, Resume parsers analyze a resume, extract the desired information, and insert the information into a database with a unique entry for each candidate [5]. Recruitment agencies work with CV/Resume Parsing tools to automate the storage and analysis of CV/Resume data. This saves recruiters' hours of work by eliminating manual processing of each job application and CV they receive. Once the resume has been analyzed, a recruiter can search the database for keywords and phrases and get a list of relevant candidates [6]. Many parsers support semantic search, which adds context to the search terms and tries to understand intent in order to make the results more reliable and comprehensive. The candidates returned are ranked based on how closely they match the keywords and job profile.

Machine learning is extremely important for resume parsing. Each block of information needs to be given a label and sorted into the correct category, whether that's education, work history, or contact information. Rule-based parsers use a predefined set of rules to parse the text. This method does not work for resumes because the parser needs to "understand the context in which words occur and the relationship between them." [7]. A rule-based parser would require incredibly complex rules to account for all the ambiguity and would provide limited coverage [8].

CHAPTER 3: SYSTEM ANALYSIS

3.1 REQUIREMENT ANALYSIS

The project utilizing machine learning technologies underwent a thorough requirements analysis to identify the critical components essential for its successful implementation. This analysis aimed to capture both user expectations and technical specifications, ensuring a comprehensive understanding of the project's requirements.

3.1.1 FUNCTIONAL REQUIREMENTS

- **Registration and Log In**

Users need to create a new account and register into the system either as an employee or an employer. System has to make sure that the user is registered in the application so the user must be authenticated by login process.

- **Job creation**

Users logged into the system as employers can create job objects along with requirements for the employees to apply to.

- **Job searching, viewing and applying**

The users of employee modules can search for and view jobs uploaded by the employers and also apply to the respective jobs.

- **Personality and Aptitude test**

The employees can also take standard personality tests and aptitude tests uploaded by the employers.

- **CV and Description annotation and analysis**

Uploaded CVs and description are annotated by the help of spaCy models and analyzed using TF-IDF algorithm with cosine similarity to rank the CVs on the basis of their similarities.

Use Case Diagram

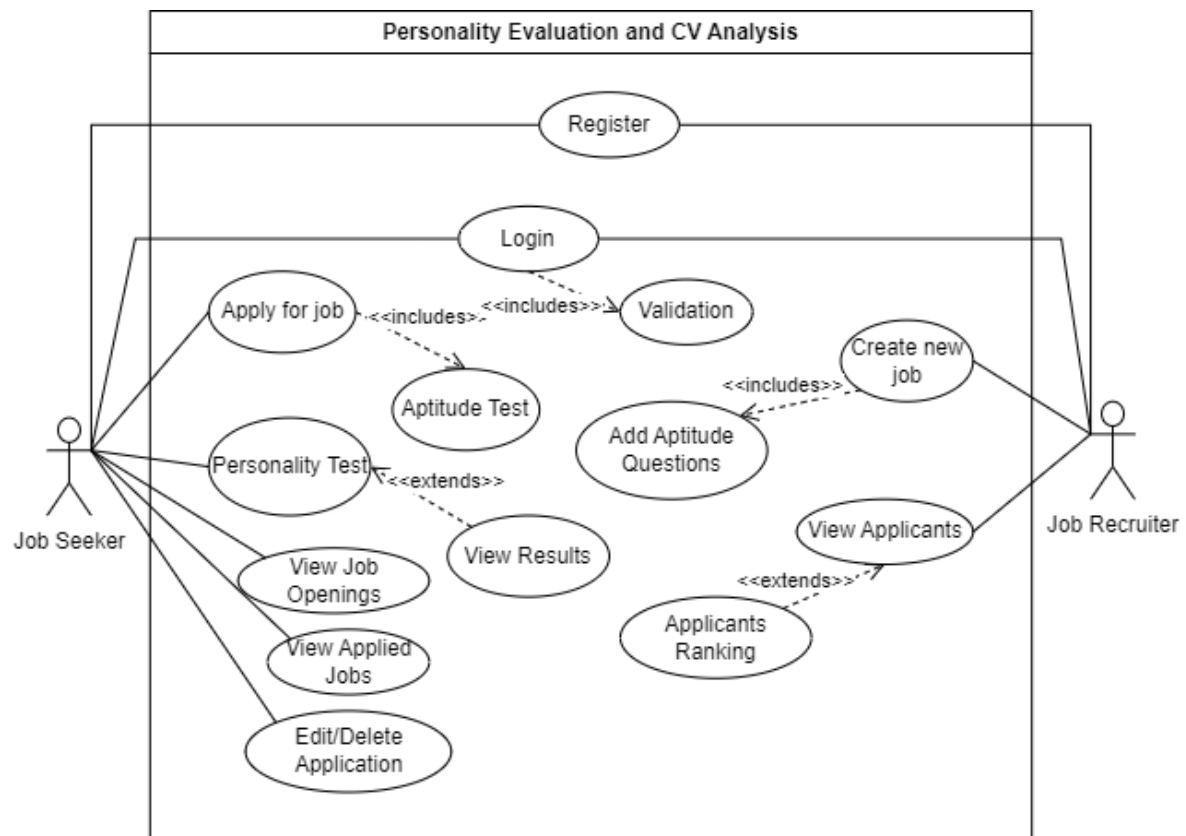


Figure 3.1.1.1: Use Case Diagram of Personality Evaluation & CV Analysis System

3.1.2 NON-FUNCTIONAL REQUIREMENTS

- **Security**

Security is achieved by registering the users and allowing access to only registered users via login.

- **Reliability and Accuracy**

Similar CVs were used as test inputs to ensure that the similarity values obtained are accurate and reliable.

- **Speed**

The similarity values are calculated only once and stored in the database for quick retrieval rather than complex calculations.

- **Data Integrity**

The system ensures that all data, including CVs, test scores, and user details, are stored and processed without corruption or loss.

- **Usability**

The system interface is user-friendly, ensuring that both recruiters and candidates can easily navigate and use the platform without requiring extensive training or support.

3.2. FEASIBILITY ANALYSIS

The system underwent a comprehensive feasibility analysis, evaluating key aspects to ensure viability and successful completion of the project.

3.2.1 TECHNICAL FEASIBILITY

The technical feasibility of the system was thoroughly assessed, and all necessary resources and technologies were determined to be readily available. The necessary hardware requirements, such as sufficient memory and processing power, are minimal due to the efficiency of the TF-IDF algorithm, which reduces the computational complexity of text analysis.

3.2.2 OPERATIONAL FEASIBILITY

The operational feasibility of the system was thoroughly assessed, evaluating and analyzing suitable CVs is a commonly faced problem in the recruitment process of any organization or company. The system can be used to minimize the work of HR departments in organizations.

3.2.3 ECONOMIC FEASIBILITY

It significantly reduces the cost and time involved in manual candidate evaluation processes. The initial investment in developing the system is offset by long-term savings through enhanced efficiency and reduced recruitment cycle times. Its ability to improve hiring accuracy further ensures better employee selection, reducing turnover costs and enhancing overall productivity.

3.2.4 SCHEDULE FEASIBILITY

The schedule feasibility of the CV analysis system was evaluated to ensure that the project could be completed within the allocated time frame. The Agile methodology was employed, allowing the project to be broken into smaller, manageable iterations with clear deliverables for each sprint. This iterative approach enabled the team to monitor progress regularly and adjust timelines as needed based on feedback and evolving requirements.

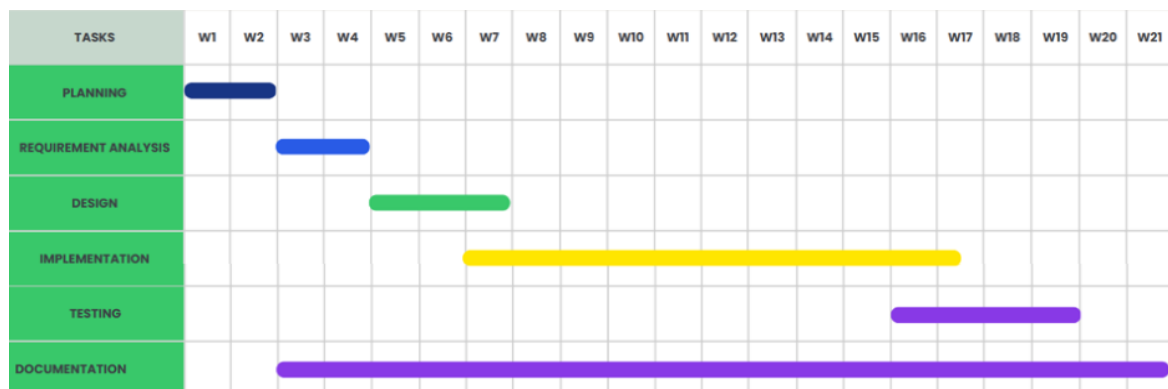


Figure 3.2.4.1: Gantt chart

3.3 ANALYSIS

3.3.1 DATA MODELING WITH ENTITY RELATIONSHIP DIAGRAM

The ER diagram below represents the system connecting users, employees, companies, and job-related information. Users can either be employees or companies. Employees have their personal details stored in a dedicated entity, linked to personality traits and aptitude tests to evaluate their compatibility with job requirements. Employees can submit job applications, including resumes, contact information, and aptitude scores. Companies manage job postings through a separate entity, detailing job categories and requirements like salary, experience, and location. A cosine similarity metric evaluates matches between employees and job roles, illustrating how users, employees, companies, jobs, and applications are interconnected.

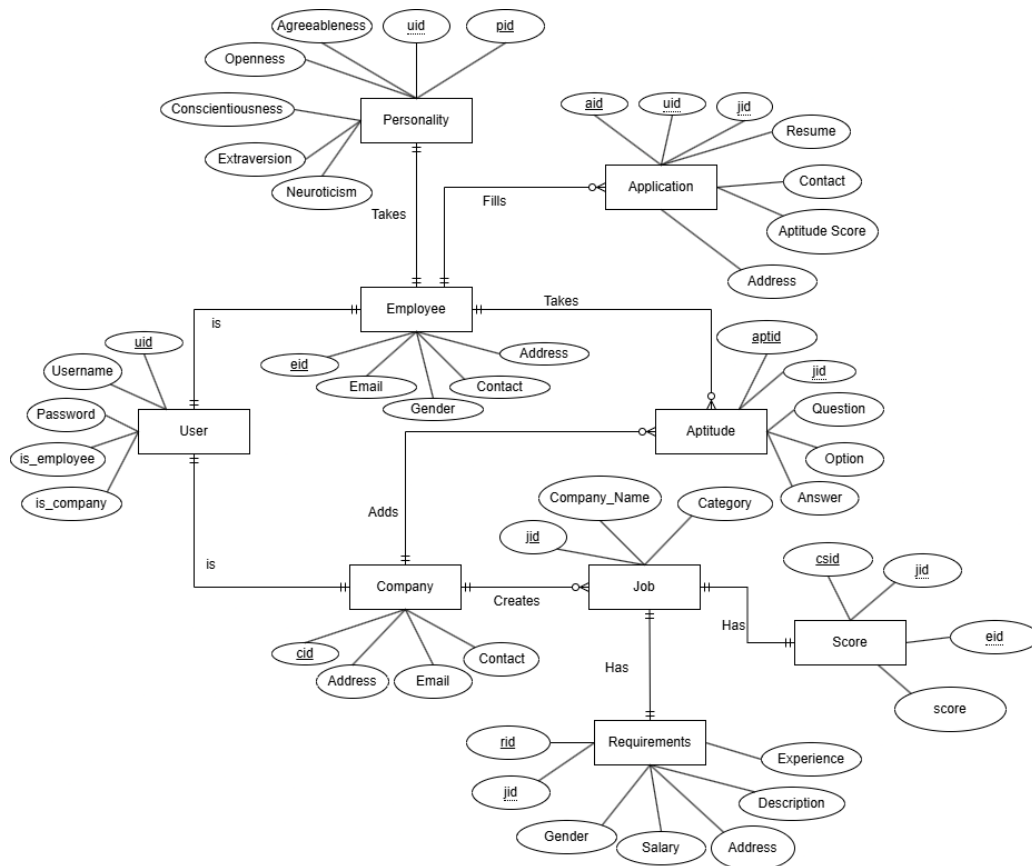


Figure 3.3.1.1: ER Diagram of Personality Evaluation & CV Analysis System

3.3.2 PROCESS MODELING WITH DATA FLOW DIAGRAM

The Context Flow Diagram (CFD) below illustrates the interaction between three main entities: Job Seekers, Job Recruiters, and a central "Personality Evaluation and CV Analysis System." Job Seekers submit login requests and, once authenticated, can request job application details, aptitude tests, and personality tests. Their responses (questionnaires or CVs) are submitted back to the system for evaluation. On the other hand, Job Recruiters interact with the system by submitting login requests, evaluation requests, and aptitude questions or job descriptions. The system processes these inputs to generate evaluation results, such as aptitude test scores and CV rankings, which are sent back to the recruiters. The diagram effectively highlights the bidirectional flow of information and the system's role in facilitating these interactions for employment evaluation.

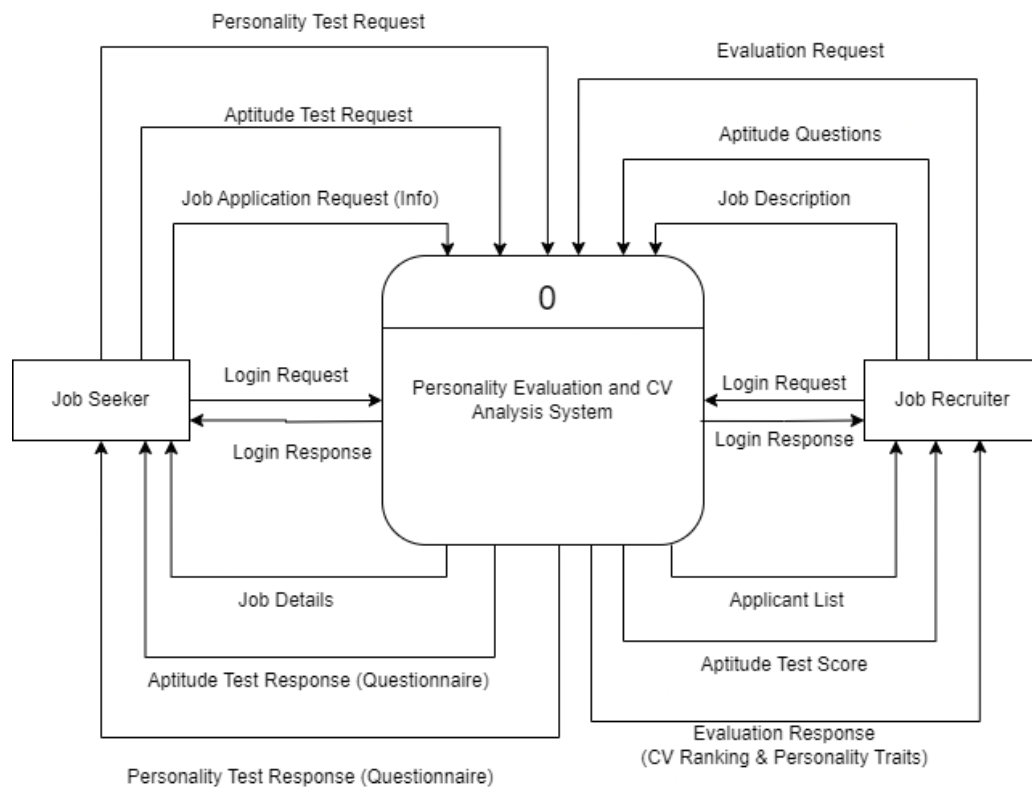


Figure 3.3.2.1: CFD (DFD Level 0) of Personality Evaluation & CV Analysis System

The Data Flow Diagram (DFD) below illustrates how the system matches job seekers and recruiters using four interconnected modules: the authentication system, the job system, the ranking system, and the personality system. The authentication system manages login requests, validates user credentials, and retrieves user records. The job system facilitates job applications, administers aptitude tests, and maintains job records, enabling job seekers to apply and recruiters to view applicants. The ranking system evaluates candidates by comparing their profiles to job requirements and ranks them based on similarity, assisting recruiters in selecting the most suitable applicants. The personality system conducts personality tests, processes results using the Big Five personality traits, and integrates this data into the ranking process. The diagram highlights the flow of data between job seekers, recruiters, and the system to ensure efficient job placement.

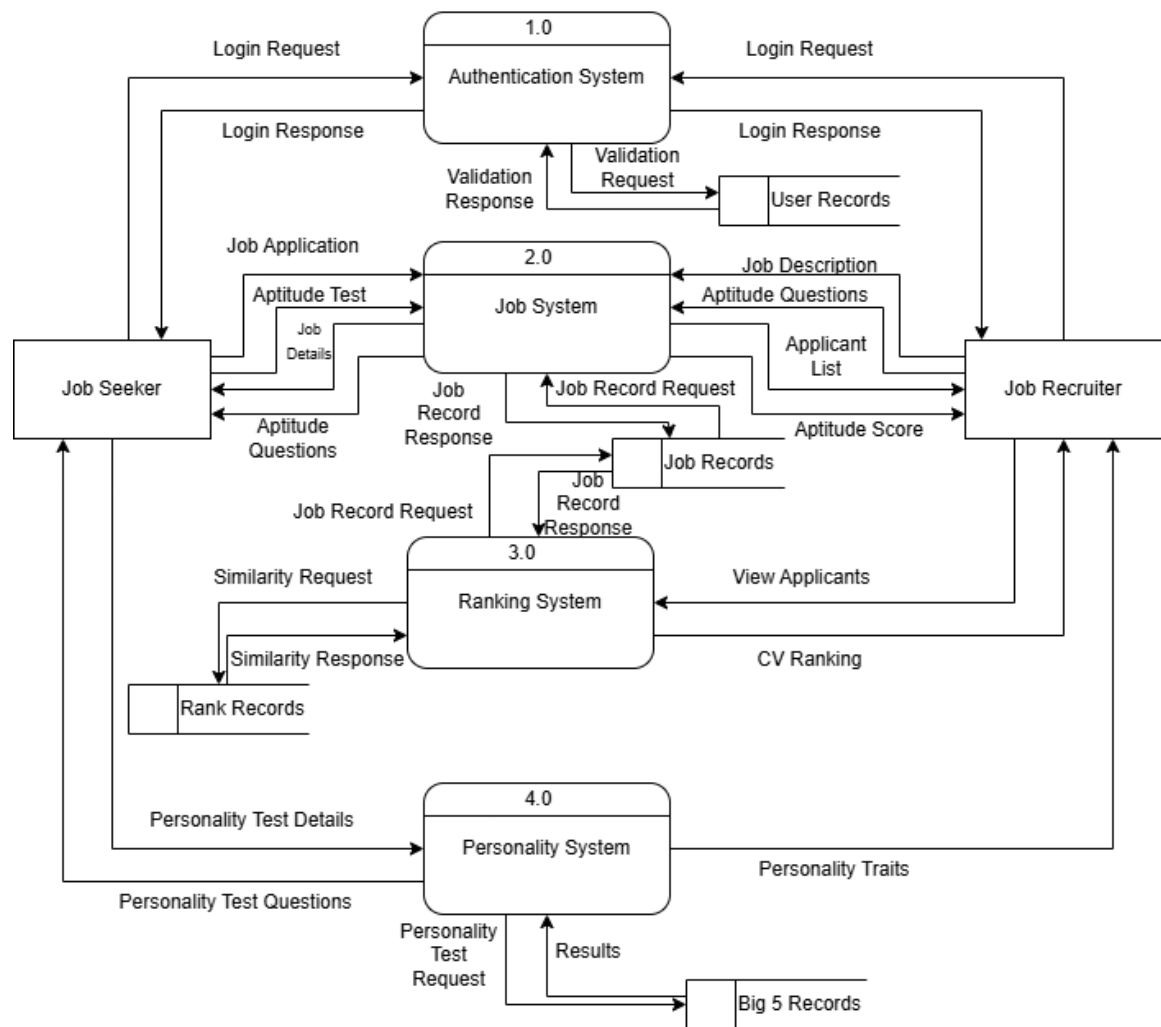


Figure 3.3.2.2: DFD Level 1 of Personality Evaluation & CV Analysis System

The Data Flow Diagram below illustrates the login process catering to job seekers and job recruiters. The job seeker and job recruiter serve as external entities interacting with the system. Both users initiate a login request, which is processed in the "Login Request Processing" stage. This process forwards the received credentials to the "Validate Credentials" stage for verification. The "Validate Credentials" process interacts with the user records database to verify the provided credentials by sending a validation request and receiving a response. Based on the validation outcome, a login response is sent back to the respective user, either granting or denying access.

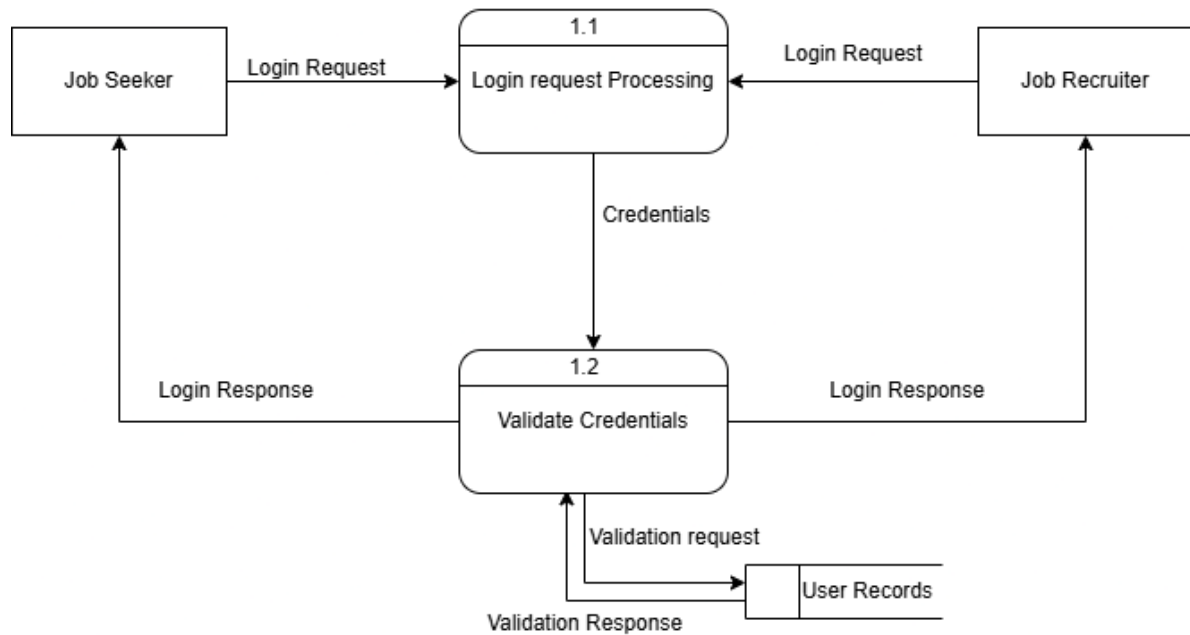


Figure 3.3.2.3: DFD Level 2 of Authentication System

The Data Flow Diagram below represents workflow for job applications and applicant management. The job seeker starts the process by submitting a job application. This triggers the "Job Application Submission" process, which requests aptitude questions from the system. The job seeker provides answers to these questions, and the responses are processed in the "Aptitude Test" phase, resulting in an aptitude score. The score, along with other job-related information, is stored in the job files.

Simultaneously, the job recruiter can request job records through the "Manage Records" process, which retrieves applicant data and provides a response. This process also generates an applicant list that is passed to the "Display List" stage, where recruiters can view the list of applicants. This streamlined flow ensures that job seekers can complete their applications efficiently, while recruiters have access to relevant applicant data.

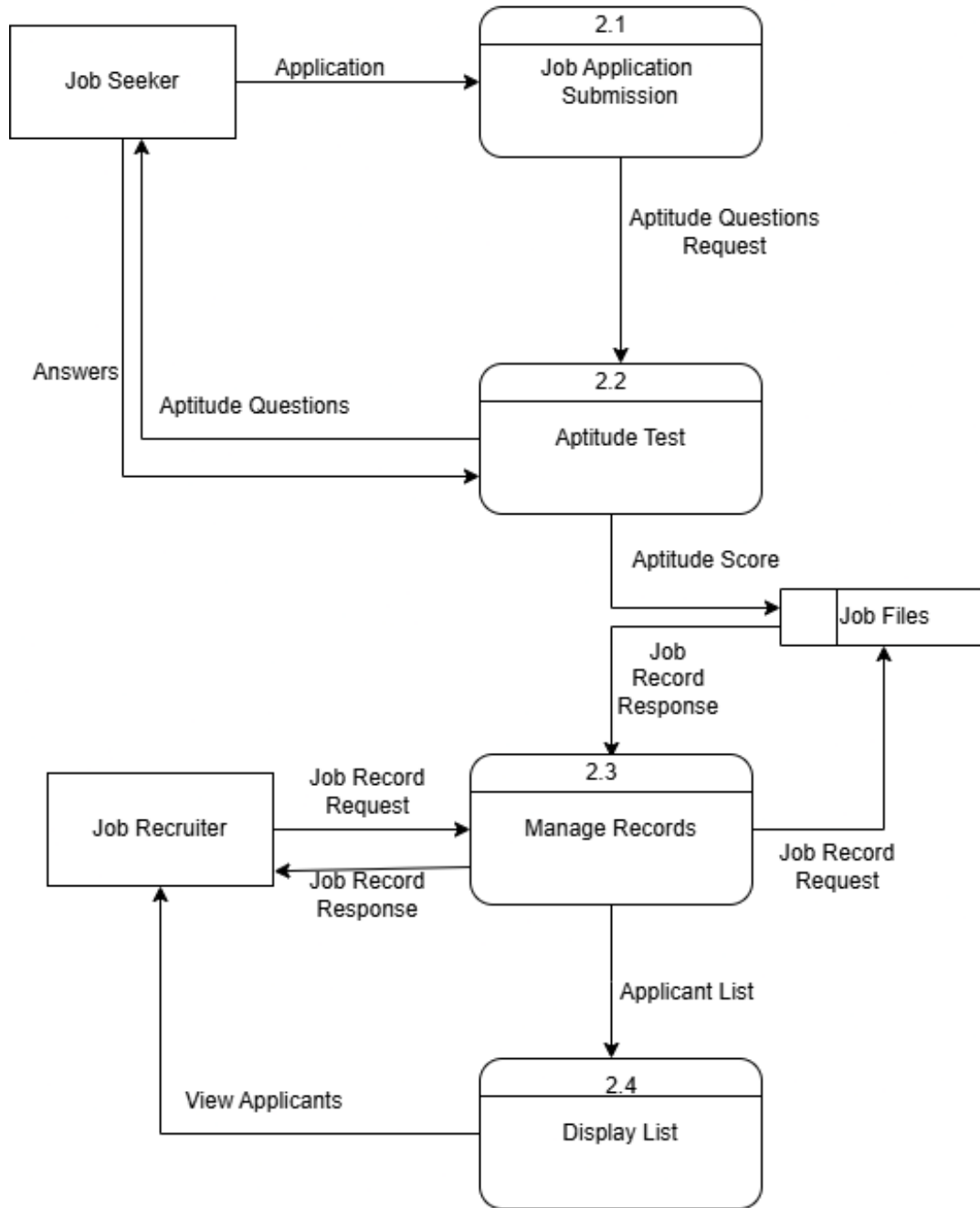


Figure 3.3.2.4: DFD Level 2 of Job System

The job seeker begins by submitting their CV, which is sent to the "Analyze Similarity" process. This process compares the CV with job descriptions obtained by sending a description request to the "Fetch Requirements" process. The "Fetch Requirements" process retrieves job requirements from job files and provides the necessary descriptions.

The "Analyze Similarity" process generates similarity scores, which are stored in rank files. These scores are used by the "Generate Ranking" process when a job recruiter sends a rank request. The "Generate Ranking" process then provides a ranking response to the recruiter, allowing them to view the ranked list of CVs based on their alignment with job requirements. This system ensures efficient CV evaluation and ranking for recruiters while providing feedback to job seekers.

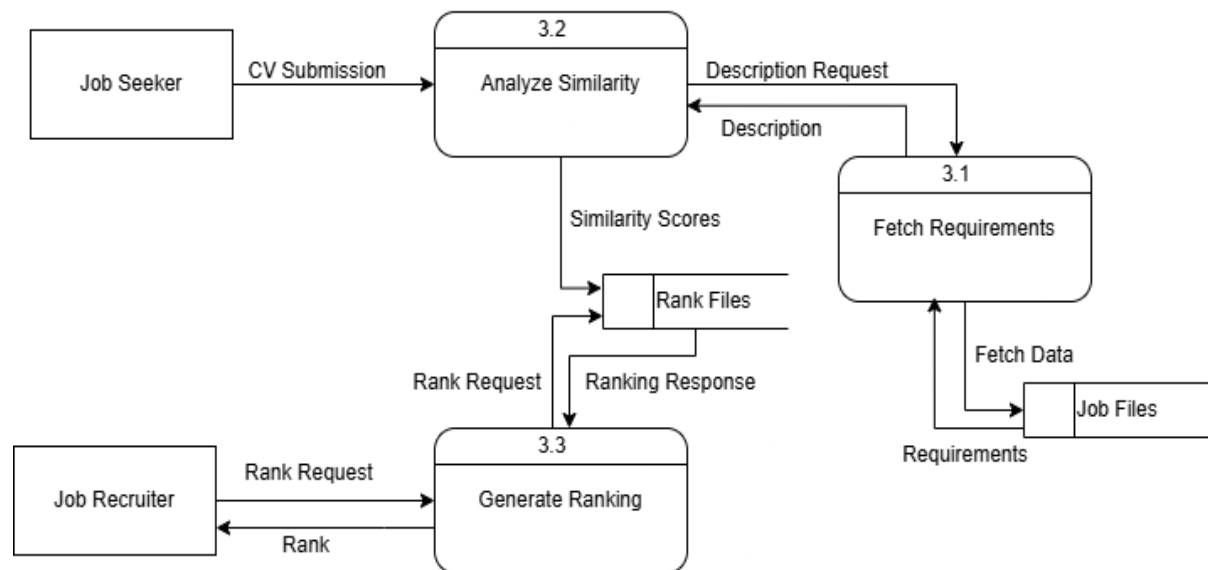


Figure 3.3.2.5: DFD Level 2 of Ranking System

The Job Seeker initiates the process by sending a test request to the Present Questions process, which responds by providing test questions. The Job Seeker answers the questions, and these answers are sent to the Analyze Response process for evaluation. The Analyze Response process generates processed results, which are passed to the Store Results process. This process stores the result data in the Big 5 Files database. The stored results are then displayed back to the Job Seeker, completing the flow of the system. This diagram captures the interactions between the Job Seeker, the processes, and the Big 5 Files database, showing the logical progression of data.

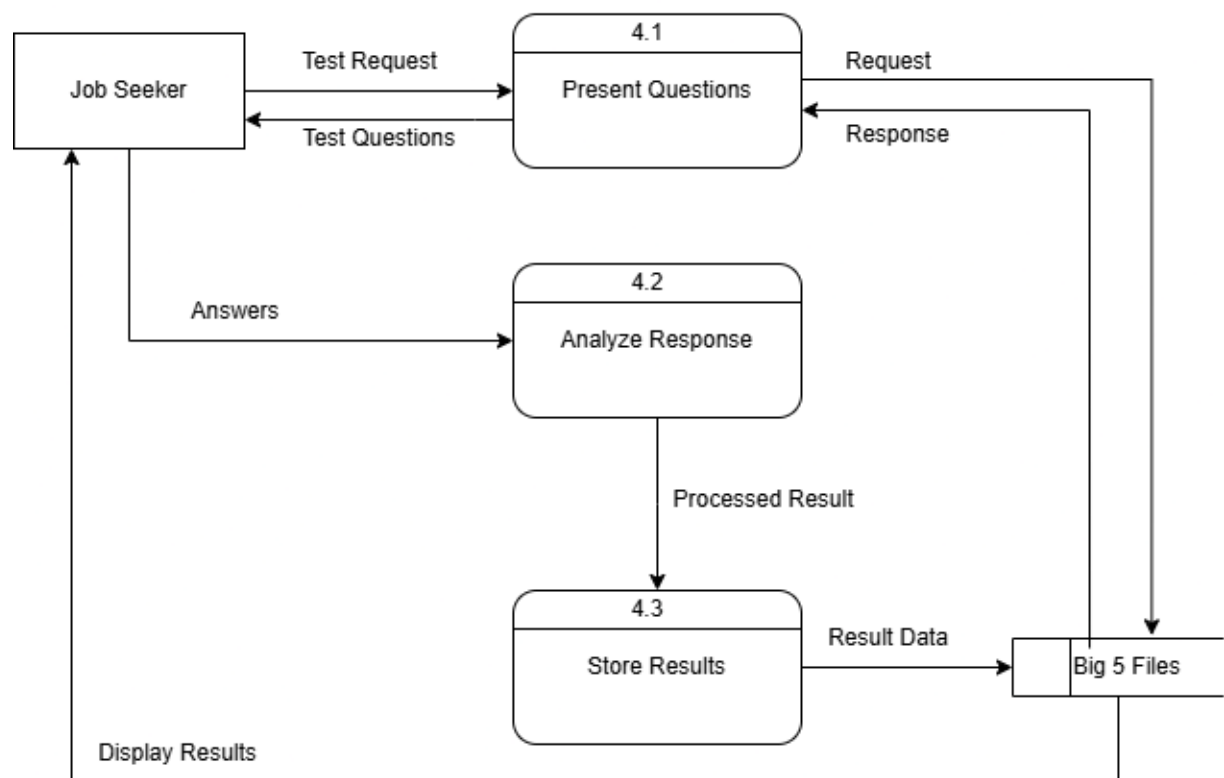


Figure 3.3.2.6: DFD Level 2 of Personality System

CHAPTER 4: SYSTEM DESIGN

4.1 DESIGN

This section provides a comprehensive overview of the system's architecture and its functional components. The design ensures the system meets both functional and non-functional requirements, offering efficiency, scalability, and user satisfaction.

4.1.1 SYSTEM FLOW DIAGRAM

The figure below represents the system flow, accessible by both employers (or admins) and candidates. Employers can upload job descriptions and, if necessary, aptitude questionnaires for vacancies, which are then published in the system. Candidates can view these vacancies, upload their CVs, take required tests, and complete a personality evaluation questionnaire. The system compares CVs with job descriptions, grading and ranking them based on job requirements and qualifications. Finally, it provides employers with a ranked list of candidates along with test results, streamlining the shortlisting process and saving time in conventional hiring.

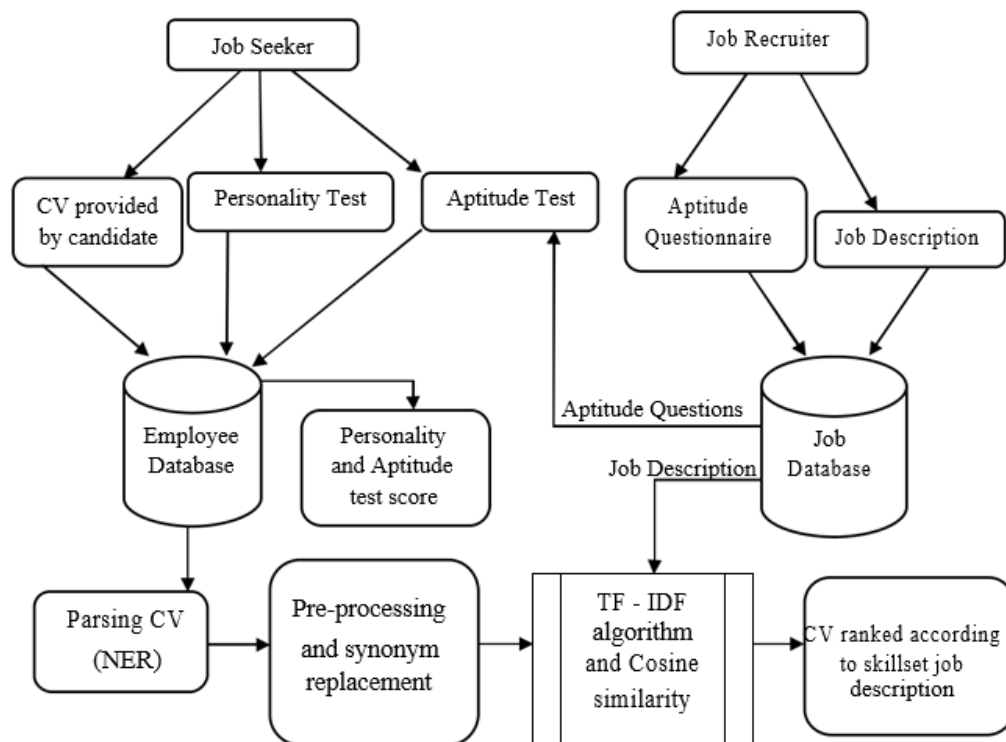


Figure 4.1.1.1: System Flow Diagram of Personality Evaluation & CV Analysis System

4.1.2 SCHEMA DIAGRAM

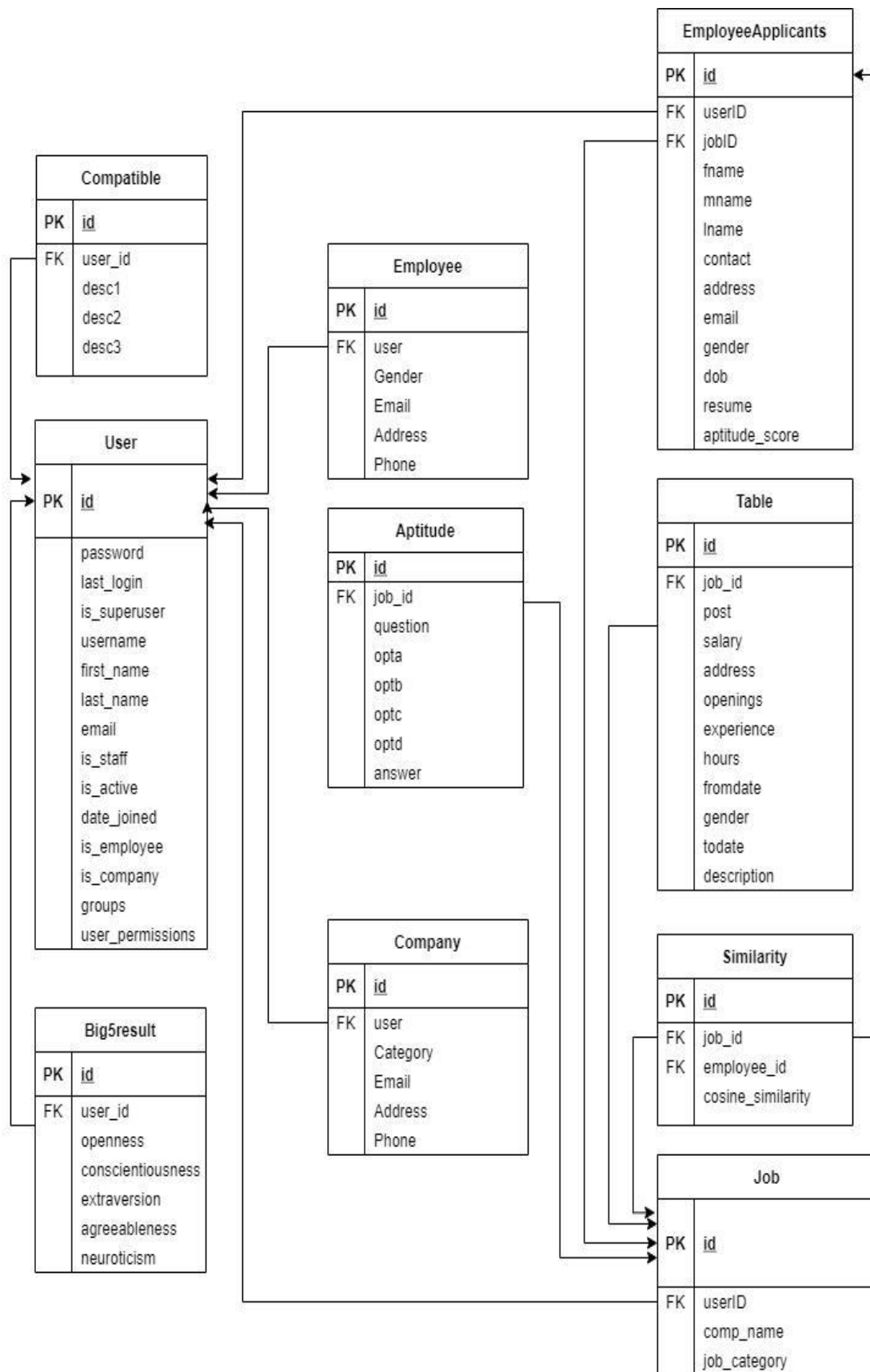
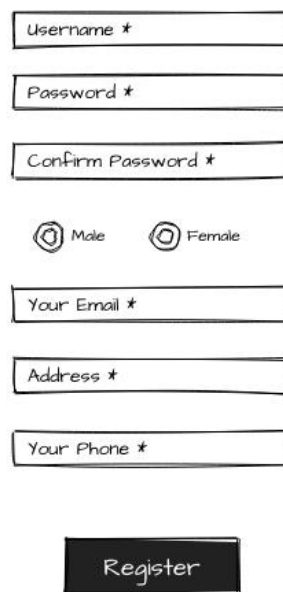


Figure 4.1.2.1: Schema Diagram of Personality Evaluation & CV Analysis System

The schema diagram illustrates the system's tables and their relationships. The User table stores general information, including login credentials, contact details, and user type, linking to Employee and Company tables to distinguish job seekers from employers. The Employee table contains specific details like gender, resume, and aptitude scores, while the Company table holds employer data such as category and contact details. The Job table represents job postings with details like job name, category, and employer, linking to additional job-specific data (e.g., salary, openings) in the Table table. The EmployeeApplicants table tracks applications, connecting employees to jobs, and the Aptitude table stores aptitude test data. The Big5result table records personality test results, and the Similarity table calculates cosine similarity between CVs and job descriptions to suggest matches. The Compatible table tracks compatibility between employees and potential roles.

4.1.3 FORM AND REPORT DESIGN

Apply as an Employee



A vertical form for employee registration. It consists of seven input fields and a submit button. The fields are: Username *, Password *, Confirm Password *, a gender selection section with radio buttons for Male and Female, Your Email *, Address *, and Your Phone *. The submit button is labeled 'Register'.

Username *

Password *

Confirm Password *

☐ Male ☐ Female

Your Email *

Address *

Your Phone *

Register

Figure 4.1.3.1: Employee Sign Up

Company Name *	Your Email *
Password *	Address *
Confirm Password *	Contact Number *
Choose Category ▼	
Register	

Figure 4.1.4.2: Company Sign Up

Your Username *
Your Password *
Login

Figure 4.1.5.3: Employee Login

Company Name *
Password *
Login

Figure 4.1.6.4: Company Login

First Name *	Middle Name *	Last Name *
Address *	Gender ▼	
Email *	Contact *	Date of Birth *

Apply

Figure 4.1.7.5: Job Application Form

Question

Option A

Option B

Option C

Option D

Correct Answer

Add Question

Finish

Figure 4.1.8.6: Aptitude Test Form

RECRUIT				
S.No	Name	Contact	Gender	Aptitude Score
1	Abc	4932.432.4	M	1
2	DeF	3432.432.4	F	5
3	GhI	1565.46899B	M	8

Ranking CV		
<p>1</p> <p>Abc</p> <p>4932.432.4</p> <p>M</p> <p>1</p>	<p>2</p> <p>DeF</p> <p>3432.432.4</p> <p>F</p> <p>5</p>	<p>3</p> <p>GhI</p> <p>1565.46899B</p> <p>M</p> <p>8</p>

Figure 4.1.9.7: Report Design

4.1.4 INTERFACE DESIGN

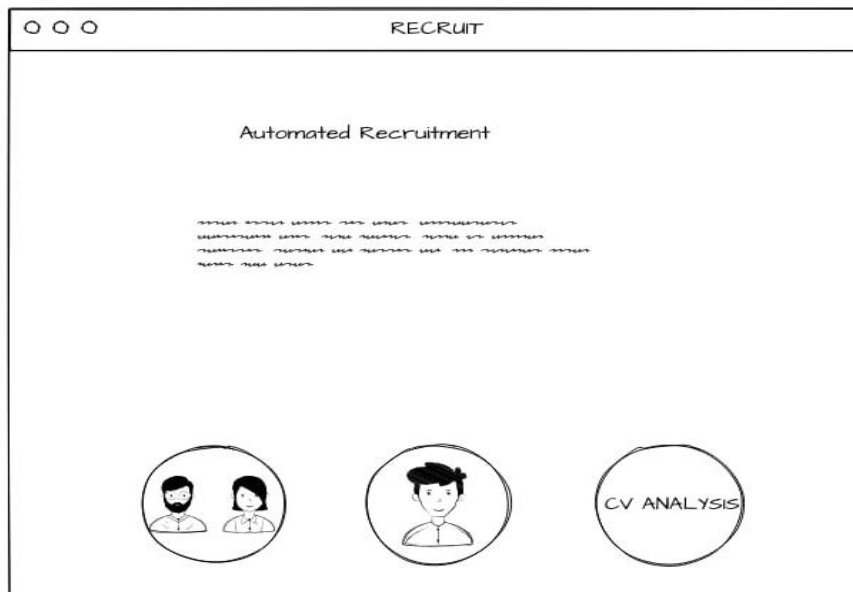


Figure 4.1.4.1: Home Page

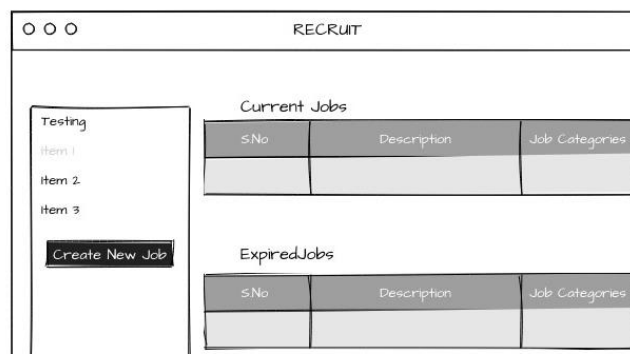


Figure 4.1.4.2: Company Dashboard

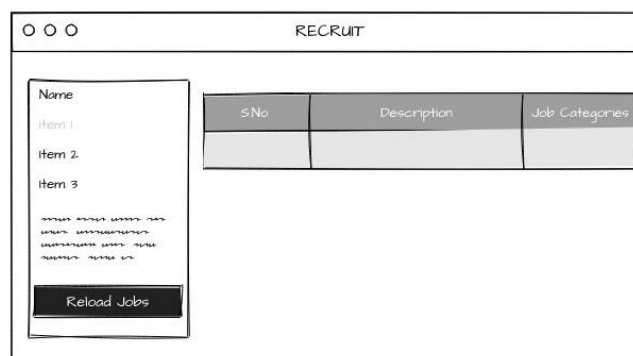


Figure 4.1.4.3: Employee Dashboard

4.2 ALGORITHM DETAILS

4.2.1 TF-IDF ALGORITHM

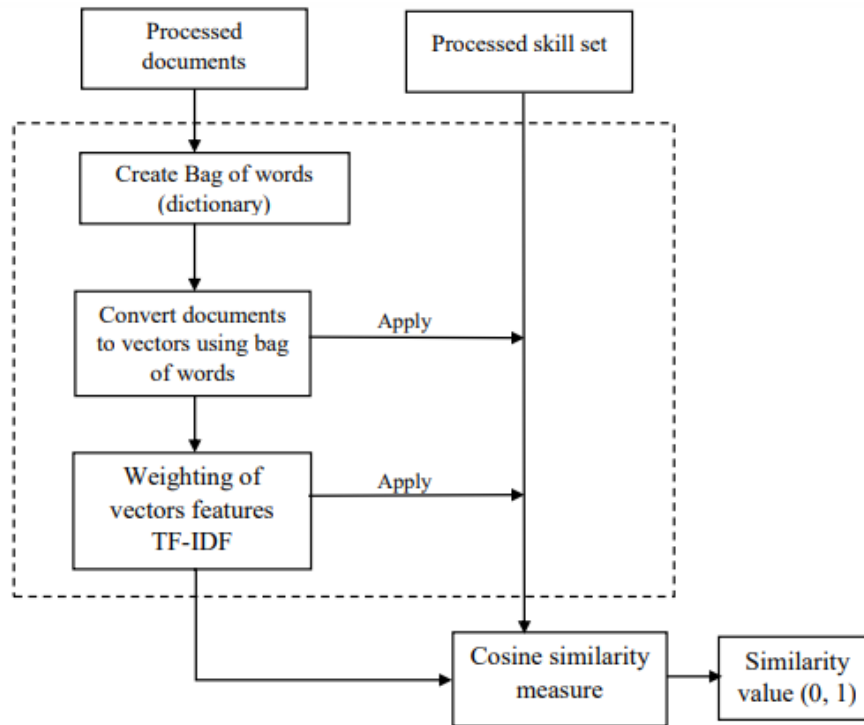


Figure 4.2.1.10: TF-IDF Algorithm

The TF-IDF weight can be calculated as:

Step 1:

$$\text{TF}(\text{'keyword'}) = \frac{\text{No. of times keyword appears in document}}{\text{Total no. of keywords in the document}} \quad \text{--- (i)}$$

The Term Frequency (TF) measures how often a keyword appears in a single document relative to the total number of keywords in that document. This gives a normalized value indicating the significance of the keyword within the document.

Step 2:

$$\text{IDF}(\text{'keyword'}) = \log \left(\frac{\text{Total no. of CV}}{\text{No. of document with term keyword}} \right) \quad \text{--- (ii)}$$

The Inverse Document Frequency (IDF) assesses the importance of a keyword by considering how unique it is across all documents in the corpus. Keywords that appear in fewer documents have higher IDF values, emphasizing their uniqueness.

Step 3:

$$\text{Weight} = \text{TF}(\text{'keyword'}) * \text{IDF}(\text{'keyword'}) \text{ ————— (iii)}$$

The TF-IDF weight combines the Term Frequency and Inverse Document Frequency to assign a final weight to the keyword. This weight represents the keyword's relevance, giving higher scores to keywords that are frequent in a specific document but rare across the corpus.

4.2.2 COSINE SIMILARITY

Cosine similarity is a metric used to determine how similar the documents are irrespective of their size.

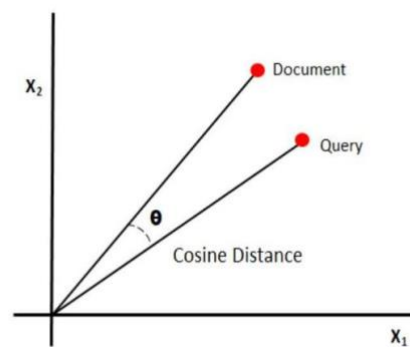


Figure 4.2.2.1: Vector plot of TF-IDF tokens

Cosine similarity can be achieved as:

$$\text{Similarity} = \frac{\text{Doc1} \cdot \text{Doc2}}{|\text{Doc1}| |\text{Doc2}|} \text{ ————— (iv)}$$

Where, Doc1 and Doc2 are the TF-IDF weight vectors of the documents.

Cosine similarity measures the similarity between two vectors of an inner product space. The greater the value of θ , the less the value of $\cos \theta$, thus the less the similarity between two documents.

CHAPTER 5: IMPLEMENTATION AND TESTING

5.1 IMPLEMENTATION

This section provides a comprehensive overview of the technologies, tools used for the completion of the system. The working mechanism as well as series of testing are also highlighted in this section.

5.1.2 TOOLS USED

1. **PostgreSQL:** It was used for fulfilling the database requirements of the system.
2. **HTML, CSS:** HTML and CSS allow for the creation of well-structured, visually appealing websites that can adapt to various screen sizes and devices, making them essential tools for front-end development.
3. **PyCharm:** PyCharm was used for coding, editing and debugging of the system.
4. **UDT (Universal Data Tool):** The Universal Data Tool is a web/desktop app for editing and annotating images, text, audio, documents and to view and edit any data defined in the extensible .udt.json and .udt.csv standard. It was used to manually annotate and label the CVs and descriptions and to convert them into json format.
5. **NLTK (Natural Language Tool Kit) library:** The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language. It was used for pre-processing functions such as tokenization, lemmatization and stop words removal.
6. **spaCy:** It was used for processing the annotated data by the use of NER (Named Entity Recognition).
7. **draw.io:** For system design and architectural diagrams, draw.io was employed.
8. **WordNet:** It was used to detect and replace synonyms to increase the effectiveness of TF-IDF algorithm.
9. **Figma:** It was used for UI/UX design.

5.1.2 IMPLEMENTATION DETAILS OF MODULES

1. CV Ranking System

The basic flow of the CV Ranking System is as given below:

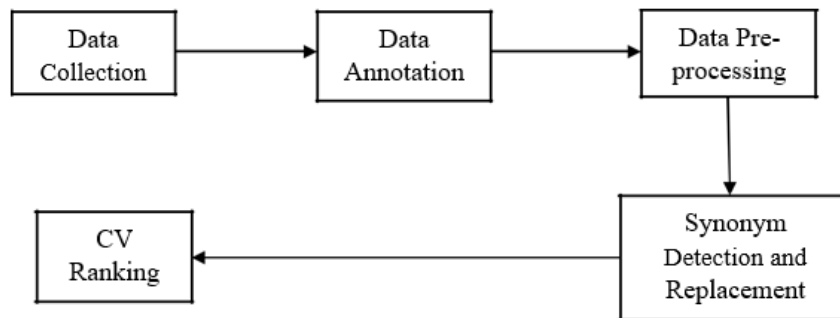


Figure 5.1.2.1: CV Ranking System Flow Diagram

Data Collection

We collected different CV in different format from various sources. We collected some from:

- Google Drive
- Kaggle Dataset

We also received some materials with the help of our supervisor. And, for job description, we collected our description in electronic format from websites like jobsnepal, careerbuilder and monster.

We collected standard questionnaire set for Big 5 Personality test from Big 5 Factor Markers and selected our standard 50 questionnaire from a GIT link.

Data Annotation

Firstly, CVs and Job descriptions are collected for the purpose of analysis and training. These are raw data with no pre-processing or labelling done.

The unannotated or raw data are manually annotated according to the designated labels for filtering of relevant information. This is done by the use of a data annotation tool UDT (Universal Data Tool). For analyzing our unstructured documents, entities are predefined and for annotating data we use these entities to label our documents.

The output of this process is a labelled data in JSON format including only necessary or relevant information.

JSON format example:

```
"samples": [  
  {  
    "_id": "s_wm6vkf08",  
    "document": "Priyanka Mehra. Certified Full Stack Web  
Developer. priyanka.mehra@gmail.com. (718)  
212-6466.  
linkedin.com/in/priyankamehra.....  
",  
    "annotation": {  
      "entities": [  
        {  
          "text": "specialized skills in Java and JavaScript",  
          "label": "Skills",  
          "start": 140,  
          "end": 181  
        },  
        {  
          "text": "Diverse experience utilizing Java tools in  
business, Web, and client server environments",  
          "label": "Experience",  
          "start": 216,  
          "end": 304  
        },  
      ]  
    }  
  }  
]
```

For the annotated data to be trained, it needs to be changed into a format accepted by the spaCy model i.e. is the pickle format which is also the output of this block.

spaCy format example:

```
[  
  ('Specialized skills in Java and JavaScript', {'entities': [(23, 26, 'Skills'),  
    (28, 37, 'Skills')]}])  
]
```

The final dataset in pickle format is then used for training with the help of NER (Named Entity Recognition) feature of spaCy to get a trained model with the ability to label and filter relevant information from additional CVs and descriptions.

Every “decision” made – for example, which part-of-speech tag to assign, or whether a word is a named entity – is a prediction based on the model’s current weight values. The weight values are estimated based on examples the model has seen during training. To train a model, we first need training data – examples of text, and the labels we want the model to predict. This could be a part-of-speech tag, a named entity or any other information.

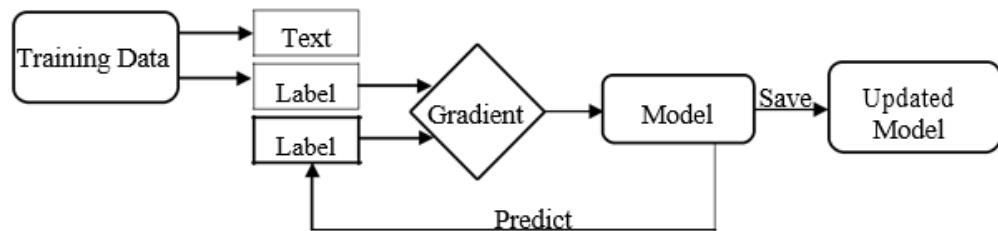


Figure 5.2.2.2: NER Training

Training is an iterative process in which the model’s predictions are compared against the reference annotations in order to estimate the gradient of the loss. The gradient of the loss is then used to calculate the gradient of the weights through backpropagation. The gradients indicate how the weight values should be changed so that the model’s predictions become more similar to the reference labels over time.

When training a model, we don’t just want it to memorize our examples – we want it to come up with a theory that can be generalized across unseen data.

After the model is trained, we are able to get labelled outputs to any CV or description fed into it.

Data 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. Text preprocessing steps are widely used for dimensionality reduction. In the vector space model, each word/term is an axis/dimension. The number of unique words means the number of dimensions.

Tokenization: Tokenization is essentially splitting a phrase, sentence, paragraph, or an entire text document into smaller units, such as individual words or terms. Each of these smaller units are called tokens. The tokens could be words, number or punctuation marks.

Input: "SLC from Nobel Academy"

Output: ['SLC', 'from', 'Nobel', 'Academy']

Lower casing: Converting a word to lower case (NLP -> nlp).

Output: ['slc', 'from', 'nobel', 'academy']

Stop words removal: Stop words are very commonly used words (a, an, the, etc.) in the documents. These words do not really signify any importance as they do not help in distinguishing two documents.

Input: "Machine learning is cool"

Output: ['Machine', 'Learning', 'cool', '!']

Explanation: Stop word 'is' has been removed

Removing punctuation: In python, we remove punctuation characters using `translate()` method. This method makes a copy of a string with a specific set of values substituted.

`text.translate(str.maketrans("", "", string.punctuation))`

Lemmatization: Lemmatization reduces the words to a word existing in the language. For lemmatization to resolve a word to its lemma, part of speech of the word is required. This helps in transforming the word into a proper root form. (change, changing, changes) -> change

Synonym Detection and Replacement

As the TF-IDF is unable to match words with the same meaning and count them as one, the accuracy of the result is greatly affected. This also creates a vector larger than required as synonyms are treated as different words. So, synonyms should be detected and replaced beforehand so this problem could be avoided and accuracy could be increased. For this, WordNet is used.

The WordNet is a part of Python's Natural Language Toolkit. It is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. WordNet's structure makes it a useful tool for computational linguistics and natural language processing. WordNet superficially resembles a thesaurus, in that it groups words together based on their meanings.

```
In [6]: modified_arr=[['poor', 'boat'], ['good', 'misfortunate', 'fruit'], ['bad', 'best']]

In [7]: skip=[]
        for d in modified_arr:
            for i in range(len(d)):
                synonyms=[]
                for syn in wn.synsets(d[i]):
                    for l in syn.lemmas():
                        synonyms.append(l.name())

                for doc in modified_arr:
                    for j in range(len(doc)):
                        if doc[j] not in skip:
                            if doc[j] in synonyms:
                                if doc[j]!=d[i]:
                                    doc[j]=d[i]
                                    skip.append(doc[j])

In [8]: print(modified_arr)

[['poor', 'boat'], ['best', 'poor', 'fruit'], ['bad', 'best']]
```

Figure 5.3.2.3: WordNet Synonym Replacement Example

As can be seen in the figure, the word list of each document is scanned to find the similar words and replace them. The word good and best are recognized as synonyms so are replaced by a single and common word i.e. best. The replaced words are then skipped during the scan so as to avoid double replacement.

Thus, by the use of WordNet Synonyms of the words were detected and replaced by a common and single word to reduce the vector size of the documents, solve the problem of not being able to detect and distinguish synonyms and improve accuracy of the TF-IDF result.

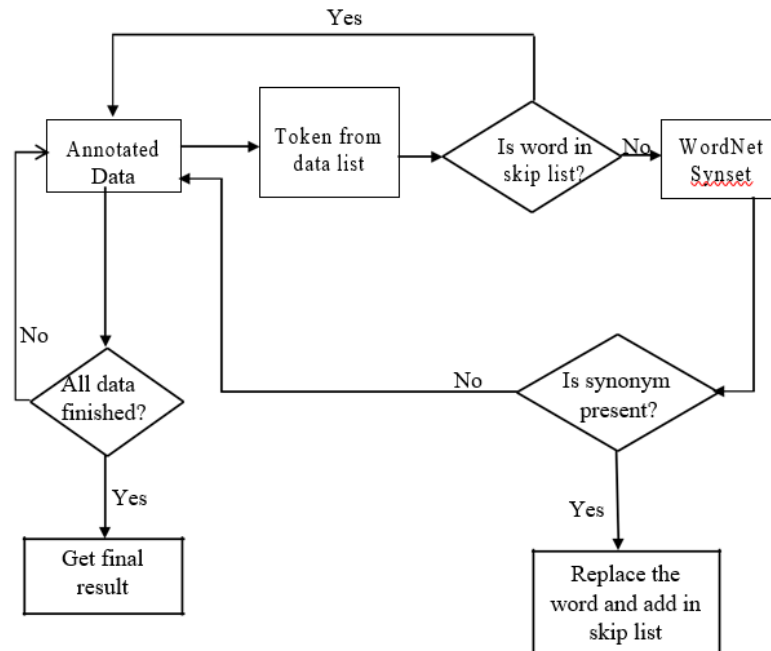


Figure 5.4.2.4: Synonym Replacement using WordNet

CV Ranking

After applying WordNet to detect and replace synonyms we can move on to calculating the TF-IDF weight of the documents and finally achieving the similarity score using cosine similarity. TF-IDF, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in searches of information retrieval, text mining, and user modeling. Cosine similarity is a metric used to determine how similar the documents are irrespective of their size. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. In this context, the two vectors I am talking about are arrays containing the word counts of two documents.

Functions:

```
def extract_text_from_pdf(file_path):
    with open(file_path, 'rb') as file:
        reader = PyPDF2.PdfReader(file)
        text = ""
        for page in reader.pages:
            text += page.extract_text()
    return text

def train(fnames):
    docs = []
    for i in range(len(fnames)):
        pdf_path = './' + fnames[i]
        print(f"Processing file: {pdf_path}")
        texts = extract_text_from_pdf(pdf_path)
        tx = " ".join(texts.split('\n'))
        docs.append(tx)
    print(docs[len(docs) - 1])
    nlp_model_annotation =
    spacy.load('./Dashboard/nlpdesc_model', exclude=['tokenizer'])
    nlp_CV_annotation =
    spacy.load('./Dashboard/nlpdesc_CV_model', exclude=['tokenizer'])
    documents = []
    for index, d in enumerate(docs):
        if index == len(docs) - 1:
            document = nlp_model_annotation(d)
        else:
            document = nlp_CV_annotation(d)
        text = "".join(ent.text for ent in document.ents)
        if len(text) < 100:
            documents.append(d)
        else:
            documents.append(text)
    wordnet = WordNetLemmatizer()
    stop_words = set(stopwords.words('english'))
    table = str.maketrans('', '', string.punctuation)
    modified_arr = [[wordnet.lemmatize(i.lower()) for i in
    tokenize(d.translate(table)) if i.lower() not in stop_words] for
    d in documents]
    skip = []
    for d in modified_arr:
        for i in range(len(d)):
            synonyms = []
            for syn in wn.synsets(d[i]):
                for l in syn.lemmas():
                    synonyms.append(l.name())
            for doc in modified_arr:
                for j in range(len(doc)):
```

```

        if doc[j] not in skip:
            if doc[j] != d[i] and doc[j] in synonyms:
                doc[j] = d[i]
            if doc[j] not in skip:
                skip.append(doc[j])
    modified_doc = [' '.join(i) for i in modified_arr]
    tf_idf = TfidfVectorizer().fit_transform(modified_doc)
    similarity = []
    length = len(documents) - 1
    for i in range(length + 1):
        cosine = cosine_similarity(tf_idf[length], tf_idf[i])
        similarity.append(cosine)
        print(cosine)
    return similarity

def train_desc(fnames):
    docs = []
    for i in range(len(fnames)):
        pdf_path = './' + fnames[i]
        print(f"Processing file: {pdf_path}")
        texts = extract_text_from_pdf(pdf_path)
        tx = " ".join(texts.split('\n'))
        docs.append(tx)

    nlp_model_annotation =
    spacy.load('./Dashboard/nlpdesc_model')
    nlp_CV_annotation =
    spacy.load('./Dashboard/nlpdesc_CV_model')
    documents = []
    for index, d in enumerate(docs):
        if index == len(docs) - 1:
            document = nlp_CV_annotation(d)
        else:
            document = nlp_model_annotation(d)
        text = "".join(ent.text for ent in document.ents)
        if len(text) < 100:
            documents.append(d)
        else:
            documents.append(text)

    wordnet = WordNetLemmatizer()
    stop_words = set(stopwords.words('english'))
    table = str.maketrans('', '', string.punctuation)
    modified_arr = [[wordnet.lemmatize(i.lower()) for i in
    tokenize(d.translate(table)) if i.lower() not in stop_words] for
    d in documents]
    skip = []
    for d in modified_arr:

```

```

        for i in range(len(d)):
            synonyms = []
            for syn in wn.synsets(d[i]):
                for l in syn.lemmas():
                    synonyms.append(l.name())
            for doc in modified_arr:
                for j in range(len(doc)):
                    if doc[j] not in skip:
                        if doc[j] != d[i] and doc[j] in synonyms:
                            doc[j] = d[i]
                            if doc[j] not in skip:
                                skip.append(doc[j])
            modified_doc = [' '.join(i) for i in modified_arr]
            tf_idf = TfidfVectorizer().fit_transform(modified_doc)
            similarity = []
            length = len(documents) - 1
            for i in range(length + 1):
                cosine = cosine_similarity(tf_idf[length], tf_idf[i])
                similarity.append(cosine)
                print(cosine)
            return similarity

def change_to_spacy(JSON_FILEPATH):
    try:
        training_data = []
        lines = []
        file = open(JSON_FILEPATH, 'r', encoding='UTF8')
        lines = file.readlines()
        for line in lines:
            data = json.loads(line)
            entities = []
            sample_no = len(data['samples'])
            for item in range(sample_no):
                diction = data['samples'][item]
                text = diction['document']
                notes = diction['annotation']
                entity_no = len(notes['entities'])
                for i in range(entity_no):
                    note = notes['entities'][i]
                    labels = note['label']
                    start = note['start']
                    end = note['end']
                    if not isinstance(labels, list):
                        labels = [labels]
                    for label in labels:
                        entities.append((start, end, label))
                training_data.append((text, {"entities" :
entities}))

```

```

        return training_data
    except Exception as e:
        logging.exception("Unable to process item " + JSON_FILEPATH +
"\n" + "error = " + str(e))
        return None

def train_model(train_data):
    if 'ner' not in nlp.pipe_names:
        ner = nlp.create_pipe('ner')
        nlp.add_pipe(ner, last = True)
    for _, annotation in train_data:
        for ent in annotation['entities']:
            ner.add_label(ent[2])
    other_pipes = [pipe for pipe in nlp.pipe_names if pipe != 'ner']
    # only train NER
    with nlp.disable_pipes(*other_pipes):
        optimizer = nlp.begin_training()
        for itn in range(10):
            print("starting iteration " + str(itn))
            random.shuffle(train_data)
            losses = {}
            for text, annotations in train_data:
                try:
                    nlp.update(
                        [text], # batch of texts
                        [annotations], # batch of annotations
                        drop=0.2,
                        sgd = optimizer,# dropout - make it harder to
memorise data
                        losses=losses)
                except Exception as e:
                    pass
            print(losses)
    train_model(train_data)
    nlp.to_disk('nlp_model_annotation')
    nlp_model_annotation = spacy.load('nlp_model_annotation')

```

2. Personality Test System

For personality evaluation we used BIG 5 or OCEAN (Openness, Conscientiousness, Extraversion, Agreeableness, and Neuroticism) personality test. The test was performed by providing a standard set of 50 questions (10 questions for each trait), both positive and negative keyed questions, to the job seekers to which the answers ranged from strongly agree (5) to strongly disagree (1).

Negative keyed questions

For example, the factor Extraversion describes someone who is outgoing, energetic, talkative, and enjoys human interaction. The first Extraversion item [EXT1] is "I am the life of the party." a positively-keyed item; whereas the second item [EXT2] is "I don't talk a lot." a negatively-keyed item.

Reverse Coding

The value for the answers were recorded and reverse-coded by subtracting 6 from the value of answers for all negative keyed questions. This results in a dataset where the item values all have a common direction and interpretation (i.e., a higher value corresponds with more of that trait).

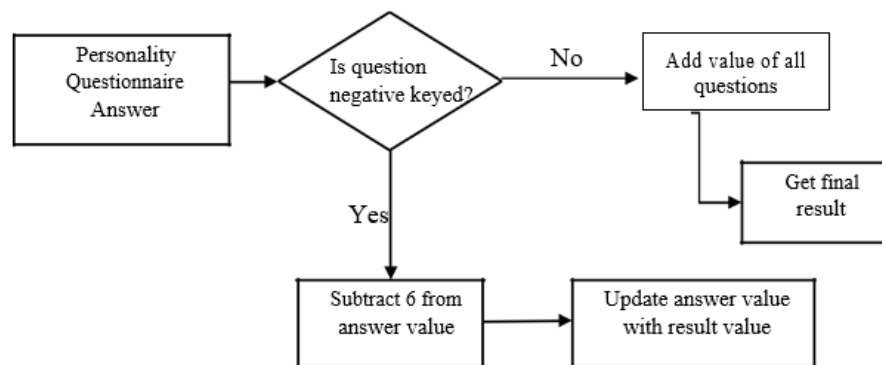


Figure 5.5.2.5: Big 5 Personality Test

Finally, the resultant values were added together and percentage value was calculated to determine what traits exist in what amount in the individuals.

```
def handle_personality_test(self, answers):
    answer_dict = {}
    for question, answer in answers.items():
        if question in self.negative_keyed:
            answer = 6 - answer
        key = self.questions_key[question]
        answer_dict[key] = answer
    score_dict = {'O_score': 0, 'C_score': 0, 'E_score': 0,
'A_score': 0, 'N_score': 0}
    for trait_key, answer in answer_dict.items():
        if 'O' in trait_key:
            score_dict['O_score'] += answer
        if 'C' in trait_key:
            score_dict['C_score'] += answer
        if 'E' in trait_key:
```

```

        score_dict['E_score'] += answer
    if 'A' in trait_key:
        score_dict['A_score'] += answer
    if 'N' in trait_key:
        score_dict['N_score'] += answer
for key, score in score_dict.items():
    score_dict[key] = score / 10
perc_dict = {}
for key, score in score_dict.items():
    perc = (score / 5) * 100
    perc_dict[key] = round(perc)
return (perc_dict)

```

5.2 TESTING

The system underwent a rigorous testing phase to ensure the application's functionality, accuracy and user experience aligned with the objectives of the system. The testing process was integrated into the Agile development methodology, allowing for continuous refinement and validation of features.

5.2.1 UNIT TESTING

Unit testing for the system focused on validating individual components and functionalities to ensure the accuracy and reliability of the application. Below is a representative table summarizing selected unit test cases:

S.N.	Test case scenario	Input / Test data	Expected Outcomes	Actual Outcome	Result
1.	User Registration: Enter user details	Username: Alok Contact: 9804706675 Password: alok Email: alokdugar4@gmail.com	The user details should be added into the database	As Expected	Passed
2.	User Login: Invalid Username	Username: alok Password: alok	Display error message	As Expected	Passed
3.	User Login: Invalid Password	Username: ALok Password: alok	Display error message	As Expected	Passed

4.	User Login: Valid credentials	Username: Alok Password: alok	User is logged in to the system	As Expected	Passed
5.	Personality Test: Take personality test	Randomly select the choices	The results should be calculated and displayed	As Expected	Passed
6.	Job creation: Create a job from company account	Fill the form to create a job vacancy	The job is created and displayed	As Expected	Passed
7.	Job Apply: Apply for a job from employee account	Fill the form to apply	The application should be sent and displayed in company dashboard	As Expected	Passed
8.	File Upload: Upload file for CV and requirements	Choose the file to upload	The file should be uploaded and viewable	As Expected	Passed

Table 5.2.1.1: Unit Testing

5.2.2 INTEGRATION TESTING

Integration testing focused on verifying the interaction between various modules and ensuring they function cohesively. Below is a representative table summarizing selected integration test cases:

S.N.	Test case scenario	Input / Test data	Expected Outcomes	Actual Outcome	Result
1.	User Registration and Login	Register a user and then log in with the same credentials.	The user should register successfully and log in without errors.	As Expected	Passed
2.	Job Creation and Viewing	Create a job posting and verify its visibility in listings.	The job should appear in the job listings visible to employees.	As Expected	Passed
3.	Job Application Workflow	Apply for a job and check the company's dashboard.	The job application should be visible in the company's dashboard.	As Expected	Passed
4.	Personality Test and Result Display	Complete the personality test and check the profile.	The results should be calculated, saved, and displayed in the user's profile.	As Expected	Passed
5.	Resume Matching and Ranking	Upload a resume and job description for matching.	Compatibility rankings should be calculated and displayed in the company dashboard.	As Expected	Passed

6.	File Upload Across Modules	Upload CVs, job descriptions, and aptitude test files.	Files should upload successfully and be accessible in respective modules.	As Expected	Passed
----	-------------------------------	--	---	-------------	--------

Table 5.2.2.1: Integration Testing

5.2.3 SYSTEM TESTING

System testing was conducted on the complete integrated system to evaluate the system's compliance with its specified requirements. System testing took, as its input, all of the integrated components that have passed integration testing. The system test was a success as the system performance as a whole was as the expected results.

S.N.	Test Case Scenario	Input / Test Data	Expected Outcomes	Actual Outcome	Result
1	User Account Workflow	Register a user, log in with valid credentials, and view the dashboard.	The user should successfully register, log in, and access the dashboard displaying personalized content.	As Expected	Passed
2	Personality Test Workflow	Complete the personality test and view the results in the profile section.	Results are calculated accurately, saved to the database, and displayed in the profile section.	As Expected	Passed
3	Job Creation and Listing	Log in as a company, create a job, and check if the job listing is displayed.	The created job should appear on the job listings page and be accessible to employees for application.	As Expected	Passed

4	Job Application Workflow	Log in as an employee, apply for a job, and verify if the company receives the application.	The application is sent successfully and visible in the company's dashboard.	As Expected	Passed
5	Resume Matching Workflow	Upload a resume and job description, and check for compatibility rankings in the company dashboard.	Compatibility rankings should be calculated, stored in the database, and displayed in descending order.	As Expected	Passed
6	File Upload Verification	Upload a CV or job description file during job creation or application processes.	Files are uploaded successfully, stored securely, and accessible via download links or previews.	As Expected	Passed
7	Error Handling	Attempt invalid operations such as logging in with wrong credentials or accessing restricted pages.	Error messages are displayed appropriately, and the user is redirected to the correct pages.	As Expected	Passed

Table 5.2.3.1: System Testing

5.3 RESULT ANALYSIS

The accuracy was measured for the proper annotation of data into its various labels for both the CVs and descriptions. For this, we compared the manually annotated and model annotated output for the same set of data to test how accurately the data was annotated. The result of comparison and testing can be seen in the tables below:

Entity	Precision	Recall	F-Score	Accuracy
Designation	0.9976	0.9953	0.9960	99.60%
Responsibility	0.9452	0.9527	0.9476	95.27%
Qualification	1.0	0.9677	0.9836	96.77%

Table 5.3.1: Accuracy for Job Description Annotation

Entity	Precision	Recall	F-Score	Accuracy
Skills	0.8138	0.7971	0.7327	79.71%
Experience	0.8073	0.7626	0.7814	76.26%
Qualification	0.9067	0.9164	0.8957	91.64%
Project	0.6663	0.7374	0.6870	73.74%

Table 5.3.2: Accuracy for CV Annotation

CHAPTER 6: CONCLUSION AND FUTURE RECOMMENDATIONS

6.1 CONCLUSION

Manually sorting through hundreds of CVs during the recruitment process is a laborious task that requires great effort and time. This project aimed to develop a much better alternative in the form of a CV grading system based on the similarity between the CVs and job description. This result was obtained by implementing different algorithms and methods which consequently generated the desired output. So, a platform for both job seekers and recruiters was created with added functionality of CV analysis and grading. Enormous knowledge was gained through the project. Discovering and solving various predicted and unpredicted problems as a team, we learnt a lot on the subject of NLP processing and also learned the use and implementation of various other tools used while development of project. We understood the importance of background research as well. Thus, we hope that the system developed will certainly assist in the field of recruitment by saving both time and effort spent on sorting the CVs and provide a new advanced method for better performance.

6.2 FUTURE RECOMMENDATIONS

Due to time constraint, many features couldn't be incorporated in the project. The system can be upgraded in many aspects such as:

- Providing a feature for comparison of CVs so as to modify and improve own CV for better chances at landing the job.
- Feature of automatically detecting all the required information by just uploading the CV with no need of entering other additional information.

REFERENCES

- [1] J. Herrity, "indeed career guide," 24 November 2020. [Online]. Available: <https://www.indeed.com/career-advice/resumes-cover-letters/what-is-a-cv>. [Accessed 5 August 2024].
- [2] E. Mellett, "Wikijob," 16 October 2024. [Online]. Available: <https://www.wikijob.co.uk/content/application-advice/job-applications/what-job-description>. [Accessed 2024].
- [3] "daxtra," [Online]. Available: www.daxtra.com/2016/10/18/what-is-cvresume-parsing/. [Accessed 2024].
- [4] "Resume Professional Writers," [Online]. Available: <https://www.resumeprofessionalwriters.com/resume-vs-job-description-comparison/>. [Accessed 2024].
- [5] "Resume Professional Writers," [Online]. Available: <https://www.resumeprofessionalwriters.com/>. [Accessed 2024].
- [6] C. Ratcliff, "Search Engine Watch," 21 October 2015. [Online]. Available: <https://www.searchenginewatch.com/2015/10/21/what-is-semantic-search-and-why-does-it-matter/>. [Accessed 2024].
- [7] "Resume Hacking," [Online]. Available: www.resumehacking.com/ready-for-automated-resume-screening. [Accessed 2024].
- [8] P. Nelson, "Search Technologies," [Online]. Available: <https://www.searchtechnologies.com/blog/natural-language-processing-techniques>. [Accessed 2024].

APPENDICES

The screenshot shows a registration page with a teal header and a light gray registration form. On the left, there is a rocket icon, the text 'Welcome', and a blue 'Login' button. The form is titled 'Apply as a Employee' and has two tabs: 'Employee' (selected) and 'Company'. The form contains the following fields: a dropdown menu with 'employee' selected, a 'Your Email *' field, a password field with '*****', an 'Address *' field, a 'Confirm Password *' field, and a 'Your Phone *' field. At the bottom of the form, there are radio buttons for 'Male' (selected) and 'Female', and a green 'Register' button.

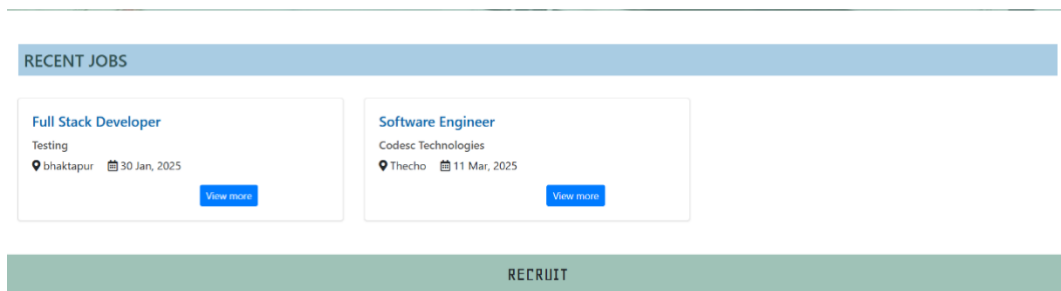
Signup Page

The screenshot shows a login page with two panels. The left panel is titled 'Employee Login' and has a white background. It contains a dropdown menu with 'employee' selected, a password field with '*****', and a green 'Login' button. The right panel is titled 'Company Login' and has a teal background. It contains a dropdown menu with 'employee' selected, a password field with '*****', and a white 'Login' button. At the bottom right of the page, there is a teal 'SignUp' button.

Login Page



Home Page



Recent Jobs

RECRUIT HOME JOBS JOB CATEGORY ABOUT PERSONALITY TEST DASHBOARD Search by Job Title logged in as EMPLOYEE Logout

Job Application

Apply for job here

First Name	Middle Name	Last Name
<input type="text"/>	<input type="text"/>	<input type="text"/>
Address		Gender
<input type="text"/>		Choose...
Email	Contact	Date of Birth
<input type="text"/>	<input type="text"/>	<input type="text" value="mm/dd/yyyy"/>
Upload CV		
<input type="button" value="Choose File"/> No file chosen		
<input type="button" value="Apply"/>		

Forms

RECRUIT
HOME
JOBS
JOB CATEGORY
ABOUT
PERSONALITY TEST
DASHBOARD

Search by Job Title

logged in as THE VIEW
Logout

Aptitude questions

Insert aptitude questions here

Question

What is highest mountain?

Option A

Everest

Option B

Dhaulagiri

Option C

Makalu

Option D

Lhotse

Correct Answer

Option A

Add Question
Finish

Activate Windows
Go to Settings to activate Windows.

Aptitude Test Form

RECRUIT
HOME
JOBS
JOB CATEGORY
ABOUT
PERSONALITY TEST
DASHBOARD

Search by Job Title

logged in as DRISTI
Logout

Aptitude questions

Answer the aptitude questions

1. What is capital of Nepal?

☐ Pokhara
☒ Kathmandu
☐ Hetauda
☐ Lumbini

Submit

Aptitude Test

RECRUIT
HOME
JOB CATEGORY
DASHBOARD
ABOUT

Search by Job Title

logged in as TESTING
Logout

TESTING

Contact: 9860923051
Address: Sanogaucharan
Email: alokdugar4@gmail.com

Create New Job

Current Jobs

S.No.	Designation	Job Category	Applicants	Deadline Status	Job Details	View Applicants	Update Job	Delete Job
1	Software Engineer	IT	2	25 Nov, 2024	View	View		
2	Full Stack Developer	IT	2	25 Nov, 2024	View	View		

Expired Jobs

S.No.	Designation	Job Category	Applicants	Deadline Status	Job Details	View Applicants	Delete Job
1	Software Engineer	IT	2	25 Nov, 2024	View	View	Delete
2	Full Stack Developer	IT	2	25 Nov, 2024	View	View	Delete

Company Dashboard

RECRUIT
HOME
JOB CATEGORY
PERSONALITY TEST
DASHBOARD
ABOUT

logged in as SHASHANK
Logout

SHASHANK

CONTACT : 9860923051

EMAIL : shashank@gmail.com

ADDRESS : Sanogaucharan

Personality

openness: 64.0%
conscientiousness: 66.0%
extraversion: 69.0%
agreeableness: 55.0%
neuroticism: 30.0%

MOST COMPATIBLE JOBS:

Full stack developer
Product Manager
Assistant Manager

Reload Compatible Jobs

S.No.	Designation	Category	Requirements	Your CV	Edit Application	Delete Application
1	Computer Lecturer	T	View	View	Expired	Delete
2	Full stack developer	IT	View	View	Expired	Delete

Employee Dashboard

RECRUIT
HOME
JOBS
JOB CATEGORY
ABOUT
PERSONALITY TEST
DASHBOARD

logged in as COMPANY
Logout

-- (Strongly disagree), - (Disagree), +/- (Neutral), + (Agree), ++ (Strongly Agree)

S. No.	Personality Test Questions	--	-	+/-	+	++
1	I am the life of the party.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
2	I don't talk a lot.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
3	I feel comfortable around people.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
4	I keep in the background.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
5	I start conversations.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
6	I have little to say.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
7	I talk to a lot of different people at parties.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
8	I don't like to draw attention to myself.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Personality Test

RECRUIT

HOME

JOBS

JOB CATEGORY

ABOUT

PERSONALITY TEST

DASHBOARD

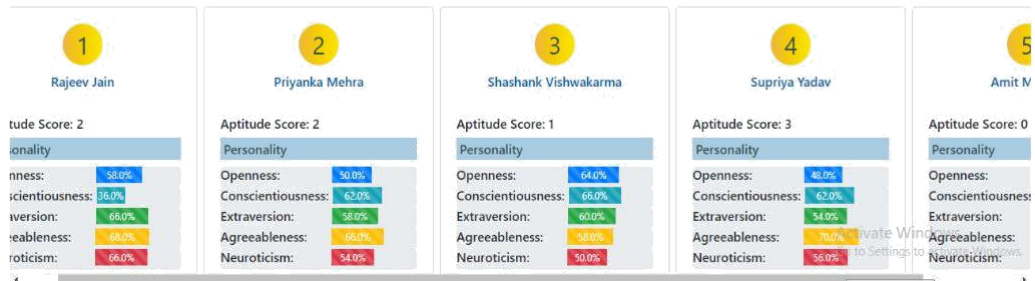
Search by Job Title

logged in as THE VIEW

Logout

2	Rajeev Jain	9843765778	rajeev@gmail.com	Mumbai	Male	08 Sep, 1950	0	View Resume
3	Amit Musale	9843765778	amit@gmail.com	Lazimpat	Male	08 Sep, 1850	0	View Resume
4	Supriya Yadav	9860923051	supriya@gmail.com	Sanogaucharan	Female	08 Sep, 1998	3	View Resume
5	Shashank Vishwakarma	9804706675	shashank@gmail.com	Lalitpur	Male	08 Sep, 1990	1	View Resume

CV RANKINGS



CV Ranking