

The islamic University of Gaza

Faculty of IT



الجامعة الإسلامية بغزة

كلية تكنولوجيا المعلومات

AI Resume Screening Tool

مشروع مراجعة وتقييم السير الذاتية باستخدام الذكاء الاصطناعي

By

Aya ABUSISI 220200290

Marah HarounJouda 220201196

Nagham Abu Warda 220200862

Supervised by

Dr. Motaz Saad

**A graduation project report submitted in partial
fulfilment of the requirements for the degree of
Bachelor of Information Technology**

April/2025

Abstract

Problem: In our project, we address the issue of the long tiresome hiring process that most companies have to go through, the frequent biases that arise in these hires, and the time and resources that are often wasted in the process. This is where our tool comes in to handle all that while also saving precious time and money.

Objectives and aims: Our goal is to match the best candidate for a job based on the job description and candidate resumes provided by the company. So, we tailored our tool to streamline candidates by identifying their key features and degree of match to the provided job description, sifting through candidates' analysed key features, then providing the top candidates, which are employed to conclude the best candidate. Our project also provides an open-source, free, and scalable solution with the most accuracy and efficiency possible.

Methodology: We went with the agile methodology basically for its flexibility in the testing and alteration phases, collaboration for team members' continuous component integration, and many more crucial features. It also satisfied our need for planning, daily stand-ups, and iterative development and adjustment.

Results: Our tool has the ability to process job text and resume files and additional criteria specified by the user, then extract the key features from each resume, match them against the job text, and calculate their similarity scores with the job text and filters them to return the resumes above a specific threshold value which then go through shortlisting to return the top candidates with a detailed evaluation of each one's strengths and weaknesses, and finally concludes the best resume justified which makes our objective specifics fairly satisfied and then some. Our tool performs its analysis based on similarity scores calculated based on embeddings generated by our system which increase the semantic meaning of the resumes and provide a better breakdown of each resume in the screening process; all these features have been thoroughly tried and tested as shown in the testing chapter.

Conclusions & Recommendations: Our project shows just how much this tool is needed demonstrating its many incredible features, and its capability for further integration and build up to become even more irreplaceable for individuals and companies of every size.

ملخص الدراسة

المشكلة: في مشروعنا، نتناول مشكلة عملية التوظيف الطويلة والمُرهقة التي تمر بها معظم الشركات، والتحديات المتكررة التي تنشأ خلال هذه العمليات، والوقت والموارد التي غالباً ما تُهدر في هذه العملية. وهنا يأتي دور أدواتنا للتعامل مع كل هذه التحديات، مع توفير الوقت والمال الثمينين.

الأهداف والغايات: هدفنا هو مطابقة أفضل مرشح للوظيفة بناءً على وصف الوظيفة والسير الذاتية المقدمة من قبل الشركة. لذلك، قمنا بتصميم أدواتنا لتبسيط عملية تصفية المرشحين من خلال تحديد ميزاتهم الرئيسية ومدى تطابق سيرهم الذاتية مع وصف الوظيفة المقدم، ثم فرز وتحليل هذه الميزات، وأخيراً تقديم أفضل المرشحين الذين يتم اختيارهم لاستخلاص المرشح الأنسب. كما يوفر مشروعنا حلاً مفتوح المصدر، مجانياً، وقابلًا للتوسع، بأقصى قدر من الدقة والكفاءة.

المنهجية: اعتمدنا منهجية Agile نظرًا لمرونتها الكبيرة في مراحل الاختبار والتعديل، وتعزيز التعاون بين أعضاء الفريق من خلال التكامل المستمر للمكونات المختلفة، بالإضافة إلى العديد من الميزات المهمة الأخرى. كما أنها لبت احتياجاتنا من حيث التخطيط، الاجتماعات اليومية، والتطوير والتعديل التكراري.

النتائج: تمتلك أدواتنا القدرة على معالجة نصوص الوظائف وملفات السير الذاتية بالإضافة إلى أي معايير إضافية يحددها المستخدم، ثم استخراج الميزات الأساسية من كل سيرة ذاتية، ومقارنتها مع نص الوظيفة، وحساب درجات التشابه بينها، ومن ثم تصفية السير الذاتية التي تتجاوز قيمة حدية معينة. بعد ذلك، يتم فرزها لاختيار أفضل المرشحين، مع تقديم تحليل تفصيلي لنقاط القوة والضعف لكل مرشح، وأخيراً اختيار السيرة الذاتية الأفضل مع مبررات دقيقة، مما يحقق أهدافنا بشكل كبير بل وتتجاوزها. تعتمد أدواتنا في تحليلها على حساب درجات التشابه استناداً إلى التضمينات (embeddings) التي يولدها نظامنا، مما يعزز الفهم الدلالي للسير الذاتية ويوفر تحليلاً أكثر دقة لكل سيرة ذاتية خلال عملية الفرز. جميع هذه الميزات قد تم اختبارها بدقة كما هو موضح في فصل الاختبار.

الاستنتاجات والتوصيات: يُظهر مشروعنا مدى الحاجة إلى هذه الأداة، حيث يوضح العديد من مميزات المذهلة وإمكاناتها للتكامل والتطوير المستمر، مما يجعلها أكثر أهمية و استحالة الاستغناء عنها للأفراد والشركات من جميع الأحجام.

Dedication

We dedicate this project to our families who have been very helpful and supportive during us making this project under the rough circumstances we have all been going through.

Acknowledgment

We'd like to acknowledge all the great people that have helped ease the creation of this project starting with our supervisor who has gone above and beyond in fulfilling his mentoring duties, we are really grateful and honored to have him as a mentor, then comes the great people that have been doing their best to make sure students continue their education by providing places with internet and electricity under these difficult circumstances.

Table of Contents

Dedication	4
Acknowledgment	5
Table of Contents	6
List of Tables	9
List of Figures	10
List of Abbreviations	11
Chapter 1	12
Introduction	12
Chapter 1	
Introduction	1
1.1 Problem Statement	9
1.2 Objectives	10
1.2.1 Main Objective	10
1.2.2 Sub Objectives	10
1.3 Scope and Limitations	10
1.3.1 Scope	10
1.3.2 Limitations	10
1.4 Importance of the project	12
Chapter 2	14
Related Works	14
Chapter 2	
Related Works	15
2.1 Introduction	15
2.2 Table of related works	15
2.3 How our project differs	24
Chapter 3	25
Methodology	25
Chapter 3	
Methodology	26
3.1 Name of the methodology	26
3.2 Rationale for choosing the methodology:	26
3.3 Key practices and phases of the methodology:	26
3.4 Adaptations or modifications to the standard methodology:	27
3.5 Tools and Equipment	27
3.6 TimeTable	28
3.7 Team Management	28
Chapter 4	30
Requirements Analysis	30
Chapter 4	
Requirements Analysis	31

4.1 Functional Requirements	31
4.2 Non-Functional Requirements	32
4.3 Conclusion	33
Chapter 5	34
System Design	34
Chapter 5	
System Design	35
5.1 Overview	35
5.2 System Architecture	36
5.2.1 System Architecture Diagram	36
5.2.2 Pipeline Breakdown	39
5.3 AI Model Details	41
5.3.1 Model Type	41
5.3.2 Training Data	41
5.3.3 Prompts	41
5.3.4 Inference	43
5.3.5 Tested LLM models	43
5.3.6 LLM models' comparison	44
Chapter 6	47
Implementation and Coding	47
Chapter 6	48
Implementation and Coding	48
6.1 Overview	48
6.2 Programming Language and Tools	48
6.3 Libraries and Frameworks	48
6.4 Development Environment	48
6.5 Key Components and Implementation	48
6.6 Conclusion	51
Chapter 7	52
Testing, Evaluation, and Deployment	52
Chapter 7	
Testing, Evaluation, and Deployment	53
7.1 Overview	53
7.2 Testing Strategy	53
7.2.1 Unit testing	53
7.2.2 Integration Testing	54
7.2.3 User Acceptance Testing (UAT)	54
7.3 Deployment	55
7.3.1 Deployment requirements	55
7.4 Test Case and Execution	55
7.5 Results and Observations	59
Conclusions and future works	62
8.1 Conclusions	62
8.2 Future Work	62

8.3 Final Remarks	63
References	64

List of Tables

Table (1): AI-powered vs. Traditional Hiring	2
Table (2): Processing Comparison	3
Table (3): LLM-based resume screening vs. other AI methods	4
Table (4): Similar tools/platforms	13
Table(5): Project Execution Timetable	26
Table (6): Pipeline Stages and Tools	35
Table(7): Tested LLM models	38
Table(8): LLM model comparison	39

List of Figures

Figure (1): Hiring process periods in days (traditional methods versus AI-driven automation).	6
Figure (2): Candidate satisfaction levels at different stages of the hiring process (traditional methods versus AI-driven automation).	7
Figure (3): Cost per hire (traditional methods vs AI-driven automation).	7
Figure (4): Automated Hiring process system architecture diagram	34
Figure (5): LLM Pipeline Flowchart	34
Figure (6): Testing text extraction from PDF and DOCX files	50
Figure (7): Integration test for similarity calculation	50

List of Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
ATS	Applicant Tracking Systems
CV	Curriculum Vitae
DOCX	Document Open XML
HR	Human Resources
IDE	Integrated Development Environment
LLM	Large Language Model
ML	Machine Learning
NLP	Natural Language Processing
PDF	Portable Document Format
UI	User Interface
XML	Extensible Markup Language

Chapter 1

Introduction

Chapter 1

Introduction

In our digital age, high accuracy and speed are crucial in all aspects, including the recruitment market. Organizations face significant challenges in their recruitment processes, with the goal being to find the most suitable candidate quickly, transparently, and cost-effectively. Traditional recruitment methods are slow, inconsistent, prone to human biases, and costly, paving the way for Artificial Intelligence (AI) to enter this market (Table 1).

With AI being faster, less biased, cheaper, and generally more efficient in hiring, it has significantly impacted the resume screening process. Compared to traditional methods, AI performs the resume screening task more smoothly and efficiently (Table 2).

Background:

Research titled *"Design and Enhancing Recruitment Processes: A Comparative Analysis of AI-Driven Automation Versus Traditional Methods"* (Shukoor & Bhaumik, 2024) highlights the significant differences between traditional and AI-based methods. Figures 1, 2, and 3 from this analysis compare the hiring time (in days), candidate satisfaction throughout the hiring process, and the cost per hire. The results demonstrate clear advantages of AI-based tools, including shorter recruitment periods, higher candidate satisfaction, and reduced hiring costs.

Utilizing Large Language Model (LLM)-based approaches that employ Machine Learning (ML) and Natural Language Processing (NLP) has significantly advanced the industry due to their ability to comprehend human-like text. These models analyze resumes contextually, reducing biases in candidate selection. Additionally, their capability to adapt quickly to different tasks, offering scalability and flexibility in processing diverse resumes and job descriptions, makes them particularly favored over other AI tools (Table 3).

Definition of Key Terms:

- Artificial Intelligence (AI): The capability of computational systems to perform tasks typically associated with human intelligence, such as learning, reasoning, problem-solving, perception, and decision-making [1].

- Resume Screening: The process of reviewing a resume to determine if the candidate is qualified for the position [2].
- Natural Language Processing (NLP): A subfield of computer science and artificial intelligence (AI) that uses machine learning to enable computers to understand and communicate with human language [3].
- Machine Learning (ML): A branch of artificial intelligence (AI) focused on enabling computers and machines to imitate the way that humans learn, to perform tasks autonomously, and to improve their performance and accuracy through experience and exposure to more data [4].
- Large Language Model (LLM): A category of foundation models trained on immense amounts of data making them capable of understanding and generating natural language and other types of content to perform a wide range of tasks [5].

Table (1): AI-powered vs. Traditional Hiring [6]

Feature	AI-Powered Hiring	Traditional Hiring
<i>Speed</i>	Lengthy hiring cycles often take weeks or months.	AI-driven screening and matching reduce hiring time by up to 70%.
<i>Efficiency</i>	HR teams manually sift through resumes and applications.	Automated resume screening and smart matching streamline the process.
<i>Cost</i>	High operational costs due to manual efforts and extended hiring timelines.	Reduced hiring costs through automation and optimized processes.

<i>Bias & Objectivity</i>	Prone to unconscious bias in resume shortlisting and interviews.	AI-driven assessments ensure hiring decisions are based on data, not personal bias.
<i>Quality of Hire</i>	Often relies on gut feeling rather than data-driven insights.	AI analyzes multiple factors, ensuring the best talent fit for the role.
<i>Scalability</i>	Difficult to handle high-volume hiring efficiently.	AI can process thousands of applications simultaneously without additional effort.

Table (2): Processing Comparison ^[6]

Process	Description
<i>AI-Powered Hiring</i>	<ol style="list-style-type: none"> 1. Real-time screening of incoming applications. 2. AI ranks candidates based on match quality. 3. Provides a shortlist of top 5 candidates efficiently.
<i>Traditional Hiring</i>	<ol style="list-style-type: none"> 1. Manually screen each resume. 2. Coordinate with the team for reviews. 3. Track candidate status in spreadsheets. 4. Manually rank each candidate.

Table (3): LLM-based resume screening vs. other AI methods

Aspect	LLM-Based Resume Screening	Other AI Methods
<i>Type of AI Technology</i>	Utilizes advanced Natural Language Processing (NLP) and Machine Learning (ML) techniques to understand and generate human-like text, enabling the analysis of resumes with contextual comprehension.	Employs various AI technologies, including:- Keyword-Based Systems: Identify specific phrases and patterns in resumes to match job requirements.- Grammar-Based Systems: Analyze sentence structures to interpret resume content.- Statistical Models: Use numerical data, such as word frequency and timelines, to assess candidate suitability.- Traditional Machine Learning Algorithms: Apply supervised learning techniques to classify and rank resumes based on predefined features.
<i>Contextual Understanding</i>	Excels at comprehending the nuances of human language, allowing for sophisticated analysis of resumes and extraction of relevant information even when candidates use varied terminologies or formats.	May rely on keyword matching and structured data, potentially missing nuanced information or context within resumes.
<i>Handling Unstructured Data</i>	Capable of processing unstructured text, making them adapt to parsing resumes that	Often require structured input data, which may necessitate additional preprocessing steps

	don't follow a standard template.	to handle diverse resume formats.
<i>Bias Mitigation</i>	By focusing on the content and context of resumes, LLMs can help reduce biases related to gender, age, or ethnicity, promoting fairer hiring practices.	Might inadvertently perpetuate existing biases present in the training data, leading to potential fairness concerns.
<i>Computational Resources</i>	Requires significant computational power, which might be a constraint for smaller organizations.	Typically have lower computational requirements, making them more accessible to organizations with limited resources.
<i>Interpretability</i>	The decision-making process can be opaque, making it challenging to understand the rationale behind specific candidate evaluations.	Especially rule-based systems, offer greater transparency in their decision-making processes, allowing for easier interpretation and adjustments.
<i>Scalability and Flexibility</i>	Can adapt to various tasks with minimal adjustments, offering scalability and flexibility in processing different types of resumes and job descriptions.	May require significant reconfiguration to adapt to new tasks or changes in resume formats, potentially limiting scalability.
<i>Development and Maintenance</i>	Implementing LLM-based systems can be complex and may require ongoing maintenance to ensure optimal performance.	Particularly rule-based systems, are generally easier to implement and maintain but may lack adaptability to

		evolving language patterns and resume formats.
<i>Processing Speed</i>	Can process large volumes of resumes efficiently, significantly reducing the time required for initial candidate screening.	Also offer efficient processing but may not match the speed and accuracy of LLMs in handling unstructured and diverse resume data.
<i>Integration with Existing Systems</i>	Integrating LLMs into existing recruitment workflows may require significant adjustments and technical expertise.	Especially those already in use, may integrate more seamlessly into current systems with minimal modifications.

[7] [8] [9] [10] [11] [12] [13] [14] [15]

Figure (1): Hiring process periods in days (traditional methods versus AI-driven automation)^[16].

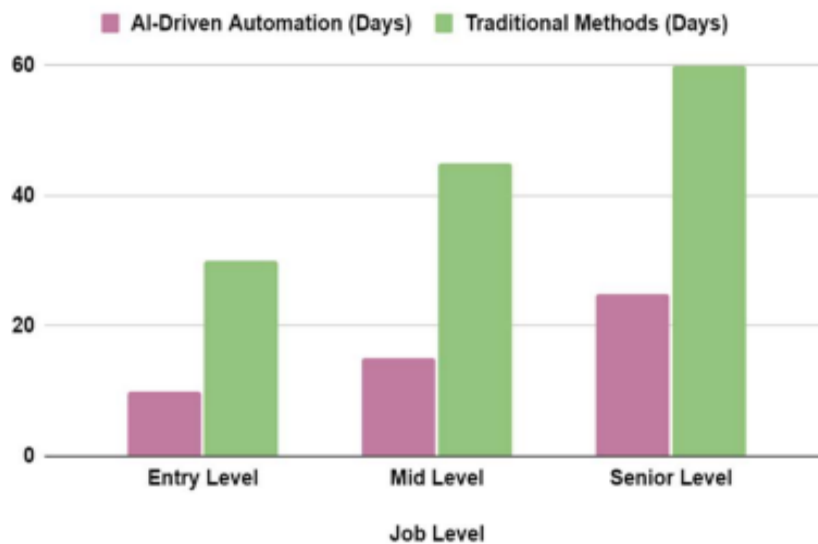


Figure 1 compares the time it takes to hire for different job levels using traditional recruitment methods and AI-powered automation.

AI-driven recruitment can significantly reduce recruiting timelines at all job levels by automating processes such as resume screening leading to faster onboarding and more efficient recruitment workflows, according to data.

Figure (2): Candidate satisfaction levels at different stages of the hiring process (traditional methods versus AI-driven automation) [16].

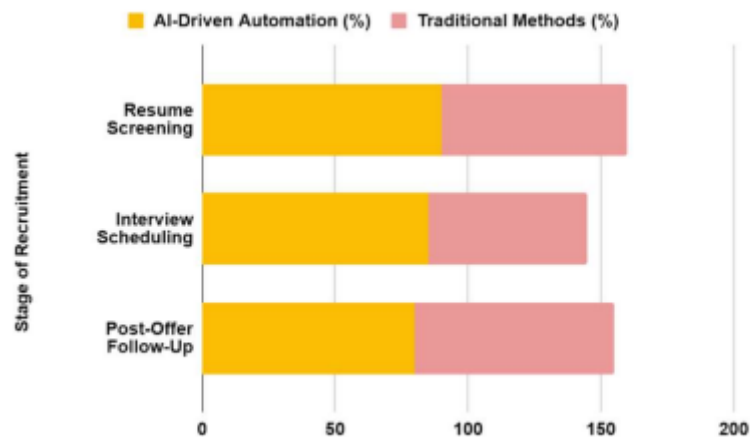
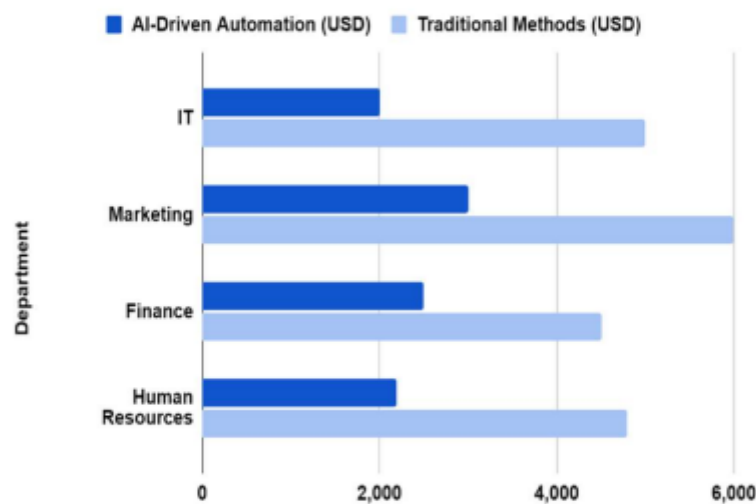


Figure 2 compares candidate satisfaction levels at different stages of the hiring process using both traditional methods and AI-powered automation. AI ensures faster and more precise screening of resumes, leading to a higher satisfaction rate compared to the satisfaction rate with traditional methods.

Figure (3): Cost per hire (traditional methods vs AI-driven automation) [16].



The cost per hire for traditional recruitment techniques and AI-driven automation is compared across departments in Figure 3.

AI automation is a cost-effective option for businesses, especially start-ups looking to save money on hiring across various departments by cutting down manual work and inefficiencies in operations.

1.1 Problem Statement

We addressed the issue of the complicated hiring process that many companies must endure, the frequent biases that arise in these hires, and the time and resources that are often wasted in the process. Our tool is designed to effectively address all of these challenges in the most efficient manner possible.

Our tool is an AI-powered resume screening tool that automates the candidate selection phase in the hiring process. Leveraging AI helps us achieve our aim to minimize time and resources, while preventing potential bias that occasionally occurs in that process. Our tool analyzes the resumes provided by the companies and helps find the top candidates for a role based on the specified job description they provide.

1.2 Objectives

1.2.1 Main Objective

The primary goal of this project is to create an AI tool that can match the best candidate for a job based on the job description and candidate resumes provided by the company. This tool is designed to streamline and improve the recruitment process by identifying the most suitable candidate for the job with greater accuracy and efficiency.

1.2.2 Sub Objectives

- Provide a simple resume submission interface to collect resumes efficiently.
- Evaluate each resume according to its match percentage to the provided job description with providing the score and explanation of why it's given.
- Produce a shortlist of the top 5 candidates based on the score of each resume summarized.
- Analyze the resulting shortlist
- Provide the Strengths and weaknesses of top 5 candidates.

1.3 Scope and Limitations

1.3.1 Scope

Our Project targets companies and organizations that prioritize credibility and transparency in hiring. We aim to help start-ups find the best fit considering their limited/low budget as our service is open-source and free. In addition, individuals seeking specific jobs and wanting to test their resumes against a chosen job description to build on their strengths and eliminate weaknesses.

1.3.2 Limitations

Although our project offers a useful and reliable tool, there are a number of limitations to consider:

1. The accuracy of the AI model: While we created our tool using LLM models that are powerful, it may not be perfect. Instances of misinterpretation or inaccuracy may occur, especially with resumes that do not follow the standard structure.
2. Dependence on the job description quality: The quality of the job description text plays a vital role in the results. A poorly written or nuclear job description may give unsatisfactory and misleading results.
3. Potential needed customization needs for some specific industries: We designed our tool to be broad, which might not address the unique needs of every industry or job role. As a result, recruiters might need some customization to the tool, which is possible because our tool is available as an open-source.
4. Data privacy and security: As our project is open-source, ensuring data privacy and security is essential. So, users of the platform must take part and act responsibly for how they handle and protect their sensitive data while using the tool.
5. No support for advanced candidate evaluation: Even though our tool provides a detailed analysis of resumes, it does not extend to other stages of the hiring process, such as interview assessments, which are important for making hiring decisions.

6. No ATS integration availability: Companies and start-ups can't integrate their ATS with our tool, which makes the data not updated and only leaves them using the tool manually.
7. Requires significant computational power, which might be a constraint for organizations with low quality resources.
8. Integrating LLMs into existing recruitment workflows may require significant adjustments and technical expertise.
9. Implementing LLM-based systems can be complex and may require ongoing maintenance to ensure optimal performance.

By understanding these limitations, users can make more conscious decisions about when and how to use our tool, and a space for development is discovered.

1.4 Importance of the project

Our project aims to elevate the hiring process by using AI models to automate resume screening. Making the project open-source and free provides an efficient, unbiased, and scalable solution. We look forward to assisting not only companies in finding the best candidates, but also individuals who want to evaluate their fit for a specific job.

Our tool helps job seekers understand the strengths and weaknesses of their CVs to a specific job description they want to apply for, improving their chances of securing that desired role. "Ranking of the CVs" feature, as well as "finding strengths and weaknesses of the top 5 candidates" one, will allow companies to "carefully choose" the best candidate for the job they're offering.

On top of that, the project contributes to the hiring industry, as human error will be minimized, and conscious and unconscious bias. Because of the use of AI and data-driven models, a more fair selection process will be ensured.

By offering an accessible and free alternative to expensive hiring platforms, we support startups and organizations that may lack the resources for traditional recruitment tools (Figure 3). Not only companies but also

individuals; as they can use our tool to evaluate their resume for each job they're applying for easily and for free.

Our project will foster continuous improvement within the AI-driven hiring landscape as it is open-source.

With these advantages, our project has the potential to make a meaningful impact on job seekers, companies, and the broader recruitment industry.

Chapter 2

Related Works

Chapter 2

Related Works

2.1 Introduction

This chapter provides a comprehensive overview of various AI-powered resume screening tools and platforms. We're trying to show the difference between our tool and the many existing ones. The chapter highlights a specific use case, which is shortlisting resumes and candidates. Overall, this chapter offers valuable insights on existing tools and the additional value we provide.

2.2 Table of related works

Table (4): Similar tools/platforms

Tool / Platform	Key Features	Benefits	Use Cases	Shortlisting
<i>SkillPool</i> ^[17]	AI-powered skills assessment	Informed decision-making, focus on competency	Prioritizing skills over qualifications	<ul style="list-style-type: none"> Find Best Fit feature (Fit scale to 5). Customize your candidate ideal profile and their AI will show the best matches. Automate initial interviews using AI.
<i>HireBeat</i> ^[18]	ATS, customizable screening, AI capabilities	Efficient resume screening, candidate ranking	Identifying top talent	<ul style="list-style-type: none"> Find top talent with the right work experience and skill sets.
<i>Textkernel</i> ^[19]	Semantic search and matching	Accurate resume matching,	Focus on semantic analysis	<ul style="list-style-type: none"> Matches talents to jobs and doesn't find

		improved candidate quality		top candidates for a job specifically.
<i>Daxtra</i> ^[20]	Advanced resume parsing and matching	Reduced manual effort, improved hiring speed	Large-scale recruitment	<ul style="list-style-type: none"> (Daxtra Search Nexus) Quickly search, filter and sort data from job boards, social networking sites and your ATS or CRM to find the most relevant candidates. Streamline your searches to save time and increase productivity, so you find the best candidates before anyone else.
<i>Sniper AI</i> ^[21]	Machine learning for candidate screening	Fast and accurate resume matching, reduced workload	High-volume recruitment	<ul style="list-style-type: none"> Allows CVs to be reviewed based on suitability score. Automate the talent sourcing process with AI to find the best candidates quickly.
<i>Hubert</i> ^[22]	Holistic AI recruiting platform	Transparent advice, reduced bias	Data-driven hiring decisions	<ul style="list-style-type: none"> The Hubert interview effectively combines all the data that is normally collected from knock-out questions, CV screening, and

				<p>telephone interviews into one single user-friendly chat interview that is automatically scored.</p>
<p><i>Mosaictrack</i> [23]</p>	<p>AI-powered resume and social profile screening</p>	<p>Predictive analytics, enhanced interview process</p>	<p>Culture fit and skill-based selection</p>	<ul style="list-style-type: none"> Start recruiting the best candidates with ease. Your resumes and job descriptions are synced from your ATS, making recruiting with AI seamless.
<p><i>XOR</i> [24]</p>	<p>AI platform for recruiting teams</p>	<p>Automated resume screening, interview scheduling, onboarding</p>	<p>Streamlined recruitment workflows</p>	<ul style="list-style-type: none"> Attracts, pre-screens and then schedules the best candidates into recruiters' calendars so that you can focus on hiring. Employs the natural language processing (NLP) capabilities of ChatGPT to provide a conversational recruiter chatbot and consolidated communication platform.
<p><i>PestoAI</i> [25]</p>	<p>AI-driven platform for remote developers</p>	<p>Efficient remote hiring, improved candidate quality</p>	<p>Hiring skilled remote developers</p>	<ul style="list-style-type: none"> Uses GPT-powered vetting.

				<ul style="list-style-type: none"> • Employs state-of-the-art AI algorithms to ensure precise matching between companies and developers based on project requirements and specific needs.
<i>HireVue</i> ^[26]	Video interviewing, AI-based assessment	Deeper candidate insights, reduced bias	Roles requiring communication skills	<ul style="list-style-type: none"> • Offers predictive analytics to forecast candidate success and fit within the organization. • Combines video interviewing with AI assessments.
<i>Pymetrics</i> ^[27]	Neuroscience-based games, AI-driven insights	Holistic candidate view, reduced bias, improved diversity	Roles requiring cognitive and emotional intelligence	<ul style="list-style-type: none"> • Uses neuroscience games for candidate assessment, providing AI-driven insights into their cognitive and emotional traits, offering a fresh and scientifically backed approach to evaluating candidates' suitability for roles.

<i>HireEZ</i> ^[28]	AI-powered talent sourcing and engagement	Expanded talent pool, improved engagement	Large-scale hiring and talent acquisition	<ul style="list-style-type: none"> • Uses advanced AI-driven sourcing algorithms that prioritize matching candidates with specific job requirements. • Provides deep insights into candidate fit through advanced analytics.
<i>XOPA AI</i> ^[29]	End-to-end talent management platform	Enhanced hiring efficiency, reduced bias	Comprehensive AI-driven hiring	<ul style="list-style-type: none"> • XOPA embedded AI Verify into its ATS solution to shortlist candidates for jobs. • Employs a gradient boosting model to help its clients predict shortlisted candidates for posted job positions.
<i>Arya</i> ^[30]	AI-powered sourcing and engagement	Reduced sourcing time, enhanced candidate experience	Data-driven recruitment	<ul style="list-style-type: none"> • Uses machine learning to identify successful sourcing patterns and draws potential candidates out of millions of online profiles. • Provides a single, deduplicated, stack-ranked list of

				best-fit candidates sourced from an organization's ATS, as well as job board accounts.
--	--	--	--	--

<i>Zoho Recruit</i> [31]	AI capabilities for resume parsing and matching	Streamlined hiring, enhanced candidate matching	Integrated recruitment solution	<ul style="list-style-type: none"> AI-powered scoring which greatly enhances the hiring process by accurately predicting candidates' success, and provides data-driven recommendations for hiring decisions, further streamlining the process and improving recruitment outcomes.
-----------------------------	---	---	---------------------------------	--

<i>Manatal</i> [32]	AI-powered recruitment software	Enhanced recruitment efficiency, improved candidate management	Comprehensive and user-friendly solution	<ul style="list-style-type: none"> AI-powered recommendation engine. This feature recommends the most suitable jobs for your candidates and the most suitable candidates for your jobs within seconds. Manatal will quickly scan through your database, give scorecards, and come
---------------------	---------------------------------	--	--	---

				up with a list of the most suitable candidates without any biases.
<i>Recruitee</i> [33]	AI-driven candidate matching and analytics	Improved candidate fit, enhanced collaboration	Data-driven hiring decisions	<ul style="list-style-type: none"> • Use ATS to filter and rank candidates: Highlights the automation capabilities of the Applicant Tracking System (ATS). • Determine the length of your shortlist: Suggests setting a manageable shortlist size.
<i>Mya</i> [34]	Conversational AI recruiting assistant	Enhanced candidate experience, reduced workload	High-volume hiring and candidate communication	<ul style="list-style-type: none"> • Employers can post job listings, set desired criteria, and customize screening questions to efficiently filter and shortlist candidates. The platform also provides detailed analytics and insights to track the performance of job postings and monitor candidate engagement.

<i>Eightfold AI</i> ^[35]	Talent intelligence platform	Improved talent discovery, enhanced engagement	Comprehensive talent management	<ul style="list-style-type: none"> Eightfold's deep learning AI scans an entire Talent Network to find candidates and ranks them in order of how closely they match your needs. This screening considers only qualifications, and shows why each candidate is recommended so the recruiter and hiring manager can decide who to shortlist.
<i>Jobvite</i> ^[36]	AI capabilities for candidate matching and engagement	Enhanced candidate engagement, improved match quality	Robust recruitment solution	<ul style="list-style-type: none"> Candidate skill matching is a strategy that helps companies to find the right qualified talent without eliminating too many candidates early in the process.
<i>Beamery</i> ^[37]	AI-powered recruitment marketing and talent engagement	Enhanced talent engagement, improved candidate experience	Recruitment marketing and talent relationship management	<ul style="list-style-type: none"> Beamery uses advanced algorithms to screen candidates based on skills and potential, rather than simply their previous experience. Beamery's AI system can provide

explainable AI that allows recruiters to understand the rationale behind the matching and ranking of candidates. It also assesses candidates based on their similarity to high performers in the same role at your company.

Harver ^[38]

AI-powered talent acquisition software

Reduced bias, improved assessment accuracy

Improved assessment and selection processes

- Determine your criteria, and decide a shortlist maximum number.
 - Try blind applicant screening.
 - Eliminate applicants who don't have the criteria you're looking for.
 - Screen candidates in, not out.
 - Try assessments during the initial application phase.
 - Conduct a screening interview.
 - Give your candidates a score.
-

-
- Let candidates know if you're not moving forward.
-

2.3 How our project differs

This section showcases the enhancements that our approach offers to the recruitment field, such as:

1. Our tool is open-source, which sets it apart from the tools listed in the table and enables clearer decision-making.
2. We offer a complimentary platform for both employers and job seekers.
3. Creating a shortlist of the top five candidates for the job opening.
4. We assess the skills, experience, and deficiencies of each candidate, providing recruiters with a more comprehensive understanding than just matching keywords.
5. Analyzing the strengths and weaknesses of the top five candidates.
6. Our model is designed to quickly process large volumes of resumes, resulting in faster and more accurate hiring decisions.
7. Implementing a model that is both more efficient and scalable.

Chapter 3

Methodology

Chapter 3

Methodology

3.1 Name of the methodology

Agile methodology was used for the development of our cv-shortlist assistant project.

3.2 Rationale for choosing the methodology:

- Agile's Flexibility: The feedback and testing phases of the project were dynamic, requiring frequent changes to be made. Agile incorporates flexibility through its iterative process.
- Agile's Collaboration: Close collaboration among team members and stakeholders was critical for the integration of different components such as OpenRouter API, embeddings and Streamlit UI.
- Agile's Incremental Progress: The project was fragmented into smaller tasks such as file processing, feature extraction, embedding generation and similarity calculation, permitting continuous development and earlier delivery of functional components.
- Agile's Adaptability: Given that the integration of AI models with embeddings was still experimental, having the ability to alter requirements or priorities was crucial.

3.3 Key practices and phases of the methodology:

The agile methodology was implemented based on the following key practices and phases:

- sprint planning: the project was divided into sprints (2 weeks-cycles), with each sprint covering specific deliverables
 1. sprint 1: File processing and text extraction(PDF, DOCX)
 2. sprint 2 : Feature extraction using OpenRouter API.
 3. sprint 3: Embedding generation and similarity calculator
 4. sprint 4: streamlit UI development and integration
- Daily stand-ups: Daily stand-up meetings were held to discuss progress, problems, and next steps.

- Iterative Development: every sprint produced a functional piece of the application, Feedback from testing and stakeholders was incorporated into the following sprint.
- Continuous integration: code was regularly integrated into the main branch via Git and Github.

Automated tests assisted in making sure new changes did not make existing functionality breaks.

- Testing and feedback:
testing of the features built was included in each sprint. Feedback from the users (e.g., recruiters) was used to enhance the application.

3.4 Adaptations or modifications to the standard methodology:

Hybrid methodology: while agile is the main framework, certain parts of waterfall were incorporated for more defined processes, such as text extraction and file processing. Extended sprints: work in the domain of AI embedded models integrated with OpenRouter API tended to be deeper, requiring a full test and tune cycle to be included within a single sprint.

3.5 Tools and Equipment

The following tools and technologies were in use throughout the entire project :

1. Software tools:

- . programming language: python

- . Frameworks and Libraries :

- StreamLit (for the web interface)

- Transformers (for embedding creation)

- Numpy and Scikit-learn (for calculating similarity)

- Requests (for API communication)

2. Working Environments:

- IDE : Visual Studio Code, pyCharm.

- Vision Control: Github and Git.

- Project Management: Task tracking and sprint planning are done in

Trello.

3. Equipment:

- PC/Laptop: Must possess strong processing capabilities to run AI models

and produce embeddings.

3.6 TimeTable

Table(5): Project Execution Timetable

Week	Task
1-2	project planning, requirement gathering, and tool setup
3-4	File processing and text extraction (PDF, DOCX)
5-6	Feature extraction using openRouter API
7-8	Embedding generation and similarity calculation
9-10	StreamLit UI development and integration
11	testing, debugging, and refinement
12	final documentation and project submission

3.7 Team Management

- a. Team Structure and Roles:
 - . Project Manager: oversaw the project timeline, sprint planning, and coordination.
 - . Backend developer: Handle file processing, feature extraction, and embedding generation.
 - . Frontend Developer: developed the StreamLit UI and integrated it with the backend.
 - . AI specialist: integrated the OpenRouter API, fine-tuned the embeddings and cosine similarity generation processes, and engineered prompts to produce the required results.
 - . Tester: Conducted testing and provided feedback for improvements.
- b. Communication and Collaboration Tools:
 - . Colab: For version control and collaboration.
 - . Zoom: For sprint review and retrospectives.

- c. Conflict Resolution Strategies:
 - . Conflicts were resolved through open communication and collaborative problem-solving.
 - . If disagreements arose, the team would discuss the issue during stand-ups or schedule a separate meeting to reach a consensus.
- d. Teamwork and Delegation:
 - . Tasks were delegated based on team members' expertise and availability.
 - . Regular check-ins ensured that everyone was on track and any bottlenecks were addressed promptly.

Chapter 4

Requirements Analysis

Chapter 4

Requirements Analysis

4.1 Functional Requirements

The functional requirements describe what the system should do, here's a detailed breakdown of the functional requirements for our project:

1. Add the job description and upload the resumes:
 - The system should allow users to add the job description text and upload the resume file in multiple formats(PDF, DOCX).
 - the system should validate the uploaded files to ensure that they are in the correct file format and contain readable text.

2. Extract features from text using AI:

The system should use the OpenRouterAPI to extract key information from the job description and resumes, including:

- Email address
- Technical skills
- Soft skills
- Experience
- Education

3. Generate Embeddings for the text:

- The system should generate embeddings(numerical representations) for the job description and resumes using a pre-trained model `openai-embed-text`.
- The embedding should capture the semantic meaning of the text for accurate similarity comparison.

4. Calculate Similarity Scores:

The system should display the following results in a clear and intuitive interface:

- Extracted feature from resumes
- Similarity score of each resume
- A shortlist of top 5 matching resumes based on similarity scores and relevance to job description.

- An analysis of the strengths and weaknesses for the shortlist of resumes, and a conclusion of the best resume.

4.2 Non-Functional Requirements

The non-functional requirements describe how the system should be performed, here's a detailed breakdown of the non-functional requirements for our project:

1. Performance: the system should handle large files(e.g., resumes with multiple pages) without significant delays.
2. Usability:
 - The interface should be intuitive and easy to use, even for non technical users.
 - The system should provide clear instructions and feedback(e.g., success message, error message).
 - The interface should be responsive and work seamlessly on both desktop and mobile.
3. Scalability:
 - The system should handle multiple files without performance degradation.
4. Reliability:
 - The system should be robust and handle errors gracefully (e.g., invalid file formats, API failures).
 - The system should provide fallback mechanisms (e.g., create API keys) in case of temporary failures.
5. Security:
 - The system should ensure the privacy and security of uploaded files and extracted data.
 - The system should not store sensitive information (e.g., resumes) permanently unless explicitly requested by the user.
6. Maintainability:
 - The codebase should be well-documented and follow best practices (e.g., PEP 8 for Python).
 - The system should be modular, making it easy to update or replace components.

7. Compatibility:

- The system should be compatible with common operating systems (e.g., Windows, macOS, Linux) and web browsers (e.g., Chrome, Firefox, Safari).

8. Cost:

- The system should minimize costs by optimizing resource usage (e.g., reducing API calls, using efficient algorithms).

4.3 Conclusion

The functional and non-functional requirements outlined above ensure that the Resume Matcher application is user-friendly, efficient, and scalable. By addressing these requirements, the system will meet the needs of recruiters and provide a valuable tool for automating the resume screening process.

Chapter 5

System Design

Chapter 5

System Design

5.1 Overview

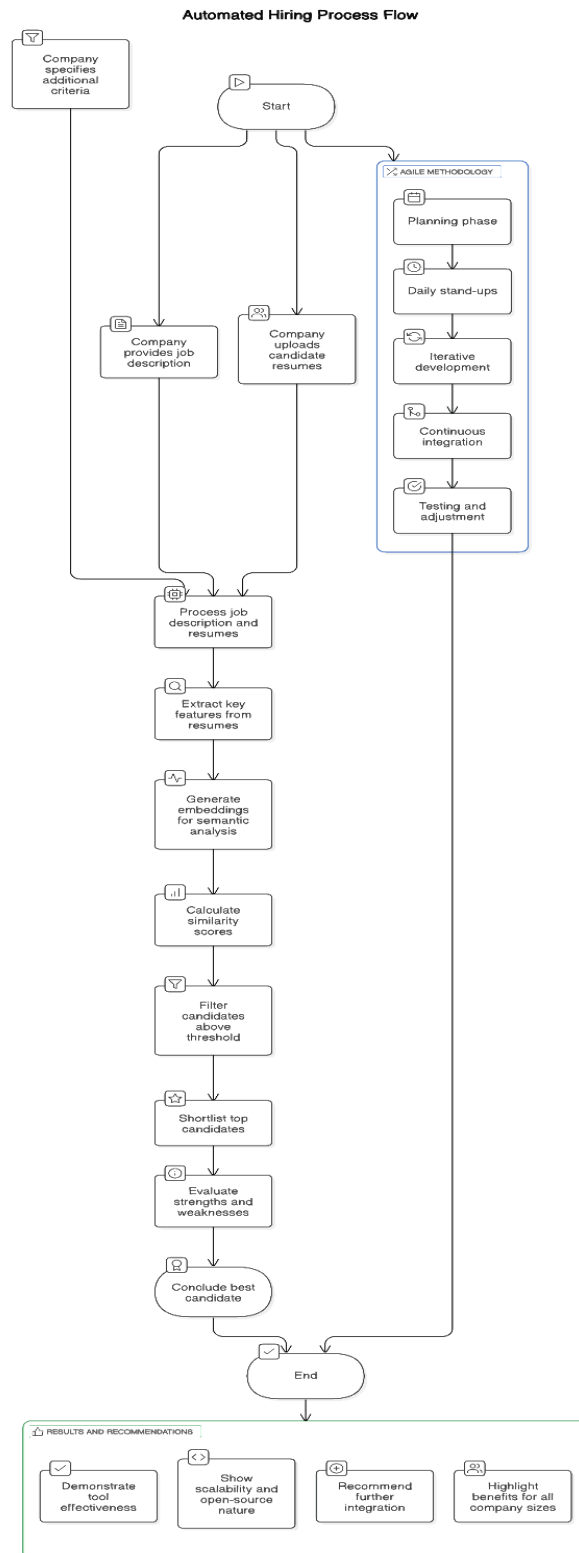
This chapter provides a comprehensive explanation of the system's architecture. The system's design ensures scalability, efficiency, and adaptability to different hiring needs while it follows a structured pipeline for efficient and unbiased candidate selection.

5.2 System Architecture

5.2.1 System Architecture Diagram

A system architecture diagram is a blueprint of a software system, showcasing its core components, their interconnections, and the communication channels that drive functionality. This allows for a clear understanding of how the system works and how it can be improved. Figure 4 demonstrates the different processes used in our automated hiring system in a sequential manner.

Figure (4): Automated Hiring process system architecture diagram



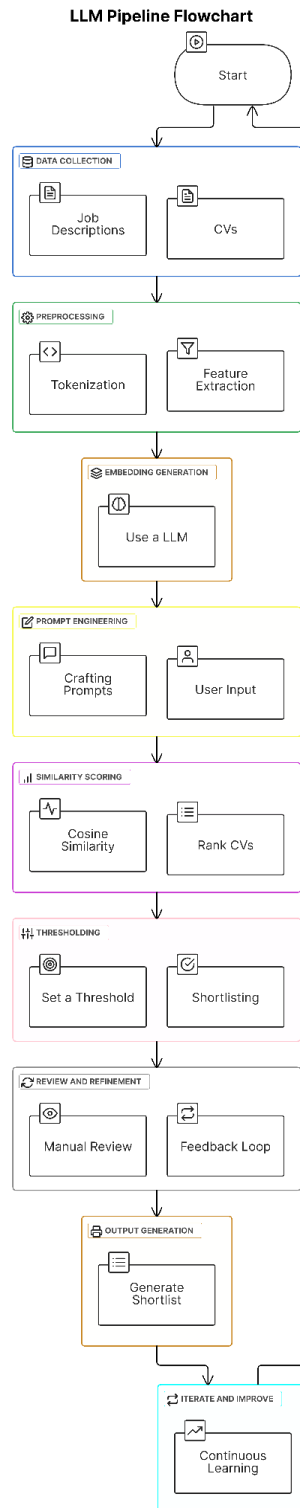
For a better view of our system architecture diagram, visit the link:
<https://app.eraser.io/workspace/ZgdxLoDESthLwnW3Pv1d?origin=share>

5.2.2 Pipeline Breakdown

Figure 5 presents the flow of the system data, starting from collecting the job description and resumes to the different stages they go through to final output.

Each block presented in the chart represents a major process, including preprocessing, prompt engineering, thresholding, and iteration and improvement.

Figure (5): LLM Pipeline Flowchart



Note: Due to the diagram's wideness, it is hard to share it clearly in the document. For a better view of the diagram, visit the link:

<https://app.eraser.io/workspace/DTtzx6UA6Z8QD1eiGKI8?origin=share>

The system operates in stages, and each has specific tasks and associated libraries.

Table (6): Pipeline Stages and Tools

Task	Subtasks	Used Libraries
<i>Data Collection</i>	<ul style="list-style-type: none"> ● Gather the job description that outlines the required skills, qualifications, and responsibilities. ● Collect the CVs or resumes that you want to evaluate against the job description. ● Extract text from different resume file extensions. 	<ol style="list-style-type: none"> 1. streamlit 2. docx, pdfplumber
<i>Preprocessing</i>	<ul style="list-style-type: none"> ● Use LLMs to Identify key features from the job description, such as required skills, experience, and qualifications. 	ollama + cheahjs/free-llm-api -resources: A list of free LLM inference resources accessible via API.
<i>Embedding Generation</i>	<ul style="list-style-type: none"> ● Utilize a pre-trained LLM (like BERT, GPT, Nomic, etc.) to generate embeddings for both the job description and the CVs. This converts the text into numerical representations that capture semantic meaning. 	<ol style="list-style-type: none"> 1. Transformers 2. torch
<i>Prompt Engineering</i>	<ul style="list-style-type: none"> ● Design specific prompts that instruct the LLM to evaluate the CVs based on the job description and user preferences. For example, prompts could include: "Select the top 5 CVs that best match the requirements outlined in this job description." ● User Input: Allow users to specify additional criteria or preferences (e.g., years of experience, specific skills) to tailor the evaluation. 	<ol style="list-style-type: none"> 1. ollama, built-in python 2. streamlit
<i>Similarity Scoring</i>	<ul style="list-style-type: none"> ● Calculate the cosine similarity between the job description embedding and each CV embedding using cosine similarity or another distance metric. 	<ol style="list-style-type: none"> 1. Scikit-learn 2. numpy

	<ul style="list-style-type: none"> Rank the CVs based on their similarity scores to the job description. 	
<i>Thresholding</i>	<ul style="list-style-type: none"> Determine a threshold score to filter out CVs that do not meet the minimum similarity requirement. Create a shortlist of CVs that exceed the threshold. 	Built-in python
<i>Review and Refinement</i>	<ul style="list-style-type: none"> Optionally, have a human reviewer assess the shortlisted CVs to ensure they meet the qualitative aspects of the job description. Incorporate feedback from the review process to refine the model or the scoring criteria. 	
<i>Output Generation</i>	<ul style="list-style-type: none"> Compile the final shortlist of CVs and present them in a user-friendly format. 	streamlit
<i>Iterate and Improve</i>	<ul style="list-style-type: none"> Continuously gather feedback and improve the pipeline based on new data and outcomes. 	ollama

5.3 AI Model Details

5.3.1 Model Type

Large Language Model (LLM) from Ollama, more specifically the model qwen was used in the tool's evaluation processes.

5.3.2 Prompts

Prompts used for the evaluation process of the resumes:

1. Feature Extraction and output tailoring prompt:

```

ollama.py

RESUME_PROMPT = """
    Extract the skills, experience, and qualifications from the following Resume text with the output
    is precise and abbreviated.
    Format the output as:
        - Email Address: [email address of the applicant]
        - Technical Skills: [list of technical skills]
        - Soft Skills: [list of soft skills]
        - Experience: [list of experience requirements]
        - Qualifications: [list of qualifications]
        - Score: [integer value of 0 to 100 for the resume compared to resume objects listed below
        based on its level of match to the job description and criteria listed below]
        - Justification of given score: [bullet point list of reasons why the resume got
        the previous score value]
    and Evaluate this resume against the job description: {job_text} and the following criteria:
        - Required Skills: {required_skills}
        - Minimum Experience: {min_experience} years
        - Required Education: {education_level}

    Resume:
    {{text}}
    All Resumes:
    {{resume_objects}}
    """

```

2. Shortlisting prompt:

```

ollama.py

RESUME_PROMPT2 = """
    Execute the following based on the text provided below:
    - Shortlist the resumes into the top 5 resumes based on the score.

    Text:
    {prev_data}
    """

```

3. Analysis and best candidate conclusion prompt:

```
ollama.py

RESUME_PROMPT3 = """
Execute the following based on the text of the resume shortlist provided below:
- Analyze each resumes general strengths and weaknesses regardless of the job description.
- Analyze each resumes strengths and weaknesses considering the job description{job_text}.

Format the output as shown for each applicant's resume:
- General Strengths: [list of strengths regardless of the job description]
- General Weaknesses: [list of weaknesses regardless of the job description]
- Strengths: [list of strengths]
- Weaknesses: [list of weaknesses]

Then Conclude the best resume based on the previous data
and output it in the format:
- Best Resume: [Name of the applicant]
- [why this was concluded as best resume]
- Email Address: [email address of the best applicant]

Text:
{prev_data}
"""
```

5.3.3 Inference

- Extracts structured data from unstructured resume text.
- Matches skills, job experience, and qualifications against a given job description and user added criteria.
- Returns a shortlist of the top 5 candidates, analyzes their general strengths and weaknesses regardless of the job description, then regarding the job description.
- Returns the concluded top candidate based on the previous rankings and analysis.

5.3.4 Tested LLM models

Table(7): Tested LLM models

Model name in Ollama	Model name in Openrouter
<i>llama3.1</i>	"meta-llama/llama-3.1-8b-instruct:free"
<i>llama3.2</i>	"meta-llama/llama-3.2-1b-instruct:free"
<i>llama3.2:3b</i>	"meta-llama/llama-3.2-3b-instruct"
<i>phi3</i>	"microsoft/phi-3-mini-128k-instruct:free"

<i>gemma2:2b</i>	"google/gemma-2-9b-it:free"
<i>mistral</i>	"mistralai/mistral-7b-instruct:free"
<i>qwen</i>	"qwen/qwq-32b:free"

5.3.5 LLM models' comparison

llama3.2:3b vs. qwen shortlisting and analysis outputs:

Table(8): LLM model comparison

Llama 3.2:3b	Qwen
<p>Generating shortlist of top 5 applicants... Based on the provided text, I will shortlist the top 5 resumes based on their scores, along with their justifications.</p> <p>Top 5 Resumes:</p> <ol style="list-style-type: none"> Resume 10: 75 Justification: The candidate has a Bachelor's degree in Mechanical Engineering, experience in industrial projects, and skills in various software tools relevant to the job. However, lacks experience in machine learning and deep learning. Resume 16: 60 Justification: The resume has some relevant technical skills and experience, but lacks specific experience in machine learning and deep learning. Education qualifications are also not directly related to the field of AI. Resume 12: 40 Justification: The candidate lacks relevant technical skills (Python, Machine Learning) and experience (3 years) for the AI Developer position. Limited technical skills and no direct education or training in AI or computer science. Resume 6: 40 Justification: The candidate lacks direct experience in AI development, limited education in relevant fields (Mechanical Engineering), and skills mentioned are more related to mechanical design and industry related to manufacturing. Resume 14: 20 Justification: The candidate lacks experience in AI development and machine learning, no relevant technical skills listed (only basic CAD and office 	<p>Generating shortlist of top 5 applicants... Based on the provided scores and cosine similarities, all resumes have the same Score: 20, indicating they do not meet the core requirements for the AI Developer role (e.g., Python, Machine Learning, AI/ML experience). Since the scores are identical, the cosine similarity (a measure of alignment with the job description) is used as a tiebreaker to rank them. Here are the top 5 resumes sorted by descending cosine similarity:</p> <p>Top 5 Resumes</p> <ol style="list-style-type: none"> Resume 16 <ul style="list-style-type: none"> Cosine Similarity: 0.6045 Reason: Highest cosine similarity among all, though still low (60.45% alignment with the job). Resume 12 <ul style="list-style-type: none"> Cosine Similarity: 0.5925 Reason: Slightly lower alignment but better than most mechanical/CAD-focused resumes. Resume 14 <ul style="list-style-type: none"> Cosine Similarity: 0.5817 Reason: Marginally better alignment due to project planning experience, though unrelated to AI/ML. Resume 7 <ul style="list-style-type: none"> Cosine Similarity: 0.5553 Reason: Moderate alignment due to leadership and technical skills in engineering. Resume 15 <ul style="list-style-type: none"> Cosine Similarity: 0.5533 Reason: Lowest in the top 5 but slightly better than the strictly mechanical resumes (Resumes 6, 10). <p>Notes:</p> <ul style="list-style-type: none"> All resumes are underqualified for the AI Developer role due to missing Python, Machine Learning, and AI/ML experience.

software), and limited education in mechanical engineering (60% in B.E.).

Analyzing shortlist's strengths and weaknesses, and concluding the best applicant...

Here are the analyses of each resume:

Resume 10

- General Strengths: Bachelor's degree in Mechanical Engineering, experience in industrial projects, skills in various software tools relevant to the job.
- General Weaknesses: None mentioned.
- Strengths: Relevant education and experience, technical skills.
- Weaknesses: Lack of experience in machine learning and deep learning.

Resume 16

- General Strengths: Relevant technical skills, experience in industry.
- General Weaknesses: Limited education qualifications directly related to AI.
- Strengths: Technical skills, industry experience.
- Weaknesses: Limited experience in machine learning and deep learning, lack of education in AI.

Resume 12

- General Strengths: Experience, technical skills mentioned (Python).
- General Weaknesses: Limited technical skills, lack of direct education or training in AI or computer science.
- Strengths: Experience, technical skills.
- Weaknesses: Limited technical skills, lack of education in AI.

Resume 6

- General Strengths: Mechanical design experience, skills in manufacturing industry.
- General Weaknesses: Limited education in relevant fields (Mechanical Engineering).
- Strengths: Relevant industry experience, mechanical design skills.
- Weaknesses: Limited education in AI, lack of experience in AI development.

Resume 14

- General Strengths: None mentioned.
- General Weaknesses: Limited education in mechanical engineering (60%), lack of experience in AI development and machine learning.
- Strengths: None mentioned.
- Weaknesses: Limited education in mechanical engineering, lack of experience in AI development and machine learning.

Based on the analysis, the top 5 resumes can be ranked as follows:

1. Resume 10:
This resume has the most relevant education and experience for the AI Developer position, despite lacking experience in machine learning and deep learning. However, this is a significant weakness.
- 2.
- 3.

- Cosine similarity is prioritized here as a secondary metric to differentiate between identical scores.
- Resumes 6 and 10 (cosine ~0.549) are excluded from the top 5 due to lower alignment.

Let me know if you need further adjustments!

Analyzing shortlist's strengths and weaknesses, and concluding the best applicant...

Resume 16

- General Strengths: [Possesses better keyword alignment with technical terms (e.g., "data analysis," "project planning," "leadership"), structured format, and relevant industry exposure (e.g., engineering, software roles)]
- General Weaknesses: [Lacks core technical skills like Python, machine learning, or AI/ML experience; insufficient depth in AI-specific roles/projects; potential gaps in education/relevant certifications]
- Strengths: [Highest cosine similarity (0.6045), indicating closer alignment with AI-related keywords compared to others (e.g., "natural language processing," "model optimization," or "data analysis projects")]
- Weaknesses: [No explicit mention of Python, machine learning, or AI/ML experience; projects/roles are tangential to AI development; insufficient technical depth for the role]

Resume 12

- General Strengths: [Better focus on technical roles over mechanical/CAD fields; structured communication of skills]
- General Weaknesses: [Limited AI/ML experience; gaps in core technical proficiencies (e.g., no Python, machine learning frameworks)]
- Strengths: [Moderate alignment with technical terms like "data analysis" or "collaboration," suggesting potential for cross-functional teamwork]
- Weaknesses: [No direct experience in AI development, NLP, or machine learning algorithms; projects unrelated to AI]

Resume 14

- General Strengths: [Project planning experience adds value to team collaboration; clear structure in outlining responsibilities]
- General Weaknesses: [Irrelevant project planning experience for AI roles; no technical AI/ML skills]
- Strengths: [Alignment with "project design" and "team collaboration" keywords, which partially match the job's collaborative nature]
- Weaknesses: [No machine learning, Python, or AI-related projects; weak technical alignment with core responsibilities]

<p>Resume 16:</p> <p>This resume has some relevant technical skills and experience, but lacks specific experience in machine learning and deep learning.</p> <p>Resume 12:</p> <p>This resume has some technical skills and experience, but lacks direct education or training in AI or computer science.</p> <p>Resume 6:</p> <p>This resume has some relevant industry experience, but lacks education in AI and experience in AI development.</p> <p>Resume 14:</p> <p>This resume has no relevant education or experience in AI development and machine learning.</p> <p>Based on the ranking, Resume 10 is the best resume. Although it lacks experience in machine learning and deep learning, it has the most relevant education and experience for the AI Developer position.</p> <p>Best Resume: Resume 10</p> <p>Why this was concluded as best resume: Despite lacking experience in machine learning and deep learning, Resume 10 has the most relevant education and experience for the AI Developer position.</p> <p>Email Address: Not provided</p>	<p>Resume 7</p> <ul style="list-style-type: none"> General Strengths: [Leadership and engineering experience demonstrate management potential; technical skills in engineering fields] General Weaknesses: [No AI/ML or Python expertise; roles focused on non-AI technical domains] Strengths: [Alignment with "leadership" and "technical skills" keywords, which are secondary to the role's core requirements] Weaknesses: [No hands-on experience in AI development, model optimization, or NLP systems] <p>Resume 15</p> <ul style="list-style-type: none"> General Strengths: [Moderate technical alignment over strictly mechanical resumes; clear communication of skills] General Weaknesses: [No AI/ML experience; limited technical depth in data science or machine learning] Strengths: [Slightly better alignment with "data analysis" or "model testing" keywords] Weaknesses: [No Python, machine learning frameworks, or AI project examples; insufficient technical relevance] <p>Conclusion</p> <p>Best Resume: Resume 16</p> <ul style="list-style-type: none"> Why: Despite all resumes being underqualified, Resume 16 has the highest cosine similarity (0.6045), indicating closer alignment with AI-related keywords (e.g., "data analysis," "model optimization," or "natural language processing") compared to others. Its structure and keyword usage make it the most promising candidate for follow-up, even if it lacks explicit ML/AI experience. Email Address: [Not provided in the data; would require access to the full resume details to retrieve the email]. <p>Note: All resumes are underqualified for the role. This conclusion prioritizes alignment with the job description's keywords over core technical qualifications. Further screening would be needed to assess practical skills and potential for upskilling.</p>
---	---

Table(8) shows the candidate shortlisting and thorough strength and weakness analysis then best candidate conclusion as it demonstrates that:

- llama3.2 uses its own generated score in the shortlisting process while qwen uses system calculated cosine similarity in addition to its own generated scores.

- qwen provides additional notes and evaluations to help with the decision making process while llama3.2 only does as specified in the prompt and not that proficiently.
- qwen provides a very detailed (strength, weakness) analysis with thorough examination of each resume with comprehensive knowledge of the resume's semantics, while llama3.2 performs superficial examination and writes none where it can't deduce a strength or a weakness.

Chapter 6

Implementation and Coding

Chapter 6

Implementation and Coding

6.1 Overview

In this chapter we will briefly describe the key components of the system and how they were implemented.

6.2 Programming Language and Tools

Python was chosen as the primary language due to its simplicity, libraries and strong support for AI and data processing.

6.3 Libraries and Frameworks

- Streamlit: For building the user interface
- Transformers: For generating embeddings using a pre-trained model
- NumPy and scikit-learn: For numerical computations and similarity calculations
- Requests: For making API calls to OpenRouter

6.4 Development Environment

- IDE : we used vsCode and pycharm
- Version control: Git and Github used for version control

6.5 Key Components and Implementation

In this section we will describe the implementation of the system.

a. File processing:

- The system supports multiple file formats(PDF, DOCX)
- Libraries like pdfplumber(for pdf), docx(for docx)
- Code Example:

```
fileProcessing.py

def extract_text_from_pdf(pdf_file):
    text = ""
    with pdfplumber.open(pdf_file) as pdf:
        for page in pdf.pages:
            text += page.extract_text() + "\n"
    return text.strip()
```

b. Feature Extraction Using OpenRouter API:

- Implementation: The OpenRouter API was used to extract features from resumes and job descriptions.
- The API was called using the requests library, and the response was parsed to extract structured data.
- Code example:

```
ollama.py

load_dotenv()
API_URL = os.getenv("API_URL")
OPENROUTER_API_KEY = os.getenv("OPENROUTER_API_KEY")
OLLAMA_MODEL = os.getenv("OLLAMA_MODEL")

def make_request(prompt):
    try:
        # OpenRouter API configuration
        headers = {"Authorization": f"Bearer {OPENROUTER_API_KEY}", "Content-Type": "application/json"}
        payload = {
            "model": OLLAMA_MODEL, # model name as shown in openrouter website
            "messages": [
                {"role": "system", "content": "You are a helpful assistant."},
                {"role": "user", "content": prompt},
            ],
        }

        # Make the API request
        response = requests.post(API_URL, headers=headers, json=payload)
        response.raise_for_status() # Raise an error for bad status codes

        # Debug: Print the response and status code
        print("API Response Status Code:", response.status_code)
        print("API Response Content:", response.text)

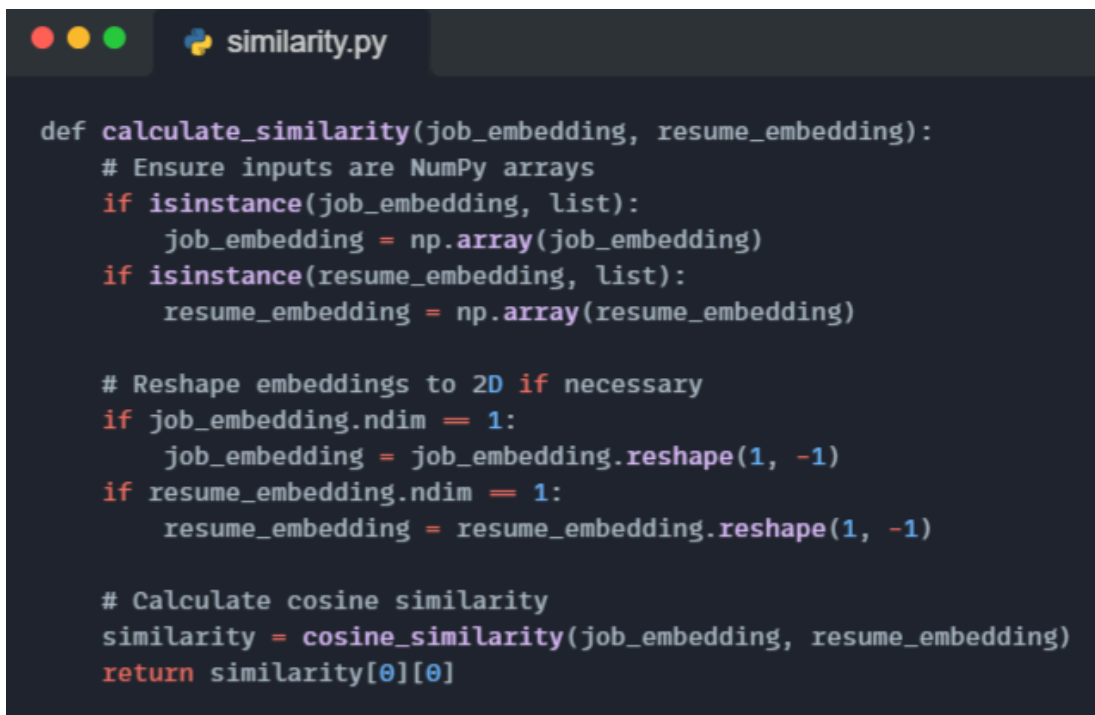
        # Extract the response content
        result = response.json()
        return (
            result["choices"][0]["message"]["content"] if "choices" in result else None
        )
    except Exception as e:
        print(f"Error: {e}")
        return f"{e}"
```

c. Embedding Generation:

- Implementation: the transformers library was used to load a pre-trained model nomic-ai/nomic-embed-text-v1 and generate embeddings
- Mean pooling was applied to convert token-level embeddings into a single vector for the entire text

d. Similarity Calculation:

- Cosine similarity was calculated using scikit-learn to compare job and resume embeddings.
- The results were normalized to scale of 0 to 1
- Code Example:



```
def calculate_similarity(job_embedding, resume_embedding):  
    # Ensure inputs are NumPy arrays  
    if isinstance(job_embedding, list):  
        job_embedding = np.array(job_embedding)  
    if isinstance(resume_embedding, list):  
        resume_embedding = np.array(resume_embedding)  
  
    # Reshape embeddings to 2D if necessary  
    if job_embedding.ndim == 1:  
        job_embedding = job_embedding.reshape(1, -1)  
    if resume_embedding.ndim == 1:  
        resume_embedding = resume_embedding.reshape(1, -1)  
  
    # Calculate cosine similarity  
    similarity = cosine_similarity(job_embedding, resume_embedding)  
    return similarity[0][0]
```

e. Streamlit User Interface:

- Implementation: The Streamlit framework was used to create a web-based interface.
- Code Example:

```
def main():  
  
    sl.title(":memo: Resume Matcher")  
    # Files uploader  
    job_text = sl.text_input("Job Description")  
    resume_files = sl.file_uploader(  
        "Upload Resumes", type=["pdf", "docx"], accept_multiple_files=True  
    )  
  
    # User-defined criteria  
    sl.write("### Evaluation Criteria")  
    required_skills = sl.text_input(  
        "Required skills (comma-separated)", "Python, Machine Learning"  
    )  
    min_experience = sl.number_input(  
        "Minimum years of experience required", min_value=0, value=3  
    )  
    education_level = sl.selectbox("Required Education Level", ["Bachelor's Degree", "Master's Degree",  
        "PHD Degree", "None"])
```

- User Interface:

The screenshot shows a web application titled "Resume Matcher" with a dark theme. It features several input fields and a submit button. The "Job Description" field is a simple text input. The "Upload Resumes" section includes a drag-and-drop area with a cloud icon and a "Browse files" button. The "Evaluation Criteria" section contains three inputs: "Required skills (comma-separated)" with the text "Python, Machine Learning", "Minimum years of experience required" with a numeric value of 3 and minus/plus buttons, and "Required Education Level" with a dropdown menu showing "Bachelor's Degree". A "Submit" button is located at the bottom left.

6.6 Conclusion

The implementation phase involved translating the system design into a functional application using Python and various libraries. By following coding standards, addressing challenges, and testing rigorously, the Resume Matcher application was successfully developed. The modular design and documentation ensure that the codebase is maintainable and scalable for future enhancements.

Chapter 7

Testing, Evaluation, and Deployment

Chapter 7

Testing, Evaluation, and Deployment

7.1 Overview

Ensuring the reliability, correctness, and efficiency in Processing resumes, generating embeddings, and ranking candidates based on job relevance is the primary goal of this stage.

The testing stage is conducted using Pytest framework and NumPy library. Multiple components are tested in order to verify their functionality, such as, document parsing, embedding generation, similarity calculation, and threshold filtering.

7.2 Testing Strategy

7.2.1 Unit testing

The unit testing focuses on verifying the correctness and functionality of the following individual components:

1. *File parsing*: The purpose of the test is to verify the accurate extraction of PDF and DOCX files.
2. *Embedding generation*: The test is carried out to ensure that the text is transformed into accurate numerical representations.
3. *Similarity calculation*: The purpose of the test is to verify that similarity scores between job descriptions and resumes are accurately calculated.
4. *Threshold filtering*: The purpose of the test is to make sure that resumes with similarities below a specific threshold are correctly eliminated.

Figure 6 is a snippet of the unit test code. This part involves accurately extracting text from PDF and DOCX files.

Figure (6): Testing text extraction from PDF and DOCX files

```
def test_pdfFile_parsing():
    print("test_pdf_parsing")
    path = "8.pdf"
    if not os.path.exists(path):
        pytest.fail(f"Missing test file: {path}")
    text = extract_text_from_pdf(path)
    print(f"Parsed text: {text}")
    assert "HTML" in text, f"Expected 'Python' in parsed text. Got: {text}"

def test_docxFile_parsing():
    path = "11.docx"
    if not os.path.exists(path):
        pytest.fail(f"Missing test file: {path}")
    text = extract_text_from_docx(path)
    assert "Mechanical" in text, f"Expected 'HTML' in parsed text. Got: {text}"
```

7.2.2 Integration Testing

Integration testing is carried out in order to confirm that the interaction between different modules is smooth. Key integrations that were tested include:

- Combining embedding generation with text extraction from resumes.
- Confirming that similarity calculations produces the anticipated rankings.
- Ensuring that result resumes are filtered and meet the threshold requirements.

Sample of the integration test code:

Figure (7): Integration test for similarity calculation

```
def test_similarity():
    path = "11.docx"
    resume_text = extract_text_from_docx(path)
    generator = EmbeddingGenerator()
    > text = "" ...
    job_embedding = generator.generate(resume_text)
    resume_embedding = generator.generate(text)
    similarity = calculate_similarity(job_embedding, resume_embedding)
    # Assert that the variable is a float
    assert isinstance(similarity, numpy.float32), "The variable is not a float!"
    # If the assertion passes, the program continues
    print(f"The similarity variable has a float value of: {similarity}")
```

7.2.3 User Acceptance Testing (UAT)

The user acceptance testing is conducted through testing real resumes and job descriptions. The objective is to determine whether the tool can filter and

rank the candidates accurately. Since we were unable to have professional HR people and recruiters test our tool, we conducted a manual examination of the ranked resumes. Although the tool successfully filtered and ranked candidates according to similarity scores, further validation from industry professionals would strengthen confidence in its real-world applicability.

7.3 Deployment

It's a very easy and simple step in our project as it was performed using streamlit's deploy option which appears when running the app.

7.3.1 Deployment requirements on streamlit

- You have a github account.
- You have your app as a repository on github.
- You are a contributor with the role of admin in your github app.
- You add the OpenRouter API key as a secret in your streamlit app's additional settings.

The link to our deployed app: <https://cv-shortlist-assistant.streamlit.app/>

7.4 Test Case and Execution

This section shows detailed tests performed throughout the development of this app:

1. Test Case: PDF File Parsing

Test Case ID: TC001

Test Description: Verify that the function `extract_text_from_pdf` accurately extracts text from a sample PDF file.

Preconditions: Ensure the sample PDF file (8.pdf) is located in the testing directory.

Test Steps:

1. Invoke the `extract_text_from_pdf` function with the path to 8.pdf.
2. Capture the returned text output.
3. Check if the string "HTML" is present in the extracted text.

Test Data: "testing/8.pdf"

Expected Result: The extracted text should contain the keyword "HTML".

Actual Result: The extracted text contains the keyword "HTML".

Status: Pass

2. Test Case: DOCX File Parsing

Test Case ID: TC002

Test Description: Verify that the function `extract_text_from_docx` accurately extracts text from a sample DOCX file.

Preconditions: Ensure the sample DOCX file (11.docx) is located in the testing directory.

Test Steps:

1. Invoke the `extract_text_from_docx` function with the path to 11.docx file.
2. Capture the returned text output.
3. Check if the string "Mechanical" is present in the extracted text.

Test Data: "testing/11.docx"

Expected Result: The extracted text should contain the keyword "Mechanical".

Actual Result: The extracted text contains the keyword "Mechanical".

Status: Pass

3. Test Case: Embedding Generation

Test Case ID: TC003

Test Description: Confirm that the `EmbeddingGenerator` produces embeddings of the correct type and dimensions.

Preconditions: Initialize the `EmbeddingGenerator`.

Test Steps:

1. Generate an embedding for the text "Python developer with NLP experience".
2. Assert that the output is a NumPy array of shape (768).

Test Data: "Python developer with NLP experience"

Expected Result: The embedding must be a NumPy array with shape (768).

Actual Result: The embedding is a NumPy array with shape (768).

Status: Pass

4. Test Case: Similarity Calculation

Test Case ID: TC004

Test Description: Validate that the system accurately calculates cosine similarity between job descriptions and resumes, ensuring the score is a numpy floating-point number.

Preconditions:

- Extract text from the sample DOCX file testing/11.docx.
- Generate embeddings for both the resume text and a predefined job description.

Test Steps:

1. Calculate the cosine similarity between the two embeddings.
2. Assert that the similarity score is a numpy float.

Test Data: “testing/11.docx”, job text

Expected Result: The similarity score must be a floating-point number.

Actual Result: The similarity score is a floating-point number.

Status: Pass

5. Test Case: Threshold Filtering

Test Case ID: TC005

Test Description: This test filters resumes by a predefined similarity threshold (e.g., 0.544), ensuring only those meeting or exceeding the threshold are retained.

Preconditions:

- Extract text from multiple resume files.
- Generate embeddings for each resume and the job description.
- Calculate similarity scores for each resume.

Test Steps:

1. Filter resumes based on the threshold of 0.544.
2. Assert that the similarity score is a numpy float.

Test Data: “testing/11.docx”, “testing/8.pdf”, “testing/5.docx”, job text.

Expected Result: Only resumes with a similarity score ≥ 0.544 must be retained.

Actual Result: Only resumes with a similarity score ≥ 0.544 are retained.

Status: Pass

6. Test Case: External API Request

Test Case ID: TC006

Test Description: Verify that the make_request function successfully sends a request and receives a response.

Preconditions: None

Test Steps:

1. Invoke the make_request function with the prompt "What can you tell me about smurfs?".
2. Capture the response.
3. Verify that the response is not empty.

Test Data: Prompt: "What can you tell me about smurfs?".

Expected Result: The response should not be empty.

Actual Result: The response is not empty.

Status: Pass

7. Test Case: Feature Extraction

Test Case ID: TC007

Test Description: Verify that the extract_info function successfully extracts key features of each resume.

Preconditions:

- Extract text from multiple resume files.
- Generate embeddings for each resume and the job description.
- Calculate similarity scores for each resume.
- Store all resume objects in a list and filter them by a threshold.

Test Steps:

1. Use the extract_info function as a parameter to the make_request function as it returns a prompt containing all required data.
2. Store each resume's key features and its cosine similarity in the extracted features list using a for loop.
3. Assert the extracted features list isn't empty

Test Data: resumes list, resume text, job text, RESUME_PROMPT.

Expected Result: The extracted features list should not be empty.

Actual Result: The extracted features list is not empty.

Status: Pass

8. Test Case: Shortlisting

Test Case ID: TC008

Test Description: Ensure that the shortlist function shortlists resumes into top 5 resumes as specified in the prompt.

Preconditions:

- Get the extracted features list from the previous test.

Test Steps:

1. Use the shortlist function as a parameter to the make_request function as it returns a prompt containing all required data.
2. Store the shortlist in a variable.
3. Assert the shortlist string includes the string "Top 5".

Test Data: RESUMES list, RESUME_PROMPT2.

Expected Result: The shortlist variable should contain the keyword "Top 5".

Actual Result: The shortlist variable contains the keyword "Top 5".

Status: Pass

9. Test Case: Analysis

Test Case ID: TC009

Test Description: Verify that the final_analysis function analyzes the shortlist of resumes' strengths and weaknesses and concludes the best resume as specified in the prompt.

Preconditions:

- Get the shortlist string from the previous test.

Test Steps:

1. Use the final_analysis function as a parameter to the make_request function as it returns a prompt containing all required data.
2. Capture the response.
3. Assert the response includes the string "Best Resume".

Test Data: SHORT string, job text, RESUME_PROMPT3.

Expected Result: The response should contain the keyword "Best Resume".

Actual Result: The response contains the keyword "Best Resume".

Status: Pass

7.5 Results and Observations

In this section, we present the outcomes of the various test functions implemented in our application. Each test was designed to validate specific functionalities, and the results are summarized below:

1. File Parsing Functions

- (test_pdfFile_parsing): The function successfully extracted text from the provided PDF resume and the displayed output was the text content of the file including key terms such as "HTML", indicating accurate text extraction.
- (test_docxFile_parsing): The function accurately extracted text from the DOCX resume, with the presence of the term "Mechanical"(key term in parsed file) confirming successful text extraction.

2. Embedding Generation

- (test_embedding): The embedding generator produced a numpy array with the expected shape of (768,) for the input text "Python developer with NLP experience," confirming correct embedding generation.

3. Similarity Calculation

- (test_similarity): The cosine similarity between the job description embedding and resume embedding was calculated and returned as a numpy float value, confirming the functionality of the similarity calculation process.

4. Threshold Filtering

- (test_threshold): Resumes with similarity scores above the threshold of 0.544 were successfully filtered. Similarity scores above the threshold were returned and a success message, validating the threshold filtering mechanism.

5. External API Request

- (test_request): The function successfully received a response from the OpenRouter API for the query "What can you tell me about smurfs?" and printed the response content, indicating proper API interaction.

6. Resume Feature Extraction

- (test_extraction): Features were successfully extracted from resumes that met the similarity threshold. The extracted info function was compiled without

errors and a success message was returned, confirming the functionality of the feature extraction process.

7. Resume Shortlisting

- (test_shortlisting): The function successfully generated a shortlist of resumes, with the response containing the phrase "Top 5", indicating that the shortlisting process identified the most suitable candidates based on the extracted features list from the previous test.

8. Final Analysis

- (test_analysis): The final analysis function successfully identified the "Best Resume" among the shortlisted candidates, as indicated by the response content. This confirms the integration and effectiveness of the application's overall functionality.

These results demonstrate that each component of the application performs as intended, contributing to the accurate analysis and shortlisting of resumes based on job descriptions. The successful execution of these test functions indicates the reliability and efficiency of the application's core features.

Chapter 8

Conclusions and future works

Chapter 8

Conclusions and future works

8.1 Conclusions

Our project successfully met its objectives by developing a fully functional resume screening tool powered by AI to improve and simplify the recruitment process. Using large language models (LLMs), embedding-based similarity analysis, and automated ranking mechanisms, the system can efficiently analyze resumes, match them with job descriptions, and create a list of the best five candidates supported by strengths and weaknesses of those candidates.

8.2 Future Work

While our project sets a solid foundation, there's room for several enhancements that can improve its efficiency, scalability, and the user experience. Such developments include :

1. Integration with ATS

In order to enhance the adoption of organizations for our tool, a suggestion is to make the tool to be integrated seamlessly with existing ATS platforms that companies use. Such steps will allow HR teams to seamlessly manage candidate applications within their current workflow.

2. Feedback and insights for candidates

By adding an automated feedback mechanism for candidates, trust and transparency will be assured. Suggested feedback could be:

- a. Personalized feedback on their application
- b. Recommendation for skill improvement.
- c. Insights why their CV was rejected with some insights where to improve(in case of rejection)

3. Upgrading the AI models used for screening

An upgrading example would be using multimodal analysis to incorporate cover letters, LinkedIn profiles and even portfolios. Another suggestion is working on the accuracy of the model when screening resumes that are more graphical and non-traditional.

4. Scalability for High-Volume resumes

Optimizing the system to be able to handle thousands of resumes efficiently especially for larger corporations.

5. User tailored criteria

Although our project mainly shortlists the resumes according to the job description and there's not much room for specificity that the employee might want, there's a great scope for advancement and creating tailored filters in the tool that employees would find valuable.

8.3 Final Remarks

Our project showcases the potential of AI-driven automation in recruitment, making hiring faster, fairer, more efficient, and devoid of cost. Although existing tools have yielded satisfactory outcomes, there remains substantial scope for future enhancements as artificial intelligence continues to progress. Our free, open-source initiative sets a solid foundation for advancement for more intelligent, data-informed recruitment solutions.

References

- [1] [Online]. Available: https://en.wikipedia.org/wiki/Artificial_intelligence.
- [2] [Online]. Available: <https://eddy.com/hr-encyclopedia/resume-screening/#:~:text=Resume%20screening%20is%20the%20process,larger%20Deployment%20screening%20proces.> .
- [3] [Online]. Available: <https://www.ibm.com/think/topics/natural-language-processing>.
- [4] [Online]. Available: <https://www.ibm.com/think/topics/machine-learning>.
- [5] [Online]. Available: <https://www.ibm.com/think/topics/large-language-models>.
- [6] [Online]. Available: <https://www.rightcruiter.com/resources/ai-vs-traditional>.
- [7] [Online]. Available: <https://www.indeed.com/hire/c/info/ai-resume-screening>.
- [8] [Online]. Available: <https://www.nanili.ai/blog/how-large-language-models-are-transforming-resume-screening>.
- [9] [Online]. Available: <https://www.jetir.org/papers/JETIR2303510.pdf>.
- [10] [Online]. Available: <https://swooped.co/opt-out-ai-resume-screening/>.
- [11] [Online]. Available: <https://www.finalroundai.com/blog/ai-resume-builders-vs-traditional-tools-a-comprehensive-comparison-of-features-efits#comparing-popular-ai-resume-builders-features-and-pricing>.
- [12] [Online]. Available: <https://resumesranker.com/blogs/ai-resume-screening-explained-how-it-works-and-why-it-matters-in-2025/>.
- [13] [Online]. Available: <https://www.hipeople.io/blog/ai-resume-screening>.
- [14] [Online]. Available: <https://recruitryte.com/blog/ai-vs-traditional-recruitment-advantages-disadvantages/>.
- [15] [Online]. Available: <https://www.aimodels.fyi/papers/arxiv/application-llm-agents-recruitment-novel-framework-res>
- [16] [Online]. Available: <https://bpasjournals.com/library-science/index.php/journal/article/view/4000/3733>.
- [17] [Online]. Available: <https://www.skillpool.tech/>.
- [18] [Online]. Available: <https://hirebeat.co/>.
- [19] [Online]. Available: <https://www.textkernel.com/>.
- [20] [Online]. Available: <https://www.daxtra.com/>.
- [21] [Online]. Available: <https://recruitmentsmart.com/sniper-ai/>.
- [22] [Online]. Available: <https://hubert.ai/>.
- [23] [Online]. Available: <https://www.mosaictrack.com/>.
- [24] [Online]. Available: <https://xor.ai/>.
- [25] [Online]. Available: <https://pesto.tech/>.
- [26] [Online]. Available: <https://www.hirevue.com/>.
- [27] [Online]. Available: <https://www.pymetrics.com/>.
- [28] [Online]. Available: <https://www.hiretual.com/>.
- [29] [Online]. Available: <https://x0pa.com/>.
- [30] [Online]. Available: <https://leoforce.com/>.
- [31] [Online]. Available: <https://www.zoho.com/recruit/>.
- [32] [Online]. Available: <https://www.manatal.com/>.
- [33] [Online]. Available: <https://recruitee.com/>.
- [34] [Online]. Available: <https://www.mya.com/>.
- [35] [Online]. Available: <https://eightfold.ai/>.
- [36] [Online]. Available: <https://www.jobvite.com/>.
- [37] [Online]. Available: <https://beamery.com/>.

[38] [Online]. Available: <https://harver.com/>.