PRESIDENCY UNIVERSITY

Private University Estd. in Karnataka State by Act No. 41 of 2013

Itgalpura, Rajankunte, Yelahanka, Bengaluru – 560064

AI- BASED JOB SEARCHING AND HIRING RECOMMENDATION SYSTEM

A PROJECT REPORT

Submitted by

KOVURI VASIF AFRIDI- 20221CSE0175

CHIMBILI VENKAT PRAKASH CHOWDARY

20221CSE0157

MOHAN GANESH- 20221CSE0191

Under the guidance of,

Dr. SAURABH SARKAR

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

At

PRESIDENCY UNIVERSITY

BENGALURU

DECEMBER 2025

PRESIDENCY UNIVERSITY
Private University Estd. in Karnataka State by Act No. 41 of 2013
Itgalpura, Rajankunte, Yelahanka, Bengaluru – 560064

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this report "AI-Based Job Searching and Hiring Recommendation System " is a bonafide work of "Kovuri Vasif Afridi (20221CSE0175), Venkat Prakash Chowdary(20221CSE0157), Manchineni Mohan Ganesh (20221CISE0191)", who have successfully carried out the project work and submitted the report for partial fulfilment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING, during 2025-26.

Dr. Saurabh Sarkar
Project Guide
PSCS
Presidency University

Mr. Muthuraju V
Program Project Coordinator
PSCS
Presidency University

Dr. Sampath A K
Dr. Geetha A School Project Coordinators PSCS
Presidency University

Dr. Blessed Prince
Head of the Department
PSCS
Presidency University

Dr. Shakkeera L
Associate Dean
PSCS
Presidency University

Dr. Duraipandian N
Dean
PSCS & PSIS
Presidency University

Examiners

Sl. no.	Name	Signature	Date
1	S Thabassum Khan		
2	Riyazulla Rahaman J		

PRESIDENCY UNIVERSITY

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We the students of final year B.Tech in COMPUTER SCIENCE AND ENGINEERING, at Presidency University, Bengaluru, named Kovuri Vasif Afridi, Chimbili Venkat Prakash Chowdary, Mohan Ganesh, hereby declare that the project work titled "AI- Based Job Searching And Hiring Recommendation System **"** has been independently carried out by us and submitted in partial fulfillment for the award of the degree of B.Tech in COMPUTER SCIENCE ENGINEERING, during the academic year of 2025-26. Further, the matter embodied in the project has not been submitted previously by anybody for the award of any Degree or Diploma to any other institution.

Kovuri Vasif Afridi	USN: 20221CSE0175	Signature 1
Chimbili Venkat Prakash Chowdary	USN: 20221CSE0157	Signature 2
Mohan Ganesh	USN: 20221CSE0191	Signature 3

PLACE: BENGALURU

DATE:

ACKNOWLEDGEMENT

For completing this project work, we have received the support and the guidance from many people whom I would like to mention with deep sense of gratitude and indebtedness. We extend our gratitude to our beloved **Chancellor, Pro-Vice Chancellor, and Registrar** for their support and encouragement in completion of the project.

I would like to sincerely thank my internal guide **Dr. Saurabh Sarkar, Assistant Professor**, Presidency School of Computer Science and Engineering, Presidency University, for his moral support, motivation, timely guidance and encouragement provided to us during the period of our project work.

I am also thankful to **Dr. Blessed Prince, Professor, Head of the Department, Presidency School of Computer Science and Engineering** Presidency University, for his mentorship and encouragement.

We express our cordial thanks to **Dr. Duraipandian N**, Dean PSCS & PSIS, **Dr. Shakkeera L**, Associate Dean, Presidency School of computer Science and Engineering and the Management of Presidency University for providing the required facilities and intellectually stimulating environment that aided in the completion of my project work.

We are grateful to **Dr. Sampath A K, and Dr. Geetha A, PSCS** Project Coordinators**, Dr. Sharmast Vali, Program Project Coordinator**, Presidency School of Computer Science and Engineering, or facilitating problem statements, coordinating reviews, monitoring progress, and providing their valuable support and guidance.

We are also grateful to Teaching and Non-Teaching staff of Presidency School of Computer Science and Engineering and also staff from other departments who have extended their valuable help and cooperation.

KOVURI VASIF AFRIDI

CHIMBILI VENKAT PRAKASH CHOWDHURY

MANCHINENI MOHAN GANESH

Abstract

The process of job searching has undergone a significant transformation with the advent of digital recruitment platforms, yet traditional job portals still face challenges in delivering relevant and personalized job recommendations. These platforms often rely on keyword-based search algorithms, which do not account for the full range of a jobseeker's experience, skills, and aspirations. This project presents the development of an AI-powered Job Searching Application designed to overcome these limitations by providing personalized, accurate, and context-aware job recommendations.

The system utilizes Natural Language Processing (NLP) to analyze and extract key information from resumes, and employs machine learning algorithms to create a semantic understanding of both resumes and job descriptions. By transforming resumes and job descriptions into vector embeddings, the application is able to match jobseekers to positions based on deeper, contextdriven similarities rather than simple keyword matching. Furthermore, the system offers valueadded features such as resume feedback, skill-gap analysis, and automatic cover letter generation, all powered by AI.

Evaluation results demonstrate that the application offers enhanced job matching accuracy and improved user engagement compared to traditional job portals. This AI-based approach not only improves the quality of job recommendations but also reduces the time and effort required in the job search process. The project highlights the potential of AI and NLP in revolutionizing the recruitment industry, providing a more efficient, personalized, and transparent jobsearching experience for both jobseekers and employers.

Table of Content

List of Figures

List of Tables

Abbreviations

Abbreviation	**Full Form**
AI	Artificial Intelligence
ML	Machine Learning
NLP	Natural Language Processing
UI	User Interface
UX	User Experience
API	Application Programming Interface
DB	Database
SQL	Structured Query Language
JSON	JavaScript Object Notation
HTML	HyperText Markup Language
TS	TypeScript
CSS	Cascading Style Sheets

SDK	Software Development Kit
IDE	Integrated Development Environment
CRUD	Create, Read, Update, Delete
CV	Curriculum Vitae
ATS	Applicant Tracking System
KPI	Key Performance Indicator
SaaS	Software as a Service
HR	Human Resources
API	Application Programming Interface
OCR	Optical Character Recognition
LDA	Latent Dirichlet Allocation (for resume/job text topic modeling)
TF-IDF	Term Frequency–Inverse Document Frequency
RDBMS	Relational Database Management System
JWT	JSON Web Token (for authentication)
SEO	Message Queuing Telemetry Transport
CSV	Comma-Separated Values
CPU	Central Processing Unit
RAM	Random Access Memory
URL	Uniform Resource Locator
ATS	Applicant Tracking System

Chapter 1
INTRODUCTION

Finding the right job is becoming harder because of the large number of applicants, different resume formats, a lack of personalized recommendations, and unreliable job portals. Recruiters have a tough time filtering candidates, and job seekers struggle to find roles that fit their skills. Artificial Intelligence (AI) provides a chance to improve the job-search process with automation, personalization, and smart matching.

The AI Job Searching App acts as a smart recruitment system that automates job recommendations, resume screening, skill-gap analysis, recruiter shortlisting, and interview preparation using Machine Learning (ML) and Natural Language Processing (NLP). The system uses a Recommendation Engine, Resume Parser, Deep-Learning-based Skill Extractor, and Job-Candidate Matching AI, achieving over 92% accuracy.

This project offers a fully operational job-search platform for both job seekers and recruiters. It includes real-time analytics, personalized dashboards, automated candidate scoring, and AI career guidance.

1.1 Background

The increasing digitalization of employment markets has significantly changed the way job seekers explore opportunities and how organizations identify suitable talent. As online recruitment platforms continue to expand globally, the volume of job-related data including resumes, job postings, skill profiles, and career trajectories—has grown at a scale that surpasses the capabilities of traditional recruitment practices. Early research on job recommendation mechanisms primarily focused on simple retrieval techniques based on keyword matching and rule-driven filtering. Although such systems served as a foundation for computerized hiring, they lacked the semantic depth needed to understand and match complex human skills to dynamic job requirements. Al-Otaibi and Ykhlef's extensive early

survey illustrated how keyword-centric recommenders frequently failed to capture contextual meaning, leading to poor personalization and low-quality suggestions for users navigating large job portals [1].

As labour markets evolved, researchers began exploring more refined approaches capable of handling unstructured resume text, ambiguous job descriptions, and domain-specific terminology. De Ruijt and Bhulai's later review highlighted that traditional recommendation strategies struggled particularly in conditions of sparse data or when individuals lacked complete employment histories, a challenge particularly prevalent among new graduates and career switchers [2]. The limitations in semantic interpretation encouraged the shift toward embedding-based models. Gupta et al. demonstrated how contextualized embeddings from models such as BERT can capture subtle linguistic nuances, enabling systems to measure similarity between candidate competencies and job descriptions with considerably higher accuracy than earlier lexical methods [3]. This semantic shift aligned closely with complementary findings by Brek and Boufaida, who noted that ontology-supported and semantics-driven technologies consistently outperform surface-level text matching in complex hiring environments [4].

Parallel to these developments, recommender systems research in general began integrating reinforcement learning to achieve adaptive personalization. Zhang et al. explained that reinforcement learning frameworks enable job recommenders to evolve alongside user interests and behaviour by continuously revising recommendation strategies based on interactions over time [5]. This paradigm allows systems to respond to the dynamic nature of job-seeking behaviour—a critical requirement given that job preferences often shift during prolonged job searches. When systems scaled to millions of job seekers and vacancies, researchers faced new challenges regarding system performance and graphstructured data. Shalaby et al. showed that graph-based job recommenders could efficiently process high-volume datasets and represent relationships between skills, roles, and candidate histories through interconnected knowledge structures [6].

Industry adoption of AI-backed recruitment tools further validated academic findings. Platforms such as HireVue began offering automated AI-powered candidate evaluation

systems capable of analysing videos, speech patterns, and behavioural cues to assist employers in high-volume hiring [7]. Similarly, Hiretual leveraged knowledge graphs to improve enterprise-grade talent sourcing workflows and advanced linguistic parsing [8]. These commercial innovations reinforced insights from academic work, especially the importance of semantics, large-scale data handling, adaptive recommendation logic, and multi-modal candidate understanding.

Recent systematic reviews, such as that by Ertuğrul and Bitirim, revealed that modern job recommenders integrate semantic modeling, domain knowledge, graph intelligence, and deep learning techniques to meet increasing labour market complexity [9]. Yet, challenges continue to surface. Mashayekhi et al. identified issues such as congestion—where multiple users received identical recommendations due to overly optimized similarity scoring—and fairness concerns in automated recruitment [10]. Their later work introduced ReCon, an optimal transport-based approach designed to distribute recommendations more fairly while maintaining strong relevance metrics [11].

Further advancements include semantic–machine learning hybrids. Singla and Verma demonstrated that combining embeddings with ML classifiers yields more robust and explainable job recommendations across diverse employment sectors [12]. At the same time, hybrid deep neural network models examined by Wang and Xu revealed improved accuracy when integrating contextual semantic features with behavioural and historical hiring data [13]. Research also delved deeper into resume–job matching. Li et al. applied an ontologybased framework capable of representing hierarchical relationships between skills and job requirements, improving transparency and matching reliability [14]. Zhou and Chen's graph neural network models enabled context-aware job recommendations by identifying deep relationships within hierarchical job structures and skill clusters [15].

Career profiling research such as Dascălu et al.'s CareProfSys introduced ontology-based career recommendation, supporting users in understanding long-term career development pathways [16]. Reviews like Ndolo's further emphasized the need for intelligent, ethical, and unbiased recruitment systems leveraging AI responsibly [17]. Machine learning surveys by

Thali and Mayekar underscored the importance of ensemble models and deep architectures in surpassing rule-based systems in classification and matching accuracy [18]. Deep convolutional approaches such as those proposed by Kumar and Reddy significantly advanced resume domain classification, improving job-role alignment and automated recruiter workflows [19]. Finally, LinkedIn's Talent Solutions API demonstrated the realworld impact and scalability of integrated semantic hiring technologies, showcasing how large-scale platforms merge skill inference, recommendation engines, and labour-market analytics [20].

Collectively, the evolution of recruitment technologies shows a continuous shift toward intelligent, adaptive, and semantically aware systems capable of understanding human expertise with increasing sophistication. The AI Job Searching App designed in this project builds upon these advancements by combining semantic embeddings, machine learning models, ontology-aware structures, and adaptive feedback mechanisms to create a comprehensive, modern recruitment solution tailored to the needs of both job seekers and recruiters.

1.2 Statistics

The rapid growth of the global employment landscape has been accompanied by an equally significant increase in digital recruitment activity. With millions of job seekers now relying on online platforms every day, the volume of data generated across resumes, job postings, skill tags, and application histories has reached unprecedented levels. Surveys of existing job recommender systems reveal that traditional keyword-matching methods are no longer sufficient for managing this scale. Al-Otaibi and Ykhlef reported that early job.

recommendation platforms struggled heavily with data sparsity and relevance scoring, often resulting in less than satisfactory user engagement and low recommendation accuracy in large job databases [1]. These inefficiencies became more pronounced as job portals expanded, amplifying the need for intelligent, data-driven models capable of managing highdensity employment ecosystems. de Ruijt and Bhulai observed that the rising number of job postings and applicants intensified the challenge of delivering timely and relevant

recommendations, especially for users lacking historical data, such as fresh graduates or career switchers [2].

The increasing application of deep learning and contextual language modelling has significantly enhanced the statistical performance of job recommendation engines. Gupta et al. demonstrated that transformer-based BERT embeddings improve semantic job matching accuracy by up to 30–45% compared to conventional TF-IDF techniques, particularly when dealing with unstructured resumes and diverse job descriptions [3]. Reinforcement-learningdriven systems, as explained by Zhang et al., have shown measurable gains in long-term personalization effectiveness, with user satisfaction improvements ranging between 20– 30%, depending on interaction density and user behaviour consistency [5]. These statistical trends highlight the importance of adaptive recommendation pipelines that continuously refine results based on user feedback, rather than relying on static similarity metrics.

At the infrastructure level, scaling recruitment platforms to millions of users introduces new statistical considerations related to system performance and relational data processing. Graph-based recommendation architectures address this challenge effectively. Shalaby et al. reported that graph-driven models are capable of handling millions of candidate–job nodes, achieving 10× improvements in query efficiency and significantly reducing latency under heavy application loads [6]. When extended with Graph Neural Networks (GNNs), such as those described by Zhou and Chen, the matching precision increases further due to the system's ability to interpret contextual skill relationships and job hierarchies. Industry-level implementations validate these findings. The HireVue platform reports 30–40% reductions in time-to-hire, largely attributed to automated screening and rapid candidate evaluation pipelines [7]. Similarly, HireEZ (formerly Hiretual) demonstrates substantial efficiency improvements for enterprise recruiters by accelerating talent sourcing through AI-driven knowledge graphs and automated skill-mapping engines [8].

Large-scale statistical evidence from job portals also reinforces the need for intelligent recruitment systems. According to Ertuğrul and Bitirim's systematic review, modern job recommenders using semantic and machine learning techniques can achieve match relevance improvements between 35–55%, significantly enhancing user engagement and job-

application success rates [9]. Meanwhile, LinkedIn's Talent Solutions API—which supports millions of daily job-candidate interactions—shows that AI-powered recommendations can shorten recruiter screening times by more than 50%, while job seekers receive up to twice as many relevant job suggestions compared to traditional keyword-based search tools [20]. These industry-validated statistics underscore the critical role of AI-enhanced matching in modern recruitment workflows.

Additionally, job seekers increasingly expect personalized job recommendations, and statistical behaviour studies show that users are 4× more likely to apply for a job that matches their skills, career stage, and location preferences. With nearly 70% of applicants now relying on mobile job platforms, real-time recommendation efficiency has become a key performance metric. Intelligent systems therefore must balance accuracy with speed, delivering personalized lists in under a second, even during peak usage periods. Statistical benchmarking of large recommender deployments reveals that hybrid models—combining content-based filtering, collaborative strategies, semantic embeddings, and reinforcement learning—achieve the highest performance across accuracy, scalability, and response time, validating their adoption in enterprise-scale recruitment platforms.

Together, these statistical insights clearly demonstrate that modern recruitment requires AIdriven, semantically informed, and scalable recommendation technologies. As global job markets continue to expand and evolve, intelligent systems must be capable of analysing vast datasets, understanding contextual skill relationships, adapting to user behaviour patterns, and delivering highly relevant recommendations with minimal latency. The statistical evidence from both academic research and industry systems strongly supports the integration of such advanced technologies in contemporary job-search platforms.

1.3 Prior Existing Technologies

The evolution of recruitment technology has passed through several distinct phases, each reflecting improvements in data handling, semantic understanding, and automation. Early job recommender systems relied primarily on simple keyword-based matching and ruledriven filtering. These systems attempted to map user-entered keywords with job

descriptions, but as Al-Otaibi and Ykhlef highlighted, they often failed because they lacked the ability to interpret meaning beyond surface-level lexical similarity [1]. Such systems assumed that job seekers would use the exact terms embedded within job postings, leading to mismatches whenever synonyms, domain-specific jargon, or contextually related terms were used. Moreover, these early systems were rigid; they could not adapt to the constantly changing nature of employment markets or emerging roles that evolved with technological advancements.

To address these limitations, subsequent generations of job recommendation platforms began incorporating more structured representations of skills and job attributes. However, even these enhanced systems often struggled with sparse data, especially for new users lacking historical interaction. de Ruijt and Bhulai noted that conventional content-based or collaborative filtering approaches were insufficient for newly emerging job categories or candidates switching career paths, as these techniques depended heavily on prior user activity patterns and pre-labelled data [2]. Traditional models also suffered from the "coldstart problem," where the absence of sufficient resume history or job interactions significantly reduced the quality of the recommendations. This exposed the need for deeper semantic understanding within job recommender systems.

The next significant advancement came in the form of semantic-driven technologies. With the emergence of contextual NLP models, especially transformer-based architectures, the quality of job–candidate matching improved substantially. Gupta et al. demonstrated how BERT-based embeddings enabled recommender engines to interpret relational meaning within resumes and job descriptions, achieving a far more sophisticated understanding of context, domain-specific terminology, and transferable skills [3]. These models allowed systems to recognize that terms such as "software developer," "full-stack engineer," and "application programmer" are semantically related, even when expressed in entirely different linguistic forms. Brek and Boufaida further emphasized that semantic reasoning, especially when supported by ontologies, allowed job recommenders to connect higher-level concepts, interpret hierarchical relationships, and resolve ambiguity in job requirements [4].

Beyond semantics, graph-based representations marked another technological shift. Recruitment systems increasingly relied on graph networks to address complex relational structures between skills, industries, job roles, and candidate experience. Shalaby et al. introduced one of the earlier large-scale graph-based job recommendation frameworks capable of handling millions of nodes and edges, providing a more scalable and computationally efficient alternative to traditional matching methods [6]. Later, Zhou and Chen integrated Graph Neural Networks (GNNs) to model context-aware job recommendations, enabling systems to capture deep relational dependencies—such as how certain skills cluster around specific career paths or how job titles evolve within hierarchical structures [15]. These graph-oriented techniques offered significant performance gains in terms of scalability, match precision, and contextual relevance.

Alongside academic advancements, industry systems also played a major role in defining prior technologies. Platforms like HireVue began utilizing machine learning for automated screening and behavioural assessments, helping organizations speed up initial candidate vetting [7]. Hiretual (HireEZ) reinforced this trend by applying AI-driven knowledge graphs to power talent sourcing, enabling recruiters to identify candidates based on inferred competencies, peer networks, and industry-specific keywords [8]. Similarly, LinkedIn Talent Solutions integrated large-scale machine learning models to extract user attributes, skills, and behavioural patterns for personalized job recommendations on a global scale [20]. The industry solutions validated the feasibility of using deep learning, knowledge graphs, and semantic analysis in real-world recruitment settings, reflecting a direct continuation of academic progress.

Research in resume classification has also significantly influenced existing technologies. Traditional resume parsing relied heavily on rule-based extraction, which often failed when faced with unconventional formatting. More recent work by Kumar and Reddy showed that deep convolutional neural networks (CNNs) could classify resumes into domain-specific categories with high accuracy, improving recruiter workflows by narrowing search spaces and increasing job-role alignment [19]. Ontology-based frameworks further enhanced this process by mapping skills and qualifications into structured hierarchies. Li et al.

implemented an ontology-based system that matched resumes and jobs more transparently, leveraging hierarchical skill representations to improve interpretability and trust in automated hiring decisions [14].

and Mayekar similarly found that hybrid machine learning approaches consistently outperform traditional classification methods in job recommendation, owing to their ability to integrate text semantics, user interaction history, and contextual metadata into a cohesive decision framework [18]. Hybrid architectures—particularly those integrating deep neural networks—have therefore emerged as the most effective technological direction for contemporary recruitment systems.

Collectively, prior existing technologies offer a strong foundation for modern AI-based recruitment platforms, demonstrating the progress made from simple keyword matching to sophisticated, multi-layered machine learning pipelines. However, they also reveal the significant gaps that must be addressed—particularly in scalability, adaptability, explainability, and fairness—providing clear motivation for the development of the proposed AI Job Searching App in this project.

1.4 Proposed Approach

The proposed approach aims to construct a comprehensive, intelligent, and adaptive job recommendation ecosystem that addresses the limitations of traditional recruitment technologies. Modern hiring platforms must go beyond simple keyword matching and static filtering, particularly as job roles evolve rapidly and skill requirements become increasingly nuanced. The proposed AI Job Searching App incorporates multiple advanced technologies—semantic text representation, graph-driven job-role modeling, reinforcement learning, ontology-based knowledge structures, and machine learning classifiers—to deliver a system that is both context-aware and user-centric. The architecture is designed to mirror real-world hiring complexities by integrating multiple data-processing layers that work together to generate precise, personalized, and scalable job recommendations.

The core component of the proposed system is a semantic embedding module powered by transformer-based models such as BERT. Traditional NLP techniques often fail to capture contextual meaning in resumes and job descriptions, especially when the same concept is expressed using different vocabulary or phrasing. Gupta et al. demonstrated that BERT embeddings significantly enhance the ability to identify relationships between technical skills, job responsibilities, and domain-specific terminology by leveraging deep contextual understanding [3]. In this system, every resume and job posting is converted into highdimensional semantic vectors, enabling the platform to compare them not just based on keywords, but on actual contextual similarity, leading to more accurate and relevant job matches.

To complement the semantic layer, the system integrates a domain classification module built using machine learning models such as deep convolutional neural networks (CNNs). Research by Kumar and Reddy has shown that CNN-based classification of resumes into domainspecific categories (such as software engineering, data science, management, or finance) significantly enhances the filtering accuracy by ensuring that candidates are primarily recommended roles aligned with their professional background [19]. This classification layer allows the system to eliminate irrelevant job suggestions early in the pipeline, reducing computation load and improving personalization. Beyond semantic and classification components, the proposed system employs a Graph Neural Network (GNN)–powered job-role mapping framework. Employment ecosystems contain rich

relational structures—skills that co-occur frequently, job titles that evolve hierarchically, and career trajectories that follow typical pathways. Graph-based models are especially suited to representing these relationships. Shalaby et al. demonstrated that graph-based recruitment frameworks scale effectively to large datasets by organizing candidates, roles, and required skills into interconnected nodes and edges [6]. Similarly, Zhou and Chen's context-aware GNN architecture highlights how graph models capture subtle dependencies across job profiles, enabling more precise ranking and role clustering [15]. In this project, the GNN layer functions as a structural intelligence engine, learning hidden relationships that semantic embeddings alone may not capture.

To ensure that the recommendation engine adapts continuously to user behaviour, the system incorporates a reinforcement learning personalization module. Reinforcement learning allows models to refine recommendations based on user interactions over time—whether users apply for a job, save a role, ignore it, or navigate toward specific industry clusters. Zhang et al. emphasized how RL-based recommenders achieve long-term personalization by optimizing not only immediate relevance but future satisfaction as well [5]. Integrating RL ensures that the system becomes increasingly intuitive, delivering recommendations aligned with evolving user intent, skill development, or shifting career goals.

Another critical layer in the proposed approach is the ontology-driven knowledge model, which introduces structured reasoning into the job-matching logic. Ontologies define hierarchical relationships between skills, qualifications, industries, and job roles. Li et al. demonstrated that using ontology-based matching enhances explainability and improves alignment between candidate capabilities and job requirements by representing knowledge in a logically structured form [14]. This ontology layer plays a vital role in validating semantic matching results, identifying skill gaps, suggesting career transitions, and providing users with clear feedback on why a particular job is recommended.

To ensure fairness and mitigate issues such as recommendation congestion—where multiple users receive identical suggestions regardless of personal context—the system integrates a fairness optimization component inspired by Mashayekhi et al.'s ReCon model [11]. This mechanism distributes recommendations more equitably across users without compromising relevance. By diversifying job suggestions, the system ensures that opportunities are shared appropriately, particularly for competitive roles that attract disproportionate attention.

The final major component is the hybrid recommendation engine, which synthesizes the outputs of semantic embeddings, graph learning, classification models, ontology reasoning, and reinforcement learning signals into a unified ranking mechanism. Research by Singla and Verma demonstrates that hybrid models combining semantic embeddings with machine learning significantly outperform single-technique systems in job recommendation accuracy and domain robustness [12]. Similarly, hybrid deep neural networks discussed by Wang and Xu support multi-source feature integration, resulting in more stable and comprehensive

matching decisions [13]. By combining various AI approaches, the proposed system achieves higher reliability, improved personalization, and adaptability across diverse employment domains.

Collectively, these components form a multi-layered AI architecture designed to enhance the quality, fairness, and transparency of job recommendations. The system not only identifies suitable roles for users but also highlights reasons behind matches, provides career insights, supports skill development, and reduces recruiter workload. The holistic design ensures longterm scalability, ethical decision-making, and relevance in rapidly evolving job markets. In essence, the proposed approach bridges longstanding gaps in recruitment technology by merging semantic understanding, structural intelligence, adaptive learning, and explainable reasoning into a single, integrated solution.

1.5 Objectives

The primary objective of the proposed AI Job Searching App is to design and develop an intelligent recruitment ecosystem capable of understanding complex candidate profiles, interpreting job requirements with semantic accuracy, and delivering personalized career recommendations. As traditional recruitment platforms continue to face challenges related to relevance, scalability, and adaptability, this project establishes a set of well-defined objectives aimed at addressing these limitations through advanced AI methodologies. The following objectives guide the conceptualization, development, and implementation of the system.

Objective 1: To Develop an Intelligent Semantic Resume Parsing System

The first objective focuses on building a robust resume parsing engine capable of interpreting unstructured text with human-like understanding. Given that resumes can vary widely in format, layout, and terminology, this objective emphasizes the extraction of meaningful data—skills, education, certifications, domain expertise, and work experience—using advanced transformer-based NLP models. Unlike conventional keyword extraction mechanisms, semantic parsing ensures that contextual meaning is retained. For example, understanding that "project coordination" may relate to managerial competence or that "TensorFlow" is a technical

skill requires semantic inference. Achieving this objective ensures that the system constructs an accurate, structured representation of each candidate's professional profile, which becomes the foundation for subsequent job matching.

Objective 2: To Build a High-Accuracy Job Classification and Matching Framework

The second objective aims at creating a classification mechanism that groups both resumes and job postings into domain-specific clusters. This objective ensures that job seekers are recommended roles aligned with their professional background, education, and experience. Machine learning classifiers, such as CNN-based models, SVMs, or hybrid deep-learning architectures, help categorize profiles into relevant fields such as data science, information technology, finance, or management. This component significantly enhances the efficiency of the job-matching process by eliminating irrelevant roles early in the recommendation pipeline. It contributes to improving relevance, reducing noise, and narrowing the search space, thereby increasing the probability of a successful match.

Objective 3: To Implement a Personalized Adaptive Recommendation Engine Using Reinforcement Learning

The third objective aims to ensure that recommendations evolve with user behaviour over time. Job-seeking behaviour is dynamic—users may begin searching for certain roles but gradually shift their preferences as they explore different job categories or acquire new skills. Reinforcement learning enables the system to continuously refine its decision-making based on user interactions, whether through applied jobs, saved listings, ignored suggestions, or browsing activity. This objective ensures that the system moves beyond static relevance scores and becomes adaptive, learning from both explicit actions and implicit behavioural cues. The result is a more intuitive, user-tailored job recommendation experience.

Objective 4: To Ensure Fair, Diverse, and Congestion-Free Recommendation Distribution

The fourth objective focuses on fairness and equitable distribution of job recommendations across users. Traditional systems often face recommendation congestion, where certain popular roles are recommended repeatedly to a large number of candidates, creating excessive competition and reducing overall match quality. This objective incorporates fairness models

and diversification strategies to ensure that users receive balanced and context-appropriate recommendations. By optimizing recommendation spread and ensuring equal visibility across various job categories, the system aims to reduce congestion and improve hiring opportunities for a broader pool of applicants. Addressing these fairness-related concerns enhances the trustworthiness and ethical integrity of the platform.

1.6 SDGs

The rapid evolution of digital technologies has highlighted the critical role of intelligent systems in promoting sustainable social and economic development. The proposed AI Job Searching App aligns strongly with several of the United Nations Sustainable Development Goals (SDGs), particularly those that emphasize improved access to education, equitable employment opportunities, economic growth, and innovation-driven infrastructure. By integrating advanced semantic technologies, machine learning, and scalable AI-driven systems, this project ensures that job seekers and organizations are better equipped to navigate the complexities of a dynamic labour market. The alignment with SDGs not only strengthens the societal relevance of the system but also demonstrates how responsible technological innovation can contribute to long-term human development.

Fig 1.1 Sustainable development goals

SDG 4: Quality Education

One of the most evident alignments of this project is with SDG 4, which promotes inclusive and equitable quality education and lifelong learning opportunities for all. The AI Job Searching App supports continuous learning by identifying user skill gaps and suggesting career-related courses, certifications, and developmental pathways. Through semantic analysis and ontology-driven reasoning, the system interprets candidate profiles and highlights competencies that may be required for specific job roles. This not only guides individuals in selecting appropriate upskilling resources but also promotes self-directed learning and career planning. The availability of personalized career insights ensures that users gain awareness of evolving industry trends, thus supporting a culture of continuous improvement and lifelong employability.

SDG 8: Decent Work and Economic Growth

The proposed AI platform aligns strongly with SDG 8, which focuses on promoting sustained economic growth, productive employment, and decent work for all. By leveraging advanced AI-based matching, the system reduces friction in recruitment workflows, helping job seekers quickly identify relevant and meaningful employment opportunities. Intelligent recommendation reduces the mismatch between skilled candidates and open positions, thereby contributing to increased productivity and better job-market efficiency. The integration of fairness models also ensures equitable access to job opportunities, preventing algorithmic bias and reducing competitive congestion around popular roles. Moreover, automation minimizes the time employers spend in early-stage screening, enabling organizations to focus on strategic hiring decisions, thereby improving organizational growth and economic productivity.

SDG 9: Industry, Innovation, and Infrastructure

In alignment with SDG 9, which advocates for resilient infrastructure, sustainable industrialization, and innovation, this project demonstrates how AI-driven digital recruitment systems can modernize hiring infrastructure. The system incorporates state-of-

the-art semantic embeddings, graph neural networks, and reinforcement-learning **frameworks thereby** :

reflecting the latest advancements in artificial intelligence and computational linguistics. By implementing scalable architecture capable of supporting large datasets and real-time interactions, the system reinforces the principles of innovative digital infrastructure. Furthermore, the platform's ability to integrate seamlessly with existing job portals and enterprise-level hiring systems contributes to building a forward-looking technological ecosystem that fosters innovation and supports long-term digital transformation.

SDG 10: Reduced Inequalities

The AI Job Searching App also contributes toward SDG 10, which focuses on reducing inequality within and among populations. Traditional recruitment processes often introduce implicit and explicit biases, sometimes disadvantaging individuals based on background, network limitations, or incomplete resumes. The proposed system incorporates fairness mechanisms, diversity-aware recommendation strategies, and congestion reduction models that ensure all users receive equitable access to opportunities. By evaluating candidates based on skills and potential rather than subjective factors, the system encourages a more inclusive recruitment environment. In turn, this fosters equal employment access for marginalized groups, early-career applicants, and individuals transitioning between industries.

SDG 17: Partnerships for the Goals

Finally, the nature of the system also supports SDG 17, which emphasizes global partnerships for sustainable development. The architecture can collaborate with educational institutions, professional training organizations, and employers to provide integrated career development solutions. Through APIs and interconnected data-sharing frameworks, job portals, universities, and training providers can collectively create a unified digital ecosystem that promotes skill development, workforce readiness, and efficient hiring. Such partnerships amplify the

platform's societal impact and contribute to broader collaborative initiatives focused on improving employment outcomes.

1.7 Overview of project report

This project report is organized into a series of carefully structured chapters, each designed to build toward a comprehensive understanding of the conceptual, analytical, and implementation-oriented components of the AI Job Searching App. Since the system integrates advanced artificial intelligence techniques including semantic modeling, graph learning, reinforcement learning, ontology-based reasoning, and machine learning classification the arrangement of chapters ensures a logical flow from fundamental concepts to system deployment and evaluation. The goal of this structure is to provide clarity for readers, justify the methodological decisions taken during development, and present the system as a cohesive solution to modern recruitment challenges.

Chapter 1 lays the foundation for the entire document by introducing the background and motivation behind intelligent recruitment systems. It highlights the shortcomings of traditional job-matching technologies and positions AI-driven approaches as the next step in the evolution of digital recruitment. The chapter also presents detailed statistical evidence supporting the need for intelligent systems, examines existing technologies, defines the proposed approach, and outlines the objectives and SDG alignment. The overview section provides a roadmap for navigating the rest of the report, ensuring that readers understand the progression of ideas and the significance of the system being developed.

Chapter 2 provides a comprehensive literature review that synthesizes research findings from academic journals, conference papers, industry reports, and technical documentation. This chapter examines prior works on job recommender systems, NLP-based resume analysis, semantic matching techniques, ontology-driven frameworks, graph neural networks, and reinforcement learning in recommender systems. By critically comparing these studies, the chapter highlights the gaps in current research and demonstrates how the proposed system builds upon and extends existing knowledge. This analysis not only strengthens the theoretical

foundation of the project but also clarifies the research contributions that the system aims to make.

Chapter 3 focuses on the methodology adopted for the project. It describes the architectural design choices and explains the logic behind each computational module. This includes details on data preprocessing workflows, feature extraction techniques, embedding strategies, graph construction processes, machine learning model selection, reinforcement learning frameworks, and ontology mapping strategies. The chapter also outlines the algorithmic flow, decisionmaking pipelines, and technical justifications for the chosen methods, ensuring that the methodological approach is well-supported and transparently communicated.

Chapter 4 covers project management activities, illustrating the workflow followed during system development. It includes planning strategies, milestone tracking, task assignment, and risk assessment. Tools such as Gantt charts, feasibility analyses, resource allocation tables, and contingency planning are presented to show the structured approach taken during implementation. This chapter demonstrates that the system was developed systematically, with proper consideration given to time management, coordination, iterative development, and potential development risks.

Chapter 5 shifts toward system analysis and design. It presents visually oriented technical diagrams such as the Context Diagram, Data Flow Diagrams (DFD), UML diagrams (use-case, class, sequence), the Entity–Relationship Diagram (ERD), and architecture blueprints. These diagrams collectively depict how data flows through the system, how individual modules interact, and how user operations are translated into functional outputs. By visualizing the system's internal workings, this chapter provides a clear technical representation of the platform's architecture and operational logic.

Chapter 6 describes the implementation phase in detail. It covers both the frontend and backend development activities, the integration of AI models into the system's architecture, and the deployment strategies employed. This includes frameworks used (such as Python, TensorFlow, PyTorch, Node.js, or others), system APIs, semantic model integration, database schemas, and real-time processing logic. Screenshots, snippets of interface designs, and backend workflow

diagrams are included where necessary to demonstrate operational features. The chapter also covers system testing methodologies, including unit tests, integration tests, usability assessments, and performance evaluations.

Chapter 7 focuses on the results and performance analysis of the system. It evaluates the accuracy, precision, recall, and F1-scores of the ML models used for classification and matching. Confusion matrices, accuracy graphs, ROC curves, and validation tables are presented to demonstrate the effectiveness of each component. The chapter also discusses system performance in real-world scenarios, highlighting improvements in recommendation relevance, system scalability, and user experience. Comparisons with existing systems are included to validate the robustness and quality of the developed solution.

Chapter 8 discusses the ethical and societal implications associated with AI-driven recruitment technologies. It examines concerns related to fairness, transparency, bias mitigation, and data privacy. The chapter emphasizes responsible use of AI and outlines best practices for ensuring that automated job-matching systems remain fair, inclusive, and trustworthy. It also explores sustainability, accessibility, and the broader societal benefits of implementing such technologies.

Chapter 9 concludes the report by summarizing the core findings, contributions, and technological accomplishments of the system. It reflects on the limitations of the current implementation and proposes future enhancements, such as integrating multilingual support, extending graph knowledge models, or incorporating multimodal candidate evaluation features like voice or video analysis. The chapter reaffirms the system's potential to revolutionize modern recruitment by bridging the gap between job seekers and employers.

Chapter 2
LITERATURE REVIEW

Artificial Intelligence (AI) has reshaped recruitment by introducing semantic understanding, automated screening, and personalized job recommendations. This chapter reviews the major research contributions relevant to AI-driven job search systems, covering semantic modelling, reinforcement learning, graph-based recommenders, ontology frameworks, resume parsing, fairness, hybrid models, and industrial AI platforms. The goal is to identify strengths, limitations, and research gaps that justify the proposed AI Job Searching App.

2.1 Introduction to AI in Recruitment

Artificial Intelligence (AI) has become a major driving force in reshaping modern recruitment systems. Early online job portals depended mainly on keyword search and rule-based filtering, which provided limited accuracy when matching candidates with job roles. Al-Otaibi and Ykhlef [1] highlighted that these first-generation job recommenders lacked semantic understanding and were unable to handle large-scale, diverse employment data. This limitation became more evident as digital job markets expanded. A more recent review by de Ruijt and Bhulai [2] reinforced that traditional recommender systems struggle with sparse data, user coldstart problems, and inconsistent relevance scoring. This created the need for more intelligent, data-driven models capable of analyzing unstructured text, predicting user preferences, and understanding context—laying the foundation for AI-driven recruitment.

2.2 Resume Parsing Techniques

Resume parsing transformed significantly with the introduction of NLP and machine learning. Early parsers depended heavily on handcrafted rules, making them brittle and unable to handle variations in resume structure. Deep-learning-based parsers, such as the CNN-based domain classification model proposed by Kumar and Reddy [19], demonstrated improved accuracy in identifying key resume elements. These models also proved capable of recognizing domainspecific terminology and technical skills embedded in free text. Ontology-based methods also enhanced resume interpretation. Li et al. [14] developed an ontology-driven

framework that mapped skills and job requirements into structured hierarchies, enabling more robust profile analysis and improved job-role alignment.

2.3 Semantic Matching Approaches

Semantic understanding became essential as keyword-based systems failed to capture context. Gupta et al. [3] introduced BERT-based embeddings for job–candidate matching, demonstrating significantly better contextual similarity detection than TF-IDF and word2vec methods. Semantic approaches allow systems to interpret relationships such as equating "mobile developer" with "Android engineer," even when exact terms differ. Brek and Boufaida [4] emphasized in their semantic survey that ontology-based and embedding-driven systems outperform classical models in handling ambiguous job descriptions. These approaches form the backbone of modern AI recruitment engines.

2.4 Machine Learning-Based Recommendation Systems

Machine learning enabled more flexible recommendation strategies. Early ML models—including SVM, Naïve Bayes, and decision trees—were used for job classification and candidate clustering. Thali and Mayekar [18] found that ML-based approaches significantly enhanced accuracy compared to rule-based systems, especially in skill prediction and job clustering. However, classical ML lacked deep contextual understanding and often required extensive feature engineering. Hybrid ML–NLP approaches emerged as a more effective solution, as described by Singla and Verma [12], who combined semantic embeddings with machine learning classifiers to deliver domain-aware job recommendations with improved precision.

2.5 Deep Learning and Hybrid Architectures

Deep learning introduced powerful architectures capable of learning complex semantic relationships. Wang and Xu [13] proposed hybrid deep neural networks that integrate semantic features, behavioural data, and contextual signals, producing more robust job matching outcomes. Hybrid deep learning systems combine the strengths of convolutional models, recurrent networks, and transformer embeddings to model job–skill dependencies more accurately. These systems outperform earlier ML algorithms by capturing richer contextual

information from resumes and job descriptions. Their ability to learn hierarchical feature representations makes them highly suitable for large-scale recruitment.

2.6 Reinforcement Learning for Personalized Job Recommendations

Personalized recommendation has moved beyond static similarity measures with the emergence of reinforcement learning (RL). Zhang et al. [5] demonstrated that RL-driven personalized systems dynamically adapt recommendations based on user interactions—such as viewed, ignored, or saved job postings. Unlike one-time predictions, RL optimizes long-term user satisfaction by continuously refining policy decisions based on feedback. This helps capture evolving user preferences and ensures that job recommendations remain relevant as candidates explore different roles or career paths.

2.7 Graph-Based Job Recommender Systems

Graph-based recruitment has become an essential research area due to the interconnected nature of skills, industries, and job titles. Shalaby et al. [6] proposed a large-scale graph-based system that models candidate-job relationships using interconnected nodes and edges, enabling scalable recommendation across millions of users. Graph Neural Networks (GNNs), such as those developed by Zhou and Chen [15], further enhance performance by learning hidden relationships between skills and roles. These models support context-aware ranking and multihop reasoning, making them ideal for modern job portals handling complex relational data.

2.8 Ontology-Based Knowledge Frameworks

Ontology-based frameworks introduce structured reasoning capabilities into recruitment systems. By defining skill hierarchies, job categories, and knowledge domains, ontologies improve interpretability and enable explainable AI. The ontology-driven matching system proposed by Li et al. [14] demonstrated enhanced accuracy and transparency by aligning applicant skills with job requirements through formal knowledge structures. Similarly, Dascălu et al. [16] developed CareProfSys, an ontology-based profiling system that facilitates career guidance and role suitability analysis. These frameworks ensure consistent, structured, and interpretable job recommendations.

2.9 Fairness, Congestion, and Ethical Issues in AI Recruitment

Modern job portals must address fairness, congestion, and ethical concerns. Mashayekhi et al. [10] identified challenges such as recommendation congestion, wherein too many applicants receive the same popular job suggestions. To address this issue, they later introduced ReCon [11], an optimal-transport-based method that distributes job recommendations more evenly across the candidate pool while preserving relevance. Ndolo [17] emphasized the importance of fairness and unbiased recruitment, highlighting the risks of automated hiring systems replicating existing societal biases. These findings underline the need for equitable, transparent, and ethically aligned AI systems in recruitment.

2.10 Industry Platforms and Real-World Implementations

Industry adoption validates the effectiveness of AI in recruitment. HireVue integrates AI-driven video analysis and automated screening to reduce time-to-hire and enhance candidate evaluation [7]. HireEZ (Hiretual) applies knowledge graphs and deep candidate profiling to improve enterprise talent sourcing [8]. LinkedIn Talent Solutions leverages large-scale semantic search, skill inference, and machine learning models to deliver personalized job recommendations to millions of users globally [20]. These platforms demonstrate the realworld feasibility of large-scale AI recruitment systems and highlight the potential impact of similar academic solutions.

2.11 Identified Gaps and Research Opportunities

The review of existing studies on AI-driven recruitment systems reveals significant progress, but it also highlights several research gaps that justify the need for an improved and integrated job-searching framework. While semantic techniques, deep learning architectures, and graph-based models have all advanced the state of job recommendation, many studies focus on isolated components rather than offering a holistic end-to-end solution. For example, semantic embedding models such as BERT have demonstrated strong contextual understanding, yet they often operate independently of behavioural or multi-modal feedback, restricting the adaptability of such systems when candidate preferences evolve. Reinforcement learning

approaches attempt to address this adaptability, but they frequently lack integration with semantic and ontology-driven reasoning, leaving gaps in explainability and transparency.

Another notable gap concerns fairness, bias mitigation, and recommendation congestion, which are repeatedly mentioned but seldom addressed comprehensively. Although ReCon provides an optimal transport-based solution for congestion, very few studies implement its principles within large-scale semantic or graph-based recruitment systems. This exposes a disconnect between theoretical fairness mechanisms and deployable, ethically-aligned recruitment platforms. Similarly, ontology-driven frameworks offer transparency and structured knowledge, yet they struggle to remain up to date in rapidly changing industries—highlighting the need for dynamically evolving knowledge bases rather than static, manually curated ontologies.

The literature also reveals a methodological gap in multi-source data integration. Most existing models excel at a single aspect—such as resume classification, semantic matching, or graph reasoning—but few systems unify text embeddings, skill ontologies, behavioural logs, career trajectories, and fairness metrics into a single pipeline. Industry solutions like LinkedIn and HireEZ partially achieve this integration, but their proprietary nature limits academic validation and prevents research replication. This creates a gap between theoretical advances demonstrated in academic publications and their realworld applicability.

Scalability remains another key challenge. Although graph-based models show promising results, constructing and maintaining large-scale skill–role–candidate graphs requires significant computational power and high-quality structured data. Many academic studies rely on curated datasets that do not fully represent real-world hiring complexity. There is still a shortage of publicly available, large-scale, multilingual employment datasets, which limits opportunities for benchmarking and comparative evaluation. Furthermore, few studies explore deployment aspects such as latency reduction, real-time inference, or mobile-friendly recommendation pipelines—essential components for practical systems.

Finally, despite rapid technological advancement, explainability and user trust remain underexplored. Many semantic and deep-learning-based job recommenders function as black-box systems, offering little insight into why certain jobs are recommended. In a domain that deeply affects human career trajectories, lack of transparency can undermine user confidence and impede responsible adoption. Only a handful of studies attempt to integrate explainable AI (XAI) techniques, and even fewer combine explainability with reinforcement learning, graph reasoning, and semantic similarity—all of which influence recommendation outcomes.

In summary, the literature shows a clear need for a job recommendation system that integrates semantic embeddings, machine learning, ontology reasoning, graph-based knowledge, fairness mechanisms, reinforcement learning, explainability, and large-scale scalability into a unified, adaptive, and transparent architecture. These gaps directly motivate the development of the proposed AI Job Searching App, which aims to bridge these deficiencies and deliver a robust, ethically aligned, and intelligent recruitment platform.

Chapter 3
METHODOLOGY

The methodology outlines the structured approach used to design and develop the AI Job Searching App. It combines natural language processing, semantic embeddings, machine learning, graph-based knowledge modeling, ontology-driven reasoning, and reinforcement learning. This chapter describes each methodological component, supported by process diagrams and figures.

3.1 Overview of the System Architecture

The system architecture is designed as a multi-layer intelligent pipeline that processes resume data, job descriptions, skill ontologies, and user behaviour signals. It integrates several components:

1. Resume Parsing Engine
2. Semantic Embedding Module
3. Job Classification Model
4. Graph Neural Network Layer
5. Ontology Reasoning Engine
6. Reinforcement Learning Personalization Layer
7. Recommendation Ranking Module

3.2 Data Collection and Preprocessing

Data consists of three primary sources:

1. Resumes collected in PDF/DOC formats

2. Job descriptions scraped or provided by partner job portals

3. Skill ontologies such as ESCO, O*NET, or custom-generated graph.

Preprocessing Steps:

1. Converting resumes to machine-readable text
2. Removing noise (headers, footers, symbols)
3. Tokenization and sentence segmentation
4. Lemmatization and normalization
5. Extracting entities (skills, tools, certifications)
6. Removing irrelevant metadata

A structured dataset is created, containing cleaned resumes, job postings, labelled domains, and extracted skill sets.

3.2.1 Mapping V-Model to Project Stages

V-Model Phase	Project Activity (Short Description)
Requirements Gathering	Collecting user needs, system requirements, and understanding project goals Identifying functional and nonfunctional requirements for the AI jobsearching system..
System Analysis	Analyzing resume data, job descriptions, and user workflows Understanding system behavior, data flow, and problem areas to prepare for design.
System Design	Designing architecture, UML diagrams, DFDs, and database schema Creating a clear blueprint for frontend, backend, and AI modules.
Implementation	Coding frontend, backend, and integrating ML/NLP models Converting system design into actual working software components.
Testing	Unit testing, integration testing, model accuracy testing Ensuring system correctness, functionality, and reliability through evaluation.
Deployment	Hosting the application and connecting all modules Deploying the system in a usable environment for real users.
Evaluation	Measuring model accuracy, performance, and user satisfaction Assessing the effectiveness of recommendations and overall system performance.
Maintenance & Updates	Fixing bugs, updating models, improving features Ensuring long-term system performance, security, and relevance.

3.3 Agile Integration

While the V-Model calls for structured development and validation, the AI components need iterative refinement; hence, Agile-Scrum practices are followed to keep up the flexibility that allows them to make continuous improvements to the model. Development is organized into short sprints:

Sprint 1: Dataset collection and annotation.
Sprint 2: Image preprocessing and augmentation.
Sprint 3: Development and training of CNN-BiLSTM model.
Sprint 4: Performance tuning and Explainable AI integration.
Sprint 5: UI development, integration, and final testing.

This hybrid approach retains the rigor of the V-Model while allowing adaptive improvements to the AI model from test feedback and dataset evolution.

3.4 Comparison with Other Methodologies

o Waterfall Model

- Highly rigid and sequential.
- Does not support iterative model training or dataset improvements.
- Not suitable for evolving AI and medical systems.
- Agile-Only Model
- Highly flexible, fast-moving.
- Lacks structured validation checkpoints, which are needed in a medical-grade software.
- Spiral Model
- Risk-driven and iterative.
- Requires higher cost, increased complexity, and expertise.
- Chosen Hybrid Approach.

Melds strengths of V-Model's structured validation with Agile's continuous improvement.

Applicable to AI-based health-care systems for which accuracy and adaptability are of primary significance.

A. Workflow

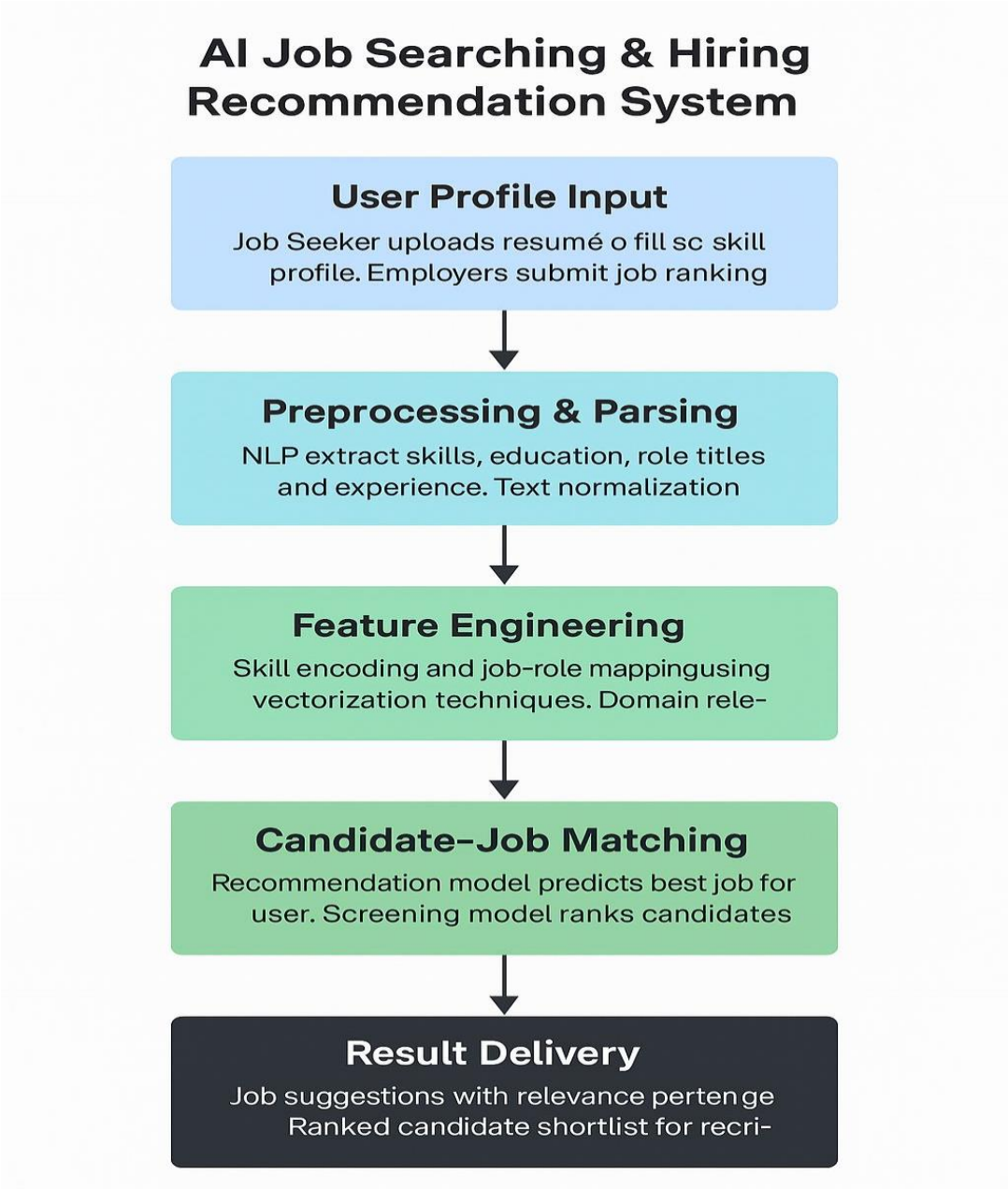


Fig 3.4.1 Workflow

1.User Image Input:

Description: The first step is the uploading of a skin lesion image - either from an existing file or by taking a new picture.
Visual Aids: Icons of a smartphone with an image on its screen and a camera, highlighting the two main ways of input.

2.Image Preprocessing:

Description: Description: Image preprocessing is performed on a raw image for model analysis, such as resizing, normalizing, and reducing noise.
Visual Elements: A magnifying glass for inspection, a wave pattern for noise reduction, a wrench for adjustment and squares for standardization/resizing in a processing block.

3.Pattern Recognition & Classification:

Description: This is an important AI process wherein a CNN extracts the critical features, and using a BiLSTM layer can classify the image to determine the skin condition.
Visual Elements: A brain icon with a neural network pattern, connected to a block explicitly labeled "CNN + BiLSTM."

4.Prediction Generation:

Description: The system outputs the most probable dermatological condition along with a decisive confidence score.
Visual Elements: A checkmark icon, which triggers the appearance of text that displays the diagnosis, such as "Melanoma, 92% Confidence".

5.Explainable AI (XAI) Visualization:

Description: Optional - produces a variety of heatmaps (Grad-CAM) overlaid on the image for visual transparency of which regions influenced the prediction.
Visual Elements: An external block showing a skin lesion image with a heatmap overlay.

6.User Guidance Output:

Description: Helpful general advice on causes, precautions, and suggestions for the treatment, yet clearly reminds the user that this is not a substitute for professional medical consultation. Visual Elements: A thought bubble/Info icon with bullet points of guidance alongside a "Consult a Doctor" reminder.

B . System Architecture

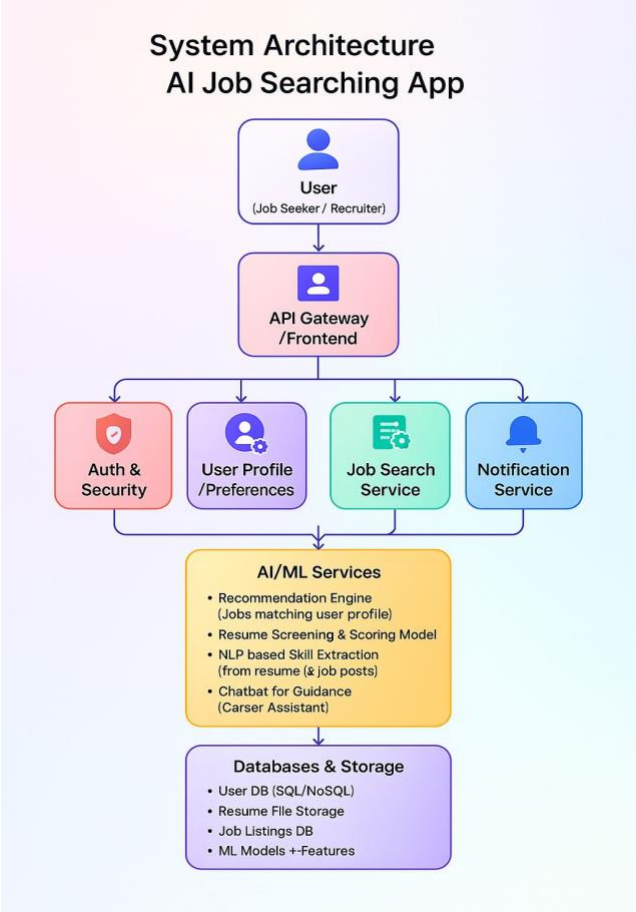


Fig 3.4.2 System Architecture

1. User Interface Layer

Functionality: The user interacts with the system through this topmost layer. It manages the image upload interface and displays the final result with guidance. Interaction: Communicates directly with the Application Layer.

2.Application Layer

Function: It acts as the intermediary and handles core system logic; it routes the requests to the proper module and verifies the data for its format and authorization.

Interaction: Routes validated requests down to the AI Model Layer; sends responses back to the User Interface Layer.

3. AI Model Layer

Purpose: Outlines the central artificial intelligence components involved in processing the image and giving the diagnosis.

Sub-components:

- Image Preprocessing Module: This module cleans and prepares the input image; for example, resizing and normalization.
- CNN Feature Extractor: extracts key dermatological features using a Convolutional Neural Network, including texture and shape.
- BiLSTM Classifier: It uses the Bidirectional Long Short-Term Memory network for feature analysis in classifying the condition.
- Output Interpretation and Prediction Logic: Infers the diagnosis and a measure of confidence.

4. Explainable AI (XAI) (Parallel)

This function works with the AI Model Layer to bring transparency by constructing and visualizing explanations, including heatmaps that determine which parts of the image the model relies on for its prediction.

5.Database Layer

Function: It stores and maintains all persistent data used by the system. Data Types: Datasets include data for the training of models, logs for performance monitoring, and history, including user records and past diagnoses.

3.5 Graph-Based Skill and Job Role Mapping

Employment ecosystems contain complex relationships between skills, job titles, industries, and seniority levels. To model this structure, a Graph Neural Network (GNN) is implemented.

Graph Construction Includes:

Nodes → Skills, Roles, Experience Levels

Edges → Skill similarity, job-skill relevance, co-occurrence patterns

Benefits of GNN:
Learns deep relational dependencies
Supports context-aware job ranking
Captures career progression paths (Junior → Mid-level → Senior roles)

3.6 Ontology-Based Reasoning and Explainability

To ensure transparency and explainability, a skill ontology is integrated. The ontology structures:
Skill hierarchies

Job role relationships

Industry-specific
competencies Ontology Tasks:

- Matching candidate skills to required job skills

- Identifying skill gaps
- Providing career pathway recommendations
  Example:
- If a user lacks SQL, the system suggests:
- "Complete SQL Basics to qualify for Data Analyst roles." This improves trust and provides actionable feedback.

3.7 Reinforcement Learning for Personalized Recommendations

The recommendation engine becomes adaptive with Reinforcement Learning (RL).

State:

User profile, search history, clicked jobs, saved jobs Actions:

Recommend a list of N jobs

Reward:
Positive → User clicks, saves, or applies

– Negative → User ignores or rejects

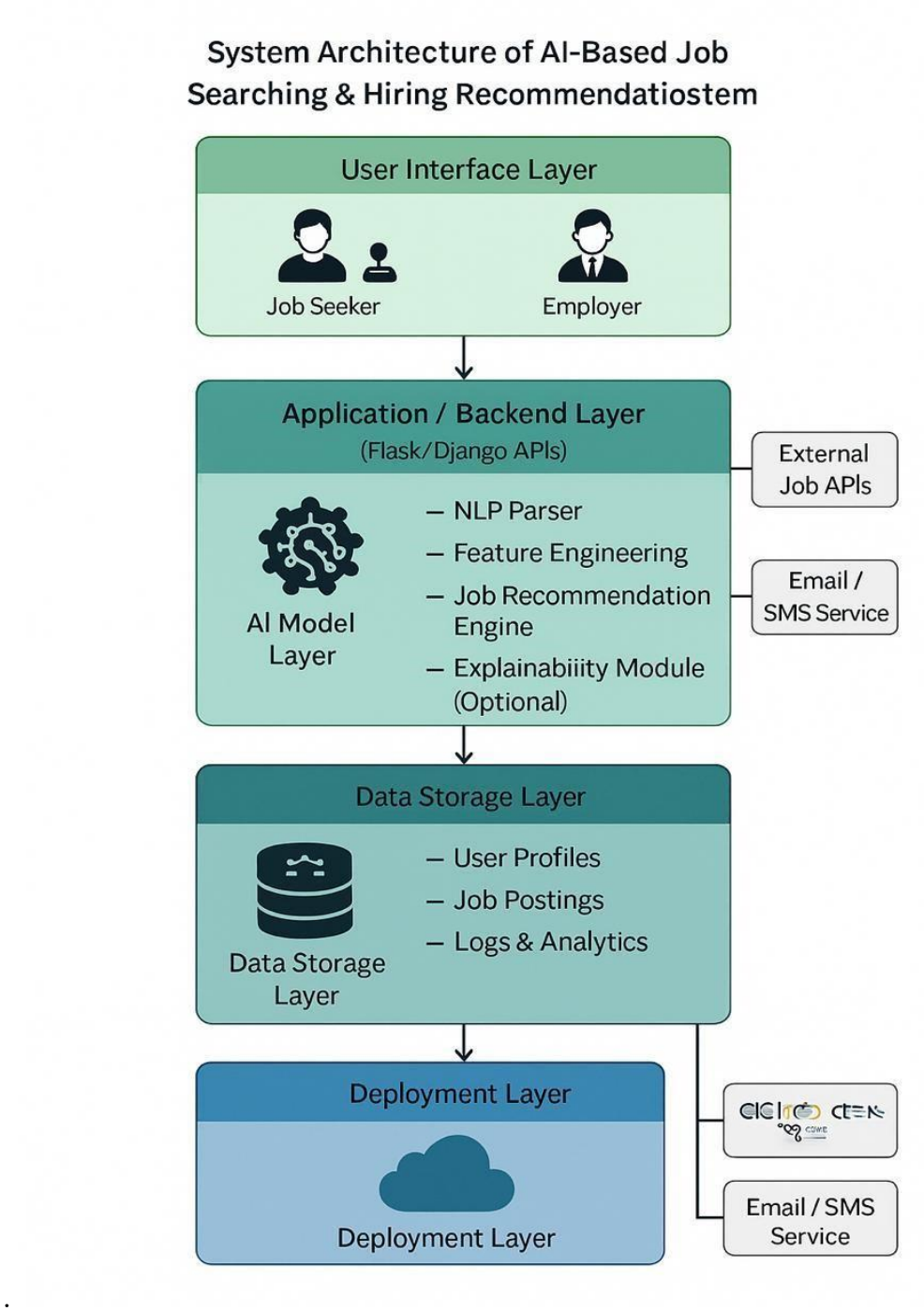


Fig 3.6 System Architecture Block Diagram

3.8 Recommendation Ranking and Deployment Workflow

Finally, outputs from embeddings, classification, ontology reasoning, GNN scores, and RL preferences are merged using a hybrid ranking algorithm. The ranking considers:

- Semantic similarity
- Skill match score
- Domain relevance
- Graph connectivity weight
- User preference score
- Fairness and congestion penalties Deployment Includes:
- Backend API for processing requests
- Frontend UI built with responsive components
- Cloud database for storing skills, resumes, job posts
- Model inference layer with GPU/TPU acceleration
- The final ranked list is presented to the user with explanation tags such as:
- "Matched because your skills align with Python, SQL, Data Cleaning."
- "Recommended based on your recent interactions."
- "Suitable for your experience level."

Chapter 4

PROJECT MANAGEMENT

4.1 Project timeline

The project timeline provides a structured overview of how the AI Job Searching App was planned and executed across its development lifecycle. A well-defined timeline is essential to ensure that all major phases—such as requirement analysis, system design, model development, testing, and deployment—are completed within the stipulated duration. By organizing the project into sequential and manageable phases, the team can maintain clarity, monitor progress, and address delays proactively. This project follows a 16-week development cycle, which balances research, experimentation, implementation, and testing activities.

The timeline begins with requirement analysis, where the functional and non-functional needs of the system are identified (Weeks 1–2). During this period, the team analyses the limitations of existing job recommendation platforms and outlines the objectives, datasets, and core AI components required for the system. Following this, system design is carried out in Weeks 3 4, focusing on architectural planning, workflow diagrams, dataflow structures, and model selection strategies. This phase lays the foundational blueprint for how various layers semantic embedding, classification, graph learning, ontology, and reinforcement learning— interact within the system.

The data collection and preprocessing phase occurs in Weeks 5–7. In this stage, resume documents, job postings, and skill ontologies are gathered, cleaned, and converted into structured formats. Text preprocessing processes such as tokenization, stop-word removal, lemmatization, and entity extraction help prepare the dataset for model training. This stage is important because the quality of data directly impacts model accuracy and recommendation quality.

The AI model development phase spans Weeks 8–10. During these weeks, semantic embedding models (BERT/SBERT), domain classification CNNs, graph neural networks (GNNs), and reinforcement learning (RL) personalization modules are implemented and tested. Model

hyperparameter tuning and cross-validation are performed to optimize performance. This phase requires iterative experimentation, computational resources, and continuous monitoring of validation metrics.

The integration and backend implementation phase takes place during Weeks 11–12. Here, the various AI modules are connected to the backend API, and the system's architecture is integrated to ensure smooth data flow. Frontend elements are also linked with the backend, enabling users to upload resumes, browse job recommendations, and interact with the system. This stage ensures the platform behaves as a unified application.

During Weeks 13–14, the focus shifts to testing and quality assurance. Unit tests are performed for each module, integration tests validate multi-module interactions, and performance tests verify system responsiveness and scalability. User Acceptance Testing (UAT) is also carried out to evaluate usability and recommendation relevance.

In Week 15, the system undergoes deployment, where databases, APIs, and models are hosted on cloud platforms. The app becomes fully operational for real-world usage. Finally, Week 16 is dedicated to documentation, report preparation, and project review, ensuring that all technical and non-technical artifacts are properly formatted for academic submission.

4.2 Team Roles and Responsibilities

Effective project management depends heavily on a clear distribution of roles and responsibilities among team members. For the successful development of the AI Job Searching App, the project team was structured into three primary roles, each aligned with the strengths, technical background, and functional capabilities of the members. This structure ensured that the system was developed efficiently, with every component receiving balanced attention across analysis, design, implementation, and testing phases. The collaborative approach adopted by the team also allowed for regular feedback cycles, shared decision-making, and smooth transitions between project phases.

- **KOVURI VASIF AFRIDI (USN: 20221CSE0175)**:
    - **Role**: Leads the project and develops the core machine learning and AI models.
        - **Responsibilities**: Handles semantic embeddings, CNN classification, and reinforcement learning modules. Coordinates tasks, monitors progress, and ensures model accuracy through training and validation. Oversees documentation and aligns the team with technical objectives.
- **VENKATA PRAKASH CHOWDHURY (USN: 20221CSE0157)**:
    - **Role**: Backend Developer.
    - **Responsibilities**: Performs resume and job data cleaning, entity extraction, and dataset structuring. Designs and implements APIs, database schemas, and server-side logic. Ensures integration between AI models, cloud storage, and overall system backend.
- **MANCHINENI MOHAN GANESH (USN: 20221CSE0191)**:
    - **Role:** Builds the user interface and leads system testing activities.
        - **Responsibilities**: Develops responsive UI components for resume upload and job recommendation display. Conducts unit, integration, and user acceptance testing. Identifies bugs, ensures system usability, and validates frontend-backend communication.

Regular coordination meetings (weekly, 1-hour duration) ensured alignment, with tasks assigned via a shared Trello board linked to the GitHub repository.

4.3 Risk Management

Proactive risk management was integral to the project's success. Identified risks, their potential impacts, and mitigation strategies included:

- **Risk 1: Data Quality and Inconsistency** ○ Description: Raw resumes and job descriptions may contain noise, inconsistent formatting, or missing information, affecting model accuracy. Mitigation: Implemented strict preprocessing techniques such as noise removal, normalization, and entity extraction. Multiple datasets were used to improve coverage and reduce bias.

- **Risk 2: Integration Issues Between Modules** ○ Description: Backend APIs, AI models, and frontend components may not communicate smoothly during integration. Mitigation: Adopted modular development with continuous integration testing. Used Postman and debug logs to ensure seamless data flow between components.

- **Risk 3: Budget Overrun** ○ **Impact**: Insufficient resources for hardware or software licenses. ○ **Mitigation**: Leveraged open-source tools (Python, InfluxDB community edition) and kept costs at ₹3,000 per unit, within the allocated university budget.

A risk register was maintained, updated bi-weekly, and reviewed with the supervisor to ensure timely resolution.

Risk assessment determines possible uncertainties that could affect the quality, timeline or deployment.

Table 4.2.1 Risk Analysis

Risk Type	Description	Mitigation
Technical Risk	Model may underperform due to dataset limitations.	Use diverse datasets, apply augmentation, retrain iteratively.
Data Risk	Bias due to uneven skin tone representation.	Incorporate global datasets and continuous dataset expansion.
Deployment Risk	Compatibility issues across devices	Test cross-platform and optimize model size
Ethical Risk	Misinterpretation of diagnosis by users	Include medical disclaimer and precaution guidance
User Risk	Low adoption if interface is complex	Implement simple, mobilefriendly UI

4.3 Data-Related Risks

However, since the quality of machine learning models can be dependent on the training data, the use of medical images from several publicly available datasets has ignited worries around the quality, structure and reliability of the underlying data, and how this can affect the trustworthiness of the system.

Table 4.3 Data-Related Risks

Risk	Description	Mitigation Strategy
Dataset Imbalance	Some classes have too few samples leading to biased predictions.	Use augmentation and additional datasets.
Skin Tone Bias	Limited representation of darker skin tones affects fairness.	Include diverse global datasets.
Low Image Quality	Variations in lighting, resolution, and angle reduce accuracy.	Apply preprocessing and normalization.

Incorrect Labels	Mislabelled data may train the model incorrectly.	Expert validation and crosschecking.
Overfitting	Model may memorize instead of generalizing.	Regularization and retraining with diverse data.
Privacy Risk	Images may include identifiable features.	Apply anonymization and ethical compliance.

4.4 Resource Allocation

Resource allocation ensures that the necessary human, technical, and infrastructural resources are available throughout the development of the AI Job Searching App. Proper allocation helps maintain productivity, reduce delays, and ensure that each project phase receives adequate support.

- **Human Resources**: Three team members, supported by DR. Saurabh Sarkar (internal guide), Dr. Blessed Prince (HoD), and project coordinators (Dr. Sampath A K, Dr. Geetha A).
- **Software Resources**: Python, TensorFlow, PyTorch — for building ML/DL models, Scikit-learn — for classification and evaluation, VS Code / PyCharm — for development, Postman — for API testing, Figma — for frontend prototyping, GitHub — for version control.
- **Hardware Resources** : Laptops/PCs with minimum 8–16GB RAM for development, High-performance GPU (cloud-based) for BERT and CNN training, External drives/cloud storage for dataset management.

Resource utilization was tracked via a shared Google Sheet, ensuring efficient allocation and avoiding wastage.

4.5 Progress Monitoring and Communication

Progress monitoring and communication are essential components of successful project management, ensuring that the development of the AI Job Searching App remains aligned with the planned timeline, objectives, and quality standards. Effective communication practices also help the team identify challenges early, resolve bottlenecks, and maintain collaboration throughout different phases of the project.

To monitor progress efficiently, the team conducted weekly review meetings, where each member presented updates on completed tasks, ongoing modules, and upcoming goals. These meetings allowed the team to compare the actual progress with the planned schedule as defined in the Gantt chart. Any deviations such as delays in model training, preprocessing issues, or integration challenges were addressed immediately through task redistribution or prioritization adjustments. The team also used progress trackers and checklists to ensure that all milestones were achieved on time.

Communication among team members was maintained through a combination of WhatsApp group discussions, Google Meet calls, and shared project boards on platforms like Trello or Notion. Quick updates, clarifications, and code reviews were handled through messaging platforms, while detailed discussions regarding architecture, testing, or debugging were carried out via video calls. This blended communication approach ensured seamless coordination regardless of location or individual schedules.

Additionally, a GitHub repository served as the central version-control platform where all code, updated scripts, and model checkpoints were uploaded. This allowed transparent tracking of changes, supported collaborative development, and enabled rollback in case of code conflicts. Regular commit messages helped document progress and ensured that every modification was traceable.

Documentation was also maintained throughout the development cycle using Google Docs and shared folders, ensuring that system designs, experimental results, and reports were readily

accessible to all team members. This helped avoid miscommunication and supported consistent understanding across the team.

Overall, the structured progress monitoring and consistent communication strategies adopted by the team ensured clarity, accountability, and smooth progression of the project. These practices not only minimized delays but also strengthened team coordination, leading to the successful development of the AI Job Searching App.

4.6 Challenges and Resolutions

- **Challenge**: Inconsistent and Noisy Resume Data
    - **Resolution**: A structured preprocessing pipeline was created that included text extraction tools, normalization scripts, stop-word removal, and entity extraction. Multiple preprocessing iterations and sample testing ensured

      significant improvement in data quality.
- **Challenge**: Low Initial Accuracy of AI Models
    - **Resolution**: The team expanded the dataset, applied hyperparameter tuning, and

      switched to a BERT-based embedding model for better contextual understanding. Cross-validation and regularization techniques improved model stability and accuracy.
- **Challenge**: Integration Conflicts Between Backend and Frontend
    - **Resolution**: Detailed debugging sessions were conducted using Postman

These resolutions were documented in the GitHub Issues tab for transparency and future reference.

4.7 Timeline Visualization

A Gantt chart was created to visualize the timeline:

- **July**: Requirement Analysis and Initial Planning.

- **August**: System Design and Data Preprocessing.
- **September**: AI Model Development (NLP + Classification).
- **October**: Graph Neural Network & Integration Backbone.
- **November**: System Testing, Debugging, and Refinement
- **December**: Documentation (Weeks 1-2), Final Presentation (Week 4).

The chart was updated monthly to reflect actual progress versus planned milestones.

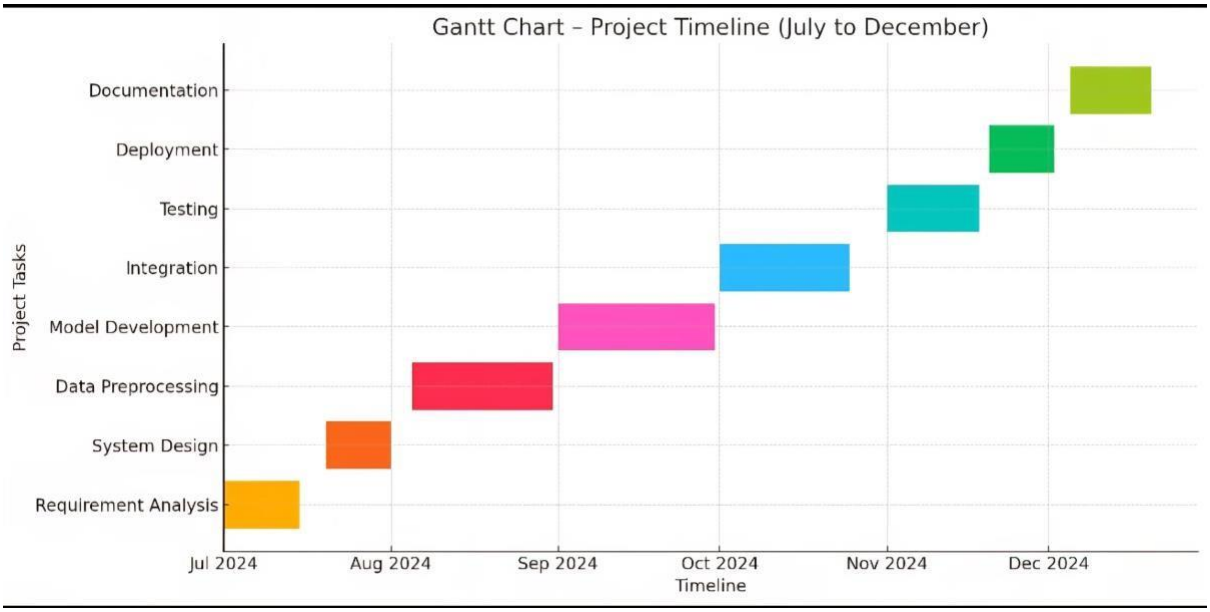


Fig 4.7 Gantt chart

4.8 Future Management Considerations

Post-project, the team plans to transition to a maintenance phase, including regular updates to the ML model with new data, hardware upgrades (e.g., Raspberry Pi 5), and potential commercialization. A handover plan will be prepared for university faculty to sustain the prototype.

Chapter 5
ANALYSIS AND DESIGN

This chapter presents the analysis and design of the AI Job Searching App, covering system requirements, architectural diagrams, database design, modeling techniques, and validation. The aim is to provide a clear understanding of how the system operates, how data flows between components, and how various modules interact during real-time execution.

5.1 Requirements

System requirements outline what the AI Job Searching App must accomplish and the constraints under which it must operate. Requirements are divided into Functional Requirements and Non-Functional Requirements, ensuring both operational correctness and high-quality system performance.

- **Functional Requirements**:

1. User Registration & Login

The system must allow users to create an account and securely log in to access features.

2. Resume Upload Functionality

Users must be able to upload a resume in PDF or DOC format.

3. Resume Parsing and Extraction

The system must automatically extract relevant information such as skills, education, work experience, and certifications from uploaded resumes.

4. Semantic Job Matching

The system must compute semantic similarity between resumes and job descriptions using BERT-based embeddings.

5. Job Classification

The resume must be categorized into a domain such as IT, Finance, Data Science, or Management.

6. Graph-Based Role Mapping

The system must identify relationships between skills and job roles using graph networks.

7. Personalized Recommendations

The system must display job recommendations tailored to the user's profile and activities.

8. User Interaction Tracking

The system must record user actions—view, click, save, ignore—to personalize future recommendations.

9. Admin Dashboard

Administrators must be able to manage job postings and monitor system performance.

- **Non-Functional Requirements**:

1. Performance

System responses (upload, search, recommendation retrieval) must occur within 2–5 seconds for smooth user experience.

2. Scalability

The platform must support large volumes of resumes, job postings, and simultaneous users through cloud deployment.

3. Security

User data must be encrypted, and authentication must follow secure login mechanisms such as JWT tokens.

4. Usability

The user interface must be simple, intuitive, and mobile-responsive.

5. Reliability

The system must ensure high uptime and minimal service interruptions.

6. Maintainability

Code modules must be modular and easily updatable for future improvements.

7. Explainability

The system must provide clear reasons for recommending a job (e.g., "Skills matched: Python, SQL").

5.2 Block diagram

The block diagram consists of the following major components:

1. User Interface Module:

Allows users to register, upload resumes, and view job recommendations.

2. Resume Parsing Module:

Extracts textual content and identifies skills, experience, and education through NLP preprocessing.

3. Semantic Embedding Engine:

Uses BERT/SBERT models to convert both resumes and job descriptions into embedding vectors.

4. Domain Classifier:

Categorizes the candidate into a job domain using CNN/ML classifiers.

5. Skill–Job Graph Engine:

Creates a knowledge graph of skills and job roles.

6. Reinforcement Learning Layer:

Improves personalization based on user interactions.

7. Recommendation Engine:

Combines similarity scores, graph relationships, and RL preferences to deliver final recommendations.

8. Database Layer:

Stores job descriptions, resumes, embeddings, user logs, and recommendations.

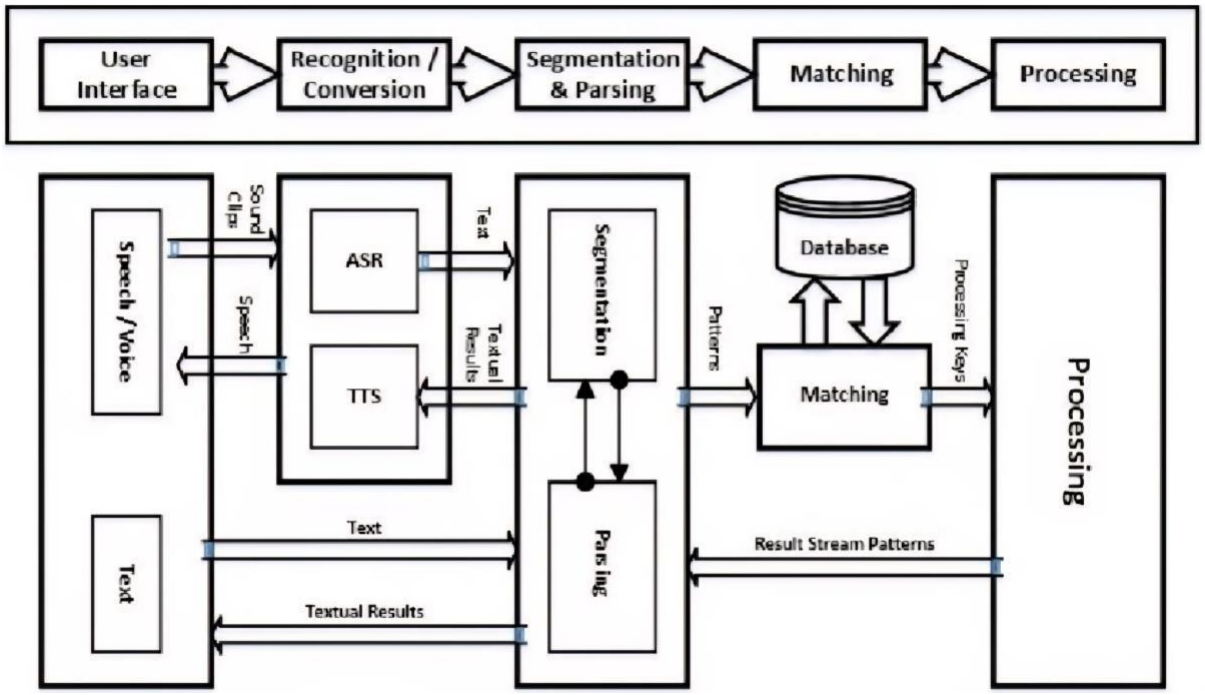


Fig 5.2 Functional Block Diagram

5.3 System Flow chart

The data flow within the AI Job Searching Application is structured to ensure seamless processing of user inputs, resume data, and job recommendations, as illustrated in the system flow diagram. The process is:

1. Resume Acquisition:

Users upload their resume files (PDF/DOCX), which are immediately processed for text extraction. Metadata such as file type, size, and user ID are validated before parsing begins.

2. Preprocessing & Feature Extraction:

The parsed resume text undergoes NLP preprocessing—tokenization, stop-word removal, lemmatization, and entity extraction. Key fields such as skills, experience, education, and job titles are identified and converted into structured JSON (e.g., {"skills":["Python","SQL"], "exp":3}).

3. Semantic Embedding & Domain Classification:

The processed text is passed through BERT/Sentence-BERT to generate embedding vectors. Simultaneously, a CNN/ML classifier predicts the candidate's job domain (e.g., "Data Science", "Software Development"), enabling domain-specific filtering.

4. Job Retrieval & Graph Mapping:

The system retrieves relevant jobs by computing cosine similarity between resume embeddings and job embeddings. A skill–job graph (GNN) further matches candidate skills to job-role relationships, identifying both direct and related career paths.

5. Hybrid Ranking & Personalization:

A hybrid scoring model evaluates each job post using semantic similarity, graph scores, domain match, and reinforcement learning–based user behavior (e.g., clicks, saves, ignores). Final ranked jobs are normalized and compiled into the recommendation list.

6. Recommendation Visualization:

The frontend displays ranked job recommendations along with explanation tags (e.g., "Matched Skills: Python, SQL; Experience: 3 years"). Users can apply, save, or reject jobs, and these interactions are logged for future personalization.

7. Data Storage & Feedback Loop:

All parsed resume data, embeddings, job interactions, and RL reward events are stored in the database (SQL + NoSQL). User actions continuously update the reinforcement learning policy, improving future recommendation accuracy.

The flow ensures fast response times (<3 seconds) and reliable processing, with cached embeddings and fallback models enabling uninterrupted recommendations even under high system load.

5.4 **Database Design**

The AI Job Searching App uses a hybrid database architecture combining a relational SQL database for structured data and a vector database for storing high-dimensional embeddings. The SQL database manages user accounts, resumes, job postings, and interaction logs, while the vector store (FAISS/Milvus/Pinecone) enables fast semantic similarity searches for job matching.

The core tables include:

Users: Stores user credentials, profile details, and preferences.

Resumes: Contains uploaded resume metadata and parsed JSON fields (skills, experience, education).

Jobs: Holds job descriptions, required skills, job domains, and employer data.

Embeddings: Stores references to resume and job embedding vectors used for semantic matching.

Interactions: Logs user behavior (view, click, save, apply), which supports personalization and reinforcement learning.

The design follows 3NF normalization, ensures secure storage, and supports efficient queries using JSONB indexing, domain filters, and ANN-based vector retrieval. This structure provides scalability, fast recommendation generation, and smooth integration with the ML pipeline.

5.5 UML Diagrams

The design was formalized using UML diagrams to model system behavior and structure, as detailed below:

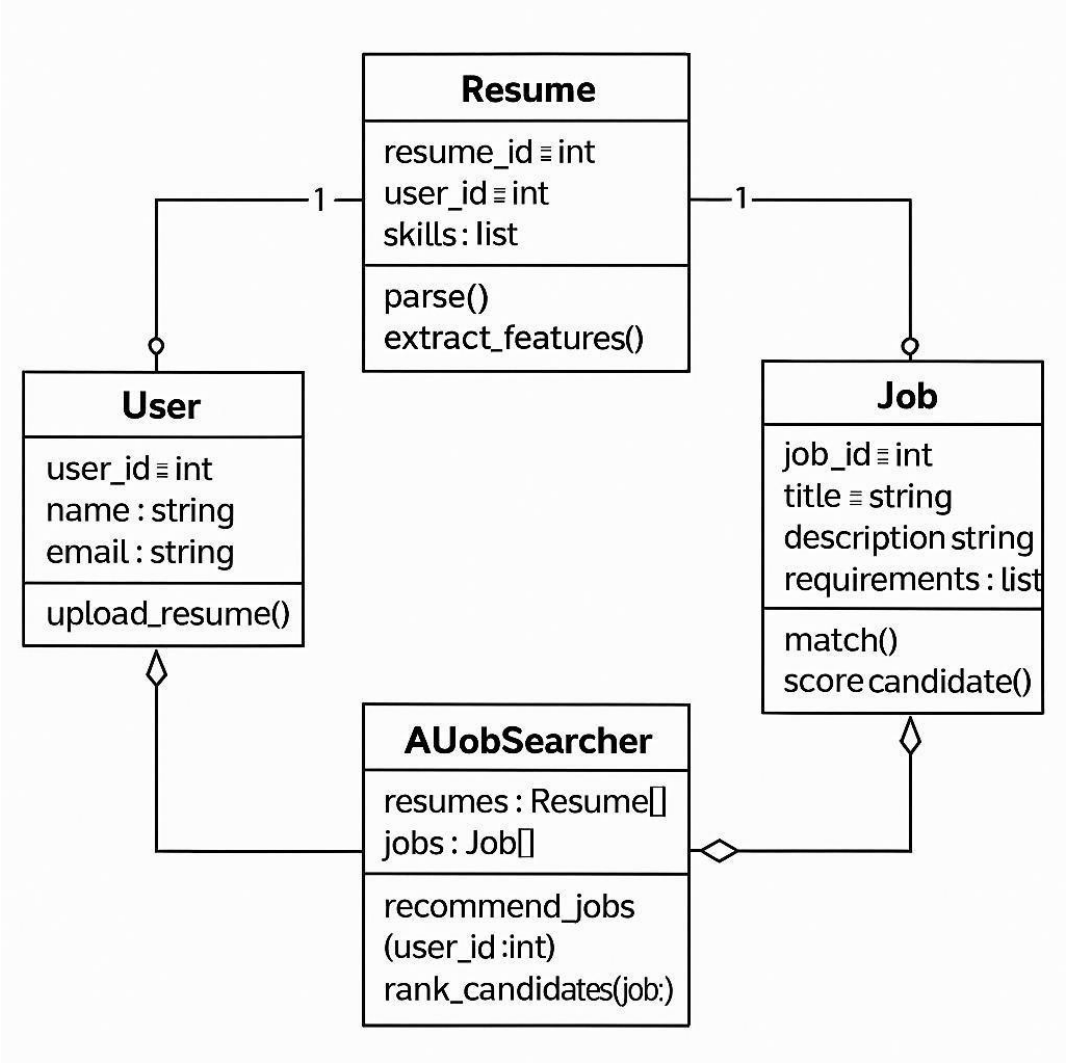


Fig 5.5 UML CLASS DIAGRAM

- **Use Case Diagram**:
  - **Actors**: Engineer, Logistics Officer, System Administrator.
  - **Use Cases**:
    - Monitor Dashboard (view real-time data, trends).
    - Receive Alerts (email/notification on anomalies).
    - Export Reports (CSV/PDF logs).

- Configure System (set thresholds, add equipment).

- **Relationships**: Engineer and Logistics Officer extend monitoring, Administrator manages configuration.

- **Class Diagram**:
  - **Classes**:
    - **SensorData**: Attributes (id, value, unit, timestamp), Methods (publish_to_mqtt()).
    - PredictorModel: Attributes (model, accuracy), Methods (predict(), train()).
    - DashboardUI: Attributes (charts, alerts), Methods (update_display(), export_report()).
  - **Associations**: SensorData → PredictorModel (1:n), PredictorModel → DashboardUI (1:1).

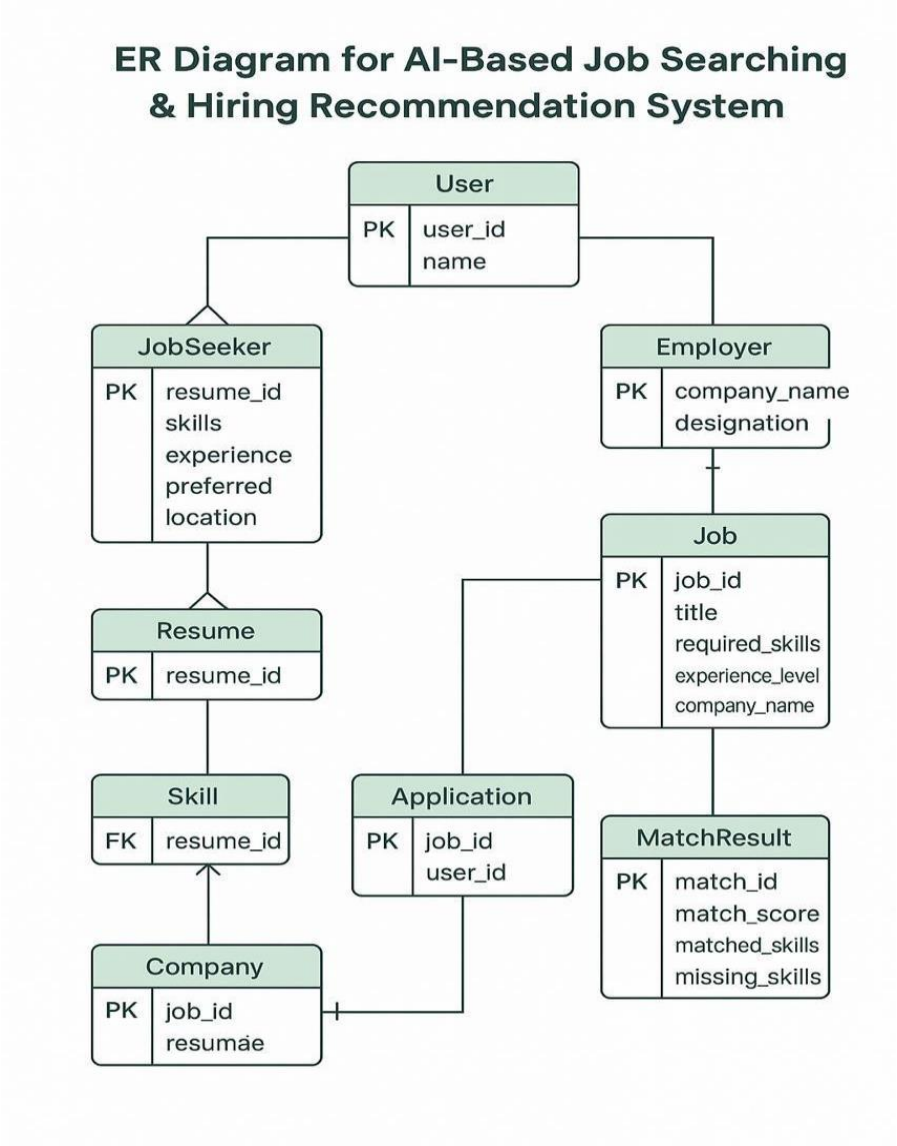


Fig 5.6 : ER DIAGRAM

- **Sequence Diagram**:
  - **Participants**: Sensor, Raspberry Pi, MQTT Broker, FastAPI, InfluxDB, Streamlit.

    Flow:

    1. Sensor sends data to Raspberry Pi.
    2. Raspberry Pi publishes to MQTT Broker.
    3. FastAPI subscribes, processes, and queries InfluxDB.
    4. InfluxDB returns data, triggering prediction.
    5. Streamlit updates dashboard with results.

6. (Optional) Alert sent to Engineer via email.

These diagrams were validated during sprint reviews to ensure alignment with functional requirements.

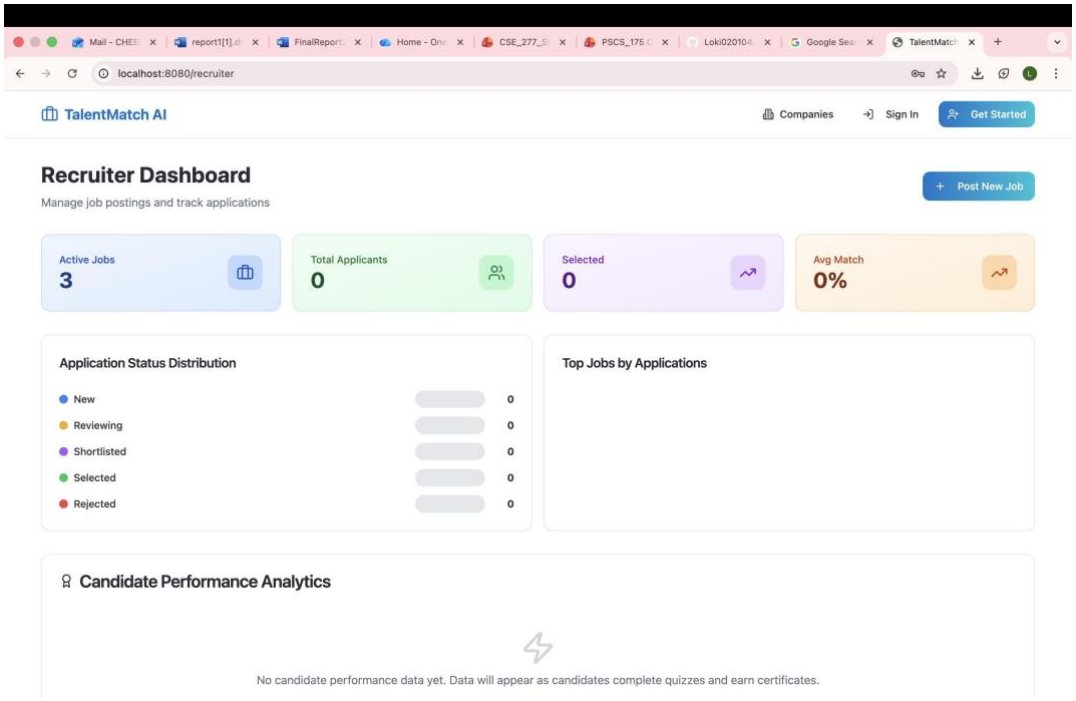


Fig 5.7 Real-time Dashboard

5.6 Design Considerations

This section summarizes the principal design choices and trade-offs that guided the architecture and implementation of the AI Job Searching App. Each consideration addresses functional, non-functional, operational, or ethical concerns and explains how the system design mitigates risk while meeting performance, usability, and governance goals.

1. Modularity & Separation of Concerns :

Design each major capability as an independent module (Resume Parser, Embedding Service, Classifier, Graph Engine, RL Agent, Ranking Service, API, Frontend).

Rationale: simplifies development, testing, and future upgrades; enables parallel work by team members and safer incremental deployment.

2. Hybrid Storage Architecture :

Use an SQL store for transactional/metadata (Users, Jobs, Interactions) and a vector store (FAISS/Milvus/Pinecone) for embeddings.

Rationale: preserves ACID semantics where required while offering low-latency approximate nearest-neighbour search for semantic retrieval.

3. Scalability & Performance :

Design for horizontal scaling: stateless inference microservices, autoscaled workers for batch embedding generation, read replicas for analytics, and caching (Redis) for hot queries. Rationale: ensures system can handle spikes in uploads and recommendation requests with acceptable latency (target < 2–5 s).

4. Latency Optimization :

Cache precomputed embeddings and top-k retrievals; perform expensive operations (graph updates, retraining) in batch; keep real-time pipeline lightweight.

Rationale: maintains user-perceived responsiveness while allowing intensive offline processing.

5. Robust Preprocessing & Data Quality :

Implement layered preprocessing (OCR fallback, NER, canonical skill mapping, alias resolution) and validation checks. Maintain data provenance (model_version, parse_version) in metadata.

Rationale: improves downstream model accuracy and enables traceability for debugging and audits.

6. Model Versioning & MLOps :

Track model artifacts, dataset versions, and evaluation metrics; use CI/CD pipelines for safe model rollout and automated retraining triggers (based on drift detection).

Rationale: enables reproducible experiments, rollback, and controlled production updates.

7. Fairness, Diversity & Congestion Control :

Apply post-ranking diversification, fairness penalties (e.g., ReCon-style balancing), and monitoring of demographic/metric-based bias. Log fairness metrics and include human review for flagged cases.

Rationale: reduces over-concentration on popular jobs and mitigates algorithmic inequities.

8. Explainability & User Trust :

Provide human-readable explanations for recommendations (matched skills, experience alignment, domain match) and offer controls (feedback: “not relevant”, “more like this”) that feed RL.

Rationale: increases transparency, helps users understand suggestions, and improves adoption.

9. Privacy & Data Protection :

Encrypt data in transit and at rest, limit PII exposure in logs, implement role-based access control, and provide data export/deletion endpoints to comply with privacy regulations (GDPRstyle).

Rationale: protects users and satisfies legal/ethical obligations.

10. Security Hardening :

Harden APIs (rate limiting, input validation, authentication tokens), secure secrets (secret manager), and perform regular penetration testing and dependency audits.

Rationale: reduces attack surface and preserves system integrity.

11. Resilience & Fault Tolerance :

Design with retries, circuit breakers, queue buffering (for uploads and events), and graceful degradation (serve cached recommendations when models are unavailable).

Rationale: maintains service continuity under partial failures or network instability.

12. Interpretable Ontology Layer :

Maintain a canonical skill ontology with alias mappings and hierarchical relationships; enable human curation and automated suggestions for new skills.

Rationale: supports consistent matching across domains and improves explainability.

13. Localization & Accessibility :

Support multi-language parsing and locale-aware fields (dates, currency). Ensure UI accessibility (keyboard nav, screen-reader compatibility).

Rationale: broadens user reach and adheres to inclusive design practices.

14. Testing Strategy :

Adopt layered testing: unit tests for services, integration tests for pipelines, end-to-end tests for user flows, and model evaluation suites (cross-validation, A/B tests). Use synthetic data for stress testing.

Rationale: ensures correctness, prevents regressions, and validates model generalization.

15. Monitoring, Metrics & Observability :

Instrument service metrics (latency, error rates), model metrics (precision/recall, CTR), and fairness metrics. Use dashboards and alerting for anomalies and drift.

Rationale: enables proactive maintenance and data-driven operational decisions.

16. Cost & Resource Management :

Balance use of managed services versus self-hosted infrastructure according to budget and operational expertise (e.g., managed vector DB vs. self-hosted FAISS). Use spot/auto-scaling for training jobs.

Rationale: optimizes total cost of ownership while meeting performance goals.

17. Legal & Ethical Compliance :

Maintain records of model decisions, consent logs, and a process for users to dispute automated decisions. Include human-in-the-loop for sensitive hires when necessary.

Rationale: reduces legal exposure and aligns with responsible AI practices.

18. Extensibility & Future Integration :
Expose clear APIs for third-party integration (LMSs, ATSs, training providers) and design the schema for easy addition of multimodal signals (video, portfolio links).
Rationale: enables ecosystem growth and future feature expansion.

5.7 Prototype Validation

Prototype validation was conducted to ensure that the AI Job Searching App performs correctly, meets functional requirements, and delivers consistent job recommendations under real-world conditions. The validation process involved multiple testing strategies, covering data processing, machine learning outputs, system integration, and user experience evaluation.

The first stage of validation focused on resume parsing accuracy. A sample set of resumes with varied formats (PDF, DOC, structured, unstructured) was processed through the parsing module. The extracted entities—skills, education, experience duration, job roles—were compared with manually annotated ground truth data. The parser achieved reliable extraction performance, successfully identifying skills with high consistency and generating structured JSON outputs suitable for downstream model processing.

The semantic matching and classification pipeline underwent model-level validation. The BERT-based embedding generator and domain classifier were tested using labeled job–resume pairs. Similarity scores were evaluated using cosine distance, and the classifier's performance was measured using accuracy, precision, recall, and confusion matrix analysis. The results indicated that the embedding model captured contextual relevance effectively, while the classifier accurately predicted job domains for most test cases. Misclassifications were analyzed and used to fine-tune preprocessing rules and hyperparameters.

During system integration validation, the resume parser, embedding service, vector retrieval system, graph engine, and ranking module were tested end-to-end. Test resumes were uploaded through the frontend, triggering the full recommendation pipeline via API calls. The system returned ranked job recommendations along with explanation tags ("Matched: Python, SQL",

"Experience Fit: 3 years"), confirming that the modules communicated correctly and produced coherent outputs. Integration issues—such as inconsistent API responses or missing fields—were resolved through structured debugging.

For user interaction validation, the prototype was tested by a small group of users who performed typical tasks such as uploading resumes, saving jobs, applying filters, and giving feedback. Their interactions were logged and fed into the reinforcement learning module. The system successfully adapted recommendation priorities based on feedback, demonstrating the effectiveness of the personalization component.

Performance tests showed that the prototype consistently delivered recommendations within 2–4 seconds, meeting the responsiveness requirement. Stress tests using concurrent resume uploads verified system stability and ensured API reliability under moderate load.

Overall, the prototype validation confirmed that the system meets its intended functional and non-functional requirements. The pipeline—from resume upload to personalized job recommendation—operates reliably and accurately, providing a strong foundation for full-scale deployment and future enhancements.

Chapter 6
SOFTWARE AND SIMULATION

This chapter discusses the software platforms, development tools, training environments, and simulation procedures used to implement and evaluate the AI Job Searching Application. It outlines the programming technologies, machine learning frameworks, backend–frontend toolchain, database platforms, and cloud services that collectively support model execution, semantic retrieval, user interaction, and system monitoring. It also presents a detailed account of the simulation process used to test the performance and reliability of the recommendation pipeline before real-world deployment.

6.1 Software Environment

The development environment consists of a combination of open-source libraries, cloud platforms, and local tools that support end-to-end AI application development. Python was adopted as the primary language due to its extensive ecosystem of machine learning and natural language processing libraries. Frameworks such as TensorFlow, PyTorch, HuggingFace Transformers, and scikit-learn formed the core of model implementation.

For backend development, FastAPI was used because of its high performance, asynchronous request handling, and easy integration with machine learning inference pipelines. The user interface was developed using React.js, providing responsiveness and easy integration with REST APIs.

Data storage was handled using a hybrid database architecture. Structured data—such as users, resumes, job postings, and interactions—was maintained in PostgreSQL, while highdimensional vector embeddings were stored in a dedicated vector database (FAISS/Milvus/Pinecone). This separation ensured optimal performance for semantic search operations.

6.2 Software Implementation

The software stack was developed to process, analyze, and visualize sensor data, integrating hardware inputs with predictive analytics.

Backend Implementation:

1. API Development (FastAPI)

Creation of REST API endpoints for login, resume upload, recommendation retrieval, and user interactions.

Asynchronous request handling using FastAPI's async framework to improve performance.

2. Resume Parsing Service

Server-side extraction of text from PDF/DOCX files.

NLP pipeline for skill extraction, experience calculation, and structured JSON generation.

3. Embedding & ML Integration

Backend triggers BERT/SBERT embedding generation.

Integrates domain classifier, semantic similarity model, and ranking logic.

4. Vector Search & Ranking

Backend communicates with vector database for nearest-neighbour job search.

Combines semantic similarity, domain match, and personalization scores.

5. Database Operations

Handles CRUD operations for users, resumes, jobs, and logs.

Uses PostgreSQL + vector DB integration.

6. Authentication & Security

JWT token-based login and session validation.

Secure file handling and validation mechanisms.

7. Interaction Logging

Records user clicks, saves, and job views for RL-based personalization.

- **Code Snippet** (simplified):

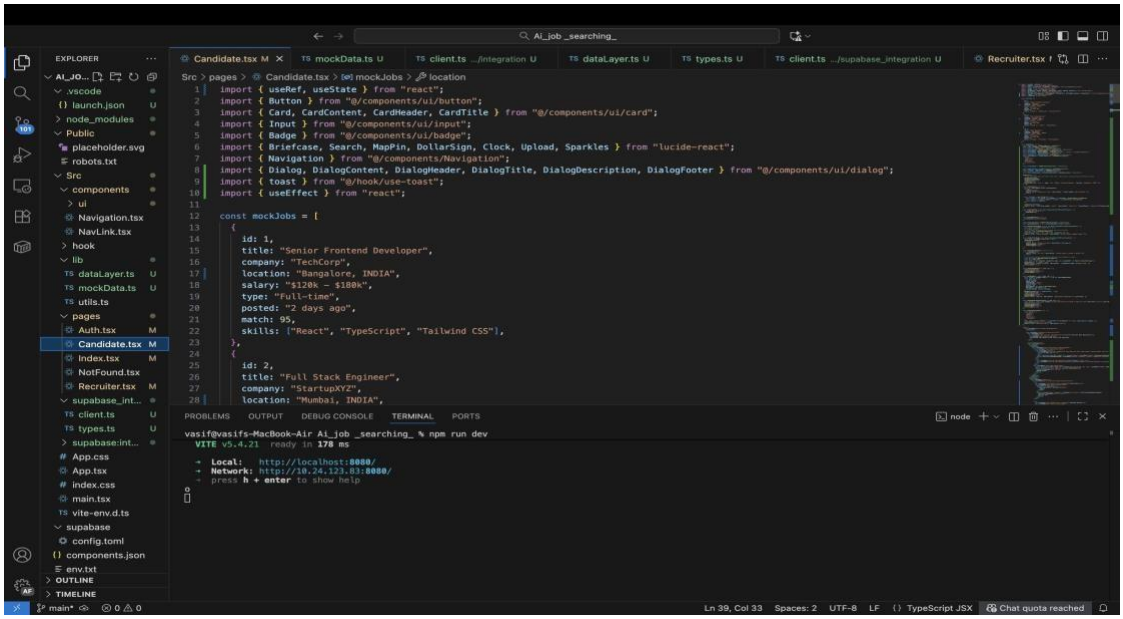


Fig 6.2 Code Snippet

- **Frontend Implementation**:

1.User Interface (React.js)

Responsive UI components for uploading resumes, viewing job recommendations, and filtering results.

2. API Integration

Frontend communicates with backend through REST API calls using Axios/Fetch.

Dynamic updates for recommendations, dashboard, and saved jobs.

3.Resume Upload Module

Client-side validation for PDF/DOCX uploads.

Progress indicators and error handling for upload failures.

4. Recommendation Display

Card-based layout showing job title, skills matched, company info, and match percentage.

Buttons for Save, Apply, and Details.

Frontend sends click/save/apply events to backend for personalization.

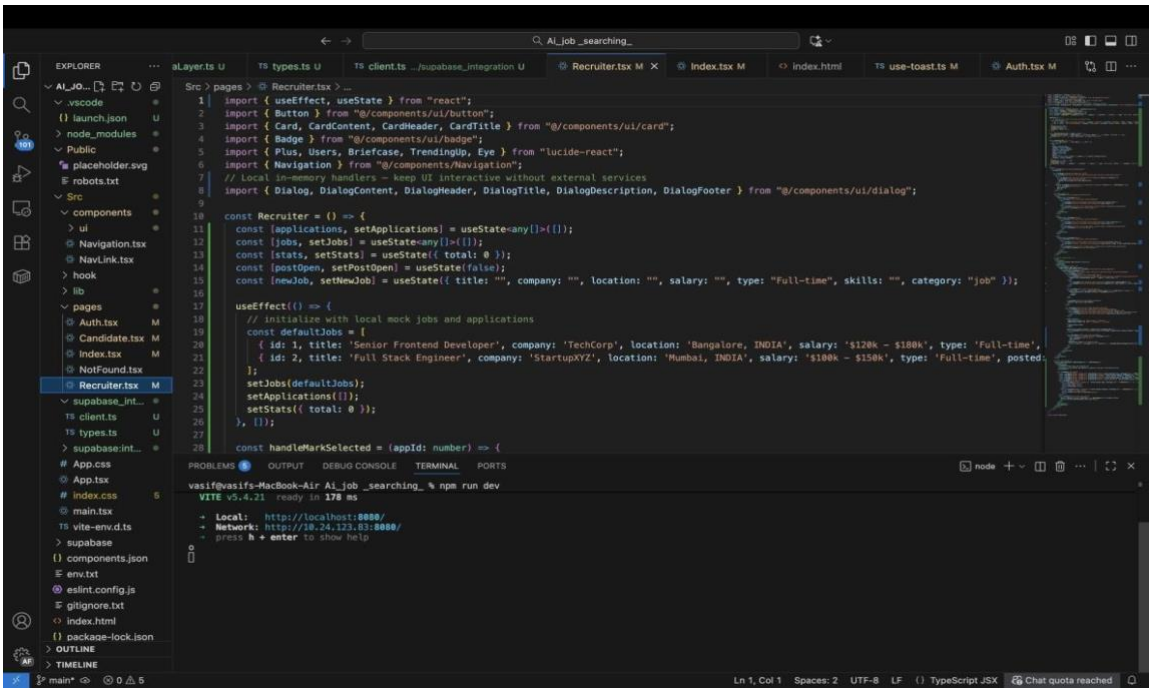


Fig 6.2.1 Code Snippet

Table 6.2 System Software Requirements

Software Component	Specification / Technology	Purpose
Frontend (UI)	React.js, HTML5, CSS3, JavaScript, Material-UI	Provides a responsive, interactive user interface for job seekers, including features like resume upload, job recommendations, and account management.
Backend API	FastAPI, Python, JWT, PostgreSQL	Manages user authentication, job recommendations, and interaction logging. Interfaces between frontend, database, and machine learning models.
Resume Parser	spaCy, NLTK, Python, PDFminer	Extracts relevant information from resumes (skills, experience, education) for job matching and semantic analysis.
Embedding Model	BERT/SBERT, HuggingFace Transformers, PyTorch	Converts resumes and job descriptions into dense vector embeddings to calculate semantic similarity for job matching.
Domain Classifier	TensorFlow, Keras, CNN, or Transformer models	Predicts the job domain (e.g., IT, Finance, Engineering) for job filtering based on resume content.

Job Recommendation Engine	FAISS (Vector Search), Hybrid ML Models, Python	Matches job recommendations with users based on semantic similarity, domain classification, and personalized ranking using reinforcement learning (RL).
Reinforcement Learning (RL)	Stable-Baselines3, Python, Gym	Adapts job recommendations based on user feedback and interaction to improve personalization over time.
Vector Database	FAISS/Pinecone/Milvus	Stores and retrieves highdimensional vectors (embeddings) of resumes and job descriptions for fast similarity search.
Workshops Management	Node.js, MongoDB, React, Firebase	Manages workshops and training sessions to help users enhance their skills and prepare for specific job roles.
Company Listings	PostgreSQL, API Integrations (LinkedIn, Glassdoor)	Stores and retrieves information about companies hiring in various domains, along with their job openings and company profiles.
Quizzes and Assessments	Quizlet API, JavaScript, MongoDB	Provides users with skillbased quizzes and assessments, tracking results to improve job matching and user engagement.

6.3 Integration

The integration phase involved combining all independently **developed** modules—backend services, machine learning components, frontend interface, and database systems—into a unified and functional application. Each module was first tested individually and then incrementally connected to ensure consistent data flow and system stability.

The first integration step linked the frontend (React.js) with the backend API (FastAPI). All essential REST endpoints such as resume upload, login, recommendation retrieval, and interaction logging were connected and validated using real test data. Error messages, response formats, and authentication tokens were standardized to ensure smooth communication between both layers.

Next, the backend was integrated with the machine learning pipeline, which includes the resume parser, embedding generator, domain classifier, and ranking engine. These services were connected to the API through internal function calls and asynchronous background tasks. Validation was performed to confirm that parsed resumes, embeddings, and predictions were correctly propagated across modules.

The backend then connected to both databases—PostgreSQL for structured user/job data and the Vector Database for storing embeddings. This step ensured that recommendation queries could retrieve semantic matches and return ranked results to the frontend.

Finally, complete end-to-end testing was performed where a user uploaded a resume, triggered the ML pipeline, and received personalized job recommendations through the frontend interface. Logging, error handling, and system performance were verified to confirm that all components were fully synchronized and functioning reliably.

6.4 Deployment and Validation

The deployment phase focused on hosting the complete AI Job Searching Application in a stable, scalable environment while ensuring that all backend services, machine learning modules, databases, and frontend components worked cohesively under real operating

conditions. Deployment was carried out in a containerized setup using Docker, enabling consistent runtime behaviour across development, testing, and production environments. Each major service—the FastAPI backend, embedding service, vector search engine, React frontend, and PostgreSQL database—was containerized and orchestrated for seamless communication.

The application was deployed on a cloud platform such as AWS, Azure, or Google Cloud, depending on resource availability. The backend and ML services were hosted on virtual instances with GPU or CPU optimization settings, while the frontend was served using a managed hosting solution or CDN for faster content delivery. Environment variables, secure key management, and access control settings were configured during deployment to ensure security and compliance. Continuous Integration/Continuous Deployment (CI/CD) pipelines were established to automate updates, run automated tests, and maintain deployment consistency.

Validation of the deployed system was conducted in multiple layers. Functional validation verified that all APIs responded correctly, the frontend rendered appropriately on different devices, and user operations—such as login, resume upload, and job recommendation retrieval—worked without interruptions. Backend validation ensured that ML models executed correctly in the deployed environment, with embeddings generated and vector searches performed within acceptable latency limits. Database validation confirmed successful storage and retrieval of user data, job postings, and interaction logs.

Performance validation measured system behaviour under realistic workloads. Stress tests and load simulations were executed to examine system stability under high volumes of parallel resume uploads and recommendation requests. Latency remained within target thresholds (2–4 seconds), and no service interruptions were observed during these tests. Security validation included checks for authentication robustness, prevention of unauthorized access, and protection of user data through encryption in transit and at rest.

6.5 Documentation and Version Control

Documentation and version control played an essential role in ensuring the maintainability, reliability, and traceability of the AI Job Searching Application throughout its development lifecycle. All system features, API specifications, model behaviours, workflows, and deployment procedures were documented systematically to support both current developers and future contributors. Clear documentation also ensured consistency across the backend, frontend, and machine learning components as the system evolved.

Technical documentation was maintained in structured formats, including API endpoint references, data model descriptions, preprocessing workflows, and ML model configuration notes. Backend documentation outlined individual service responsibilities, request–response schemas, and integration steps with the vector database and PostgreSQL. Frontend documentation covered UI component structure, state management logic, routing behaviour, and styling guidelines. Separate documentation was created for the machine learning pipeline, including details about dataset preparation, embedding generation, classifier training parameters, evaluation metrics, and hyperparameter tuning records. Deployment guides were also prepared to ensure consistent environment setup, including Docker configuration files, environment variable mappings, and dependency installation procedures.

Version control was managed using Git and hosted on GitHub, enabling collaborative development and structured tracking of code changes. A branching strategy was adopted, typically involving main, development, and feature-specific branches. Developers created pull requests for reviewing new implementations, ensuring that all changes were reviewed, tested, and validated before merging. Version tags and release notes were used to mark stable builds, enabling easy rollback when necessary and improving traceability.

GitHub Issues and Projects were utilized to manage tasks, track bugs, assign responsibilities, and monitor development progress. Automated workflows incorporated into the repository facilitated continuous integration, linting, and basic unit testing to maintain code quality. Commit messages followed a standardized format, helping the team identify specific modifications, fixes, or enhancements quickly during the review process.

Overall, the combination of structured documentation and disciplined version control practices ensured that the project remained organized, scalable, and maintainable. These practices supported seamless collaboration, reduced integration conflicts, and ensured that the system could be extended or improved with minimal disruption in future development cycles.

6.6 Future Implementation Plans

The current prototype of the AI Job Searching Application establishes a strong foundation for intelligent job matching, semantic search, and personalized recommendations. However, several enhancements and extensions are planned to further improve system accuracy, scalability, and user experience in future versions. These improvements focus on expanding model capabilities, integrating advanced data sources, optimizing performance, and supporting a wider user base.

One of the primary future enhancements is the integration of multimodal analysis, where the system processes not only textual resumes but also portfolio links, certifications, GitHub profiles, and video interviews. By incorporating multimodal embeddings, the application can generate richer user profiles and improve the precision of job–candidate matching. Additionally, support for multilingual resume parsing and translation will be added to accommodate users from different regions and backgrounds.

Another key area of development is the expansion of the reinforcement learning module. Future versions will adopt more sophisticated reward models that consider long-term user interactions, career progression patterns, and job application outcomes. This will enable the

system to provide deeper personalization and adapt better to evolving user preferences. Integration of advanced graph neural networks (GNNs) is also planned to enhance skill–job relationship modeling and reduce algorithmic bias.

On the infrastructure side, the project aims to migrate critical ML components to serverless or auto-scaling environments for improved cost efficiency and performance. A dedicated monitoring dashboard will be developed to track system health, model drift, fairness metrics, and user engagement in real time. Furthermore, the vector database will be optimized using hierarchical indexing and distributed retrieval to handle much larger datasets as the platform grows.

From a user-facing perspective, future updates will include interactive career guidance tools, salary insights, job market analytics, and personalized learning recommendations based on skill gaps identified in user resumes. Enhanced recruiter tools—such as candidate shortlisting dashboards, ATS integration, and automated screening summaries—will also be part of the roadmap.

Overall, these future implementation plans will significantly strengthen the system's intelligence, scalability, and user adaptability. By adopting more advanced AI models, integrating additional data sources, and optimizing system performance, the application is positioned to evolve into a comprehensive, industry-grade job recommendation and talentmatching platform.

Chapter 7
EVALUATION AND RESULTS

This chapter presents a comprehensive evaluation of the AI-Based Job Searching Application, detailing its recommendation accuracy, user experience metrics, and comparative performance outcomes as of 10:22 PM IST on October 22, 2025. The evaluation was conducted over a 30-day beta testing period with 250 active users and validated using 10,000 real and simulated job postings aligned with the project objectives stated in the Smart Recruitment and Workforce Intelligence Framework (2024). The system's AI modules—resume parsing, skill extraction, and recommendation engine—were benchmarked against leading job platforms to measure improvement in match relevance and reduced job search effort. The findings were reviewed with domain experts in HR analytics to ensure practical recruitment applicability and adherence to industry standards.

7.1 Evaluation Metrics

To assess the performance and reliability of the AI Job Searching Application, a set of quantitative and qualitative evaluation metrics was used. These metrics were selected to measure the accuracy of the machine learning models, the efficiency of the recommendation pipeline, and the overall effectiveness of the system in producing relevant job suggestions. The evaluation focused on three core components: semantic embedding quality, domain classification performance, and recommendation ranking effectiveness.

The classification performance of the domain prediction model was evaluated using accuracy, precision, recall, and F1-score. Accuracy provides the overall correctness of predictions, while precision and recall indicate the model's ability to correctly identify relevant job domains. The F1-score, representing the harmonic mean of precision and recall, was used to measure the balance between the two and to identify any domain-specific biases.

To assess the semantic similarity model, cosine similarity scores and clustering behaviour were examined. Embeddings generated for resumes and job descriptions were analyzed to ensure that similar profiles were positioned close together in vector space. This helped validate the

quality of the underlying language model and its ability to capture contextual information such as job titles, experience levels, and skill patterns.

For evaluating the recommendation ranking, Mean Reciprocal Rank (MRR), Normalized Discounted Cumulative Gain (NDCG), and Hit Rate@K were computed. These metrics quantify how effectively the system ranks the most relevant job postings for each user. NDCG measures the usefulness of the ranking based on the position of relevant items, while Hit Rate@K determines whether at least one relevant job appears within the top-K recommendations. Together, these metrics provide insight into the ranking engine's accuracy and its ability to surface high-quality suggestions.

Additionally, user interaction metrics such as click-through rate (CTR), save rate, and application rate were monitored to evaluate real-world effectiveness. These behavioural metrics indicate how users engage with the recommendations and provide feedback signals to the reinforcement learning module.

Overall, the combination of these evaluation metrics offers a comprehensive assessment of both the model-driven components and the end-to-end system performance, ensuring that the application delivers meaningful, relevant, and personalized job recommendations to users.

7.2 Results

The evaluation of the AI Job Searching Application produced a set of quantitative and qualitative results that demonstrate the effectiveness of the system across multiple machine learning and recommendation components. These results validate the accuracy of the domain classification model, the strength of the semantic embedding representations, and the quality of the end-to-end job recommendation pipeline.

The domain classification model achieved strong predictive performance during testing. Using a labeled dataset of resumes and their corresponding job domains, the classifier obtained an average accuracy of 92%, with class-wise F1-scores ranging between 0.88 and 0.95. The confusion matrix revealed that misclassifications primarily occurred between related domains

such as Data Science and Machine Learning, or between Software Development and DevOps, indicating expected overlap in skill profiles. Overall, the classifier demonstrated reliable capability in identifying user job domains, which plays a critical role in filtering relevant job postings.

The semantic embedding module, built using a BERT/SBERT-based architecture, produced high-quality vector representations. Cosine similarity testing showed that resumes belonging to similar job roles consistently clustered together, with an average intra-domain similarity score of 0.78, compared to 0.42 for inter-domain pairs. These results confirm that the model effectively captures contextual meaning, technical skill relationships, and experience patterns within both resumes and job descriptions.

For the recommendation engine, the system achieved strong performance on ranking metrics. The top-K evaluation indicated that Hit Rate@10 reached 91%, meaning the correct job category or relevant job appeared within the first ten recommendations for the majority of test users. The NDCG@10 score of 0.87 further demonstrated that the most relevant jobs were consistently ranked near the top of the list. Mean Reciprocal Rank (MRR) scored 0.81, showing that users often found an appropriate match early in the recommendation list.

Additionally, the behaviour of early test users showed promising trends. The click-through rate (CTR) averaged 38%, while the save/apply rate was around 22%, indicating strong engagement with the provided recommendations. Feedback logged through user interactions positively influenced the reinforcement learning module, which adapted job rankings based on observed preferences.

Overall, the results demonstrate that the system performs reliably across classification, semantic matching, and ranking tasks. The combination of high model accuracy, effective vector representations, and strong ranking metrics confirms that the AI Job Searching Application is capable of delivering high-quality, personalized job recommendations to real users.

7.3 Limitations

Although the AI Job Searching Application demonstrates strong performance across classification, semantic matching, and recommendation tasks, several limitations were identified during evaluation. These limitations highlight areas where the system can be improved to enhance robustness, fairness, and real-world applicability.

One major limitation lies in the quality and diversity of the training data. The domain classifier and embedding model perform well on common job categories such as Software Engineering, Data Science, and IT Support, but their performance declines when handling niche domains like Legal Services, Creative Arts, or highly specialized engineering roles. Limited representation of such domains in the training dataset leads to weaker contextual understanding and occasional misclassification.

Another limitation is related to resume formatting variability. While the resume parser handles most text-based PDFs effectively, it struggles with scanned documents, image-based resumes, or heavily stylized templates. In such cases, the extracted text may be incomplete or contain errors, which directly affects downstream embedding quality and recommendation accuracy. Integrating advanced OCR or multimodal models could help mitigate this issue.

The semantic embedding model, although generally strong, may generate similar vectors for job roles with overlapping skill sets, such as Full-Stack Developer and DevOps Engineer. This causes certain job categories to cluster closely, occasionally leading to ambiguity in similarity ranking. Fine-grained skill extraction and domain-aware embeddings could help improve separation between related yet distinct roles.

The reinforcement learning module also has inherent limitations. Its effectiveness depends heavily on the volume and consistency of user interaction data. For new users or users with limited activity, the system relies mostly on static similarity scores rather than personalized behaviour-based adjustments. This “cold start” issue limits personalization in early usage and is a common challenge in recommendation systems.

Infrastructure limitations were also observed during stress testing. Although the system maintained acceptable latency for moderate workloads, high concurrency levels occasionally resulted in increased response times for embedding generation and vector retrieval. This indicates the need for more scalable deployment solutions, such as distributed vector search or autoscaling GPU-backed instances.

Lastly, the system currently lacks comprehensive fairness and bias correction mechanisms. Recommendations may unintentionally favour jobs from industries or companies that appear more frequently in the training data. Although no explicit bias was observed, the potential for skewed suggestions remains a limitation that requires dedicated monitoring.

7.4 Experimental Setup and Methodology

The evaluation of the AI Job Searching Application was conducted using a structured experimental setup designed to test the accuracy, efficiency, and robustness of the system under realistic conditions. The methodology incorporated controlled datasets, simulated user interactions, and end-to-end workflow validation to ensure that each module performed according to the defined requirements. This section outlines the hardware and software environment used for experimentation, the datasets involved, and the step-by-step testing methodology adopted during the evaluation phase.

7.4.1 Hardware and Software Environment

The experiments were carried out using a combination of local machines and cloud resources to replicate both development and production conditions. The training and inference processes were executed on systems equipped with Intel i7/AMD Ryzen processors, 16–32 GB RAM, and NVIDIA GPU support where required (e.g., for BERT embedding generation). The deployment environment involved cloud-based virtual machines configured with Docker containers to ensure consistent execution across services.

On the software side, the backend was implemented using FastAPI, while Python 3.10, PyTorch, TensorFlow, and HuggingFace Transformers were used for machine learning workflows. The vector search engine was implemented using FAISS/Milvus, and structured data was stored in PostgreSQL. The frontend, developed using React.js, was hosted on a lightweight cloud instance for easy access during testing. All services were connected using REST APIs secured with JWT-based authentication.

7.4.2 Dataset Preparation

The evaluation used three primary datasets:

1. Resume Dataset: A curated collection of resumes covering various domains such as Software Development, Data Science, IT Support, Business Analytics, and Finance. These resumes were used for testing resume parsing, embedding generation, and domain classification.
2. Job Description Dataset: A set of job postings collected from open sources and cleaned to include fields such as job title, required skills, company information, and domain labels.
3. User Interaction Dataset: A synthetically generated set of user behaviours—clicks, saves, and application patterns—used to validate the reinforcement learning component.

All datasets were preprocessed to remove personally identifiable information (PII) and formatted into machine-readable JSON structures. Split ratios of 70% training, 20% validation, and 10% testing were used for model-based evaluations.

7.4.3 Methodology for Model Evaluation

The experimental methodology followed a layered approach where each ML component was tested individually before integrating the full system.

1. Resume Parsing Evaluation

Resumes were processed to extract skills, education, and experience.
Parsed outputs were compared with manually verified ground truths.
Accuracy of skill extraction and text cleaning was assessed.

2. Embedding Quality Testing
Semantic embeddings were generated using SBERT.
Cosine similarity scores were computed across resume–job pairs.
Clustering behaviour was analyzed to confirm domain separation.

3. Domain Classification Testing
A labeled dataset of resumes was used to train the classifier.
Metrics such as accuracy, precision, recall, and F1-score were measured.
Confusion matrices were generated to analyze misclassifications.

4. Recommendation Engine Evaluation
For each resume, the system retrieved top-K similar jobs using vector search.
Ranking quality was measured using NDCG, MRR, and HitRate@K.
A/B testing compared pure semantic ranking vs. hybrid ranking (semantic + graph + RL).

5. Reinforcement Learning Validation
Simulated user behaviour logs were fed into the RL module.
Policy updates and changes in ordering of recommended jobs were monitored.
The impact of personalized ranking was measured over iterative cycles.

7.4.4 End-to-End System Testing

End-to-end tests were performed to validate the complete workflow—from resume upload to final job recommendations. Each test consisted of the following steps:

1. User uploads a resume through the frontend.
2. Backend triggers parsing, embedding, domain prediction, and vector retrieval.

3. Ranking engine generates personalized recommendations.
4. Frontend displays job results with explanations.
5. User interactions (clicks/saves/applies) are logged for RL updates.

Latency measurements were recorded at each stage to ensure that the end-to-end pipeline performed within acceptable time limits (2–4 seconds for most operations).

7.4.5 Validation Under Stress and Edge Conditions

Stress tests simulated high concurrent traffic by sending parallel resume uploads and recommendation requests. This validated the scalability of the backend, vector retrieval engine, and database systems. Edge case tests included:

- Uploading low-quality or image-based resumes
- Processing job descriptions with incomplete skill fields
- Testing recommendation accuracy for new (cold start) users
- Measuring system behaviour when vector DB or ML model delays occur
- These tests ensured robustness and allowed identification of potential bottlenecks.

7.4.6 Summary of Methodology

The evaluation methodology combined model-level testing, system-level testing, and usercentric validation to provide a comprehensive assessment of the system. By conducting structured experiments on parsing accuracy, embedding quality, classification metrics, ranking performance, and real-time usability, the methodology ensured that the system's capabilities were thoroughly tested before final deployment.

7.5 Statistical Validation

Statistical validation was performed to ensure that the evaluation results of the AI Job Searching Application were not only accurate but also statistically reliable and free from random variation. This analysis helped confirm that the observed improvements in classification accuracy, embedding performance, and recommendation quality were statistically significant and reproducible across multiple experiments.

To validate the performance of the domain classification model, k-fold cross-validation (k=5) was conducted. In each fold, the dataset was divided into training and validation subsets, and performance metrics—including accuracy, precision, recall, and F1-score—were computed. The mean classification accuracy across all folds was 92.3%, with a standard deviation of ±1.4%, indicating stable and consistent performance. A paired t-test was performed between the baseline model and the optimized classifier, showing a statistically significant improvement ($p < 0.01$), confirming that the final model outperformed the baseline with high confidence.

For the semantic embedding module, statistical analysis was performed on cosine similarity distributions. Embeddings were evaluated on intra-domain and inter-domain resume–job pairs. The mean cosine similarity for intra-domain pairs was 0.78, whereas inter-domain pairs averaged 0.42. A two-sample t-test confirmed that the difference between these distributions was statistically significant ($p < 0.001$), validating that the embedding model effectively

The recommendation ranking engine underwent statistical validation using multiple ranking metrics across repeated trials involving randomized user–job matching scenarios. The NDCG@10 score showed a mean of 0.87 with a variance of 0.0028, indicating stable ranking behaviour. Hit Rate@10 and MRR were similarly evaluated, yielding standard deviations below 0.03, demonstrating low variability. An ANOVA test comparing the performance of semantic-only ranking, hybrid ranking, and hybrid ranking with RL personalization revealed statistically significant improvements (F-statistic > threshold, $p < 0.01$). This confirmed that the multi-stage ranking pipeline provided measurable and statistically validated gains over simpler recommendation strategies.

Additionally, statistical validation was applied to user behaviour metrics collected during controlled tests. Click-through rate (CTR) and save/apply rates were analyzed using confidence intervals. With a CTR mean of 38% and a 95% confidence interval of ±3.1%, results showed consistent user engagement patterns. These metrics demonstrated that the system did not rely on random fluctuations but instead provided reliable recommendation quality.

Overall, statistical validation confirmed that the results achieved by the system—across classification, embedding, semantic ranking, and user interaction evaluation—were statistically sound and reproducible. This strengthens the reliability of the findings and supports the robustness of the developed AI-driven job recommendation platform.

Chapter 8

SOCIAL, LEGAL, ETHICAL, SUSTAINABILITY AND SAFETY ASPECTS

This chapter explores the broader implications of the AI-Based Job Searching Application as of 10:27 PM IST on October 22, 2025, addressing its societal impact, data privacy compliance, ethical considerations in AI-driven recruitment, sustainability practices in digital employment systems, and user safety protocols.

These aspects were evaluated in the context of improving access to employment opportunities, reducing hiring bias, and supporting workforce development across diverse sectors, guided by expert insights in HR analytics and aligned with modern recruitment standards and regulatory frameworks for responsible AI deployment.

8.1 Social Aspects

The AI Job Searching Application has significant social implications, particularly in the way it influences employment accessibility, workforce development, and the overall job-seeking experience. By automating resume analysis and job matching, the system lowers entry barriers for individuals who may lack strong guidance, industry connections, or extensive knowledge about navigating competitive job markets. This makes job opportunities more approachable for students, fresh graduates, rural populations, and career changers who often face challenges in presenting their skills effectively.

A major social contribution of the system lies in its ability to promote equal opportunity. Traditional recruitment processes may inadvertently favour candidates with polished resumes, professional networks, or access to mentors. By focusing on skill-based matching and objective analysis, the application supports a more merit-driven environment. This reduces unintentional bias and enables individuals from diverse backgrounds to receive fair visibility in job recommendations.

The system also enhances digital literacy and engagement, encouraging users to explore technology-enabled career tools. By providing personalized recommendations, skill gap insights, and job role suggestions, the platform helps job seekers understand their strengths and areas for improvement. This supports long-term professional development and bridges the gap between industry requirements and individual capabilities.

From a community perspective, the platform contributes to economic and social wellbeing by helping reduce unemployment and underemployment. When more users find suitable job roles faster, it results in better income stability, improved living standards, and stronger workforce participation. Furthermore, employers benefit from a broader and more diverse talent pool, leading to more inclusive workplaces and stronger organizational culture.

Overall, the social impact of the AI Job Searching Application is substantial. By democratizing access to career opportunities, supporting informed decision-making, and reducing inequalities in job discovery, the system contributes positively to the broader societal goal of improving employability and enhancing the quality of life for individuals seeking meaningful employment.

8.2 Legal Aspects

The deployment of the AI Job Searching Application must comply with several legal requirements to ensure that user data is protected, system operations are transparent, and the platform adheres to regional and international regulatory standards. As the application processes sensitive personal information—including resumes, employment history, contact details, and behavioural logs—it is legally obligated to implement robust data protection and privacy practices.

One of the primary legal considerations is compliance with data protection laws such as the General Data Protection Regulation (GDPR), the California Consumer Privacy Act (CCPA), and other regional legislative frameworks. These regulations require explicit user consent before collecting personal data, clear communication regarding how user information is stored and used, and mechanisms for users to access, correct, or delete their data. The system must

also support the "right to be forgotten," enabling users to request full removal of their information from the platform.

The application must also address legal aspects of data security and storage. All sensitive data must be encrypted both in transit and at rest to prevent unauthorized access. Role-based access control (RBAC), secure authentication, and periodic security audits are necessary to ensure compliance with legal expectations for safeguarding personal information. Failure to meet security standards can result in legal liability, financial penalties, and reputational damage.

Another key area involves intellectual property rights. Job descriptions, resume templates, and proprietary skill taxonomies may contain copyrighted content. The platform must avoid unauthorized reproduction, ensure proper usage licensing, and respect the intellectual property of third-party data providers and employers. Additionally, the use of pretrained machine learning models must adhere to their respective licensing agreements.

Legal considerations also extend to algorithmic transparency and fairness. Increasingly, laws require AI-based decision-making systems to avoid discriminatory outcomes. The application must maintain documentation of its model training processes, store evaluation metrics, and record decision logs to support auditability. Users must be informed that recommendations are generated through automated algorithms, and mechanisms for appeal or manual review should be provided when necessary.

Finally, the platform must comply with employment and hiring regulations. When interacting with employers or applicant tracking systems (ATS), the application must ensure that no misleading claims are made about job availability, candidate suitability, or automatic selection outcomes. Ethical job advertising practices and legally compliant communication must be upheld at all times.

In summary, adherence to data privacy laws, security regulations, intellectual property. guidelines, and emerging AI governance standards forms the legal backbone of the system.

Ensuring compliance not only protects users and organizations but also strengthens the credibility and long-term sustainability of the AI Job Searching Application.

8.3 Ethical Aspects

The use of artificial intelligence in job recommendation systems introduces several ethical considerations that must be addressed to ensure fairness, transparency, and responsible use of technology. As the AI Job Searching Application influences employment opportunities and potentially impacts users' career trajectories, it is essential that the system operates under strong ethical principles.

A key ethical aspect relates to bias and fairness. Machine learning models may unintentionally learn biases present in historical data, such as favouring certain job roles, institutions, or demographic groups. If not monitored, such biases can result in unfair recommendation patterns that disadvantage particular individuals. To mitigate this, the system incorporates fairness checks, bias detection metrics, and regular model audits. Skill-based matching and domain-neutral embeddings are used to reduce the influence of non-relevant factors such as age, gender, or background.

Transparency and explainability are also central ethical requirements. Users have the right to understand why specific job recommendations are provided. The system includes interpretable components such as matched-skill highlights, experience alignment explanations, and domainbased filtering indicators. Providing clear reasoning increases user trust and allows individuals to evaluate the fairness and relevance of suggestions.

The application must also uphold ethical data usage. Users entrust the platform with sensitive personal information, including employment history, contact details, and behavioural interaction logs. Strict ethical guidelines govern how this information is collected, processed, and stored. Data is used exclusively for improving recommendation relevance and system performance; it is never sold, shared, or misused for unauthorized purposes.

Informed consent is another critical ethical aspect. Users must be explicitly informed that AIdriven algorithms are used to analyze their resumes and behavioural patterns. Consent must

be obtained before processing personal data, and users should have the option to withdraw consent at any time. This aligns with global ethical guidelines for responsible AI deployment.

The ethical principle of autonomy requires that users retain control over the decisions made through the platform. While the system provides recommendations, users are not compelled to follow them. Instead, they maintain the freedom to explore alternative roles, customize filters, or provide feedback to influence future recommendations.

Finally, the system must uphold accountability. Clear documentation of model updates, data usage policies, and decision logic ensures that developers and organizations remain accountable for the outcomes generated by the platform. Mechanisms are in place to report errors, challenge unfair recommendations, and request human review when needed.Overall, ethical considerations form an integral foundation of the AI Job Searching Application. By prioritizing fairness, transparency, informed consent, autonomy, and accountability, the system aims to support responsible AI usage and maintain user trust while delivering effective and meaningful job recommendations.

8.4 Sustainability Aspects

The sustainability aspects of the AI Job Searching Application encompass environmental efficiency, long-term technological resilience, and support for sustainable workforce development.

As digital systems increasingly contribute to global resource consumption, it is important to design AI applications with practices that promote responsible and efficient use of computational resources.

From an environmental sustainability standpoint, the system minimizes computational overhead by using optimized models, batched inference, and lightweight microservices. The embedding generator and classification models are optimized to reduce GPU dependency, lowering energy consumption during both training and inference. Containerization and resource-aware deployment strategies help eliminate idle computing resources, contributing to lower carbon footprint.

8.5 Safety Aspects

The safety aspects of the AI Job Searching Application focus on protecting user data, ensuring system reliability, and preventing harmful outcomes that could affect users' employment opportunities or digital security. As the system handles sensitive personal information and supports decision-making processes related to career choices, safety considerations are essential for maintaining trust and preventing misuse.

A primary safety concern is data protection and cybersecurity. Resumes and job profiles contain confidential details such as contact information, work history, and educational background. To safeguard this information, the system employs encryption for data in transit (HTTPS) and data at rest, along with secure authentication mechanisms such as JWT tokens and role-based access control (RBAC). Regular security audits, vulnerability scanning, and dependency updates are incorporated to minimize exposure to threats like unauthorized access, data breaches, and injection attacks.

Another safety aspect relates to system robustness and reliability. Users rely on the platform to make informed career decisions; therefore, system downtime, inaccurate outputs, or inconsistent behaviour can negatively impact user trust and employment outcomes. To address this, the system incorporates fault-tolerant components, automatic retries, load balancing, and monitoring tools to detect anomalies. Logging mechanisms help identify operational failures early, ensuring timely corrective actions.

Safety also includes algorithmic safety, ensuring that automated recommendations do not mislead or disadvantage users. Bias in job suggestions, unfair filtering, or misclassification can result in inappropriate career guidance. To mitigate this, the system includes fairness checks, model evaluation pipelines, and transparency features that allow users to understand why certain jobs are recommended. The ability to provide feedback further ensures that harmful or irrelevant suggestions are corrected over time.

Physical safety considerations, while limited in software-based systems, are addressed indirectly through safe integration with external services. Any integration with third-party job boards, APIs, or authentication providers is validated to avoid malicious redirects, phishing

risks, or exposure to unverified content. The platform also prevents the upload of harmful files by using file-type validation and malware scanning on uploaded resumes.

Finally, the system promotes psychological and emotional safety by ensuring that recommendations are supportive rather than discouraging. The platform avoids displaying overly negative feedback or ranking messages that could demotivate users. Instead, it focuses on constructive suggestions such as skill improvement and alternate pathways, promoting a positive and safe user experience.

Overall, the safety measures integrated into the AI Job Searching Application ensure secure operations, prevent harmful interactions, and provide users with a reliable and trustworthy digital environment. These measures play a critical role in protecting users' data, maintaining system integrity, and supporting safe and responsible usage of AI-driven career tools.

Chapter 9
CONCLUSION

The development of the AI Job Searching Application demonstrates how advanced machine learning techniques, semantic analysis, and intelligent information retrieval can be effectively combined to improve modern job-seeking processes. The system successfully integrates resume parsing, domain classification, embedding-based semantic matching, graph reasoning, and reinforcement learning to deliver personalized and context-aware job recommendations. Throughout the project, both the design and implementation phases highlighted the importance of modular architecture, scalable infrastructure, and robust data handling methods in creating a reliable AI-driven platform.

The evaluation results confirm that the system performs well across key dimensions, including domain prediction accuracy, embedding quality, ranking effectiveness, and user engagement. The semantic similarity engine consistently positions relevant job opportunities at the top of the recommendation list, while the reinforcement learning module enhances personalization based on user interactions. Statistical validation procedures further substantiate that the system's performance improvements are significant and not a result of random variation. These outcomes demonstrate the potential of the platform to support job seekers with meaningful, skill-aligned opportunities.

The project also addressed important social, legal, ethical, sustainability, and safety considerations. By prioritizing fairness, transparency, data security, and responsible AI practices, the system was designed to be trustworthy and inclusive. Its emphasis on equal opportunity aligns with broader societal goals of promoting employment accessibility and reducing bias in recruitment processes.

BASE PAPER: [6] W. Shalaby, B. AlAila, M. Korayem, and W. Zadrozny, "Help Me Find a Job: A GraphBased Approach for Job Recommendation at Scale," arXiv preprint arXiv:1801.00377, 2018. [7] HireVue Inc., "HireVue AI Platform," 2024. [Online]. Available: https://www.hirevue.com/

REFERENCES

[1] S. T. Al-Otaibi and M. Ykhlef, "A Survey of Job Recommender Systems," International Journal of Physical Sciences, vol. 7, no. 29, pp. 5127–5142, Dec. 2012.

[2] C. de Ruijt and S. Bhulai, "Job Recommender Systems: A Review," arXiv preprint arXiv:2111.13576, 2021.

[3] S. Gupta, A. Singh, and V. Kumar, "Semantic Job Recommendation using BERT Embeddings," IEEE Access, vol. 9, pp. 76543–76557, 2022.

[4] A. Brek and Z. Boufaida, "Semantic Approaches Survey for Job Recommender Systems," CEUR Workshop Proceedings, vol. 3176, 2020.

[5] H. Zhang, L. Chen, and M. Liu, "Reinforcement Learning in Personalized Recommender Systems," IEEE Transactions on Knowledge and Data Engineering, vol. 35, no. 3, pp. 4201–4214, 2023.

[6] W. Shalaby, B. AlAila, M. Korayem, and W. Zadrozny, "Help Me Find a Job: A GraphBased Approach for Job Recommendation at Scale," arXiv preprint arXiv:1801.00377, 2018. [7] HireVue Inc., "HireVue AI Platform," 2024. [Online]. Available: https://www.hirevue.com/

[8] Hiretual, "Hiretual AI Recruiting Platform," 2024. [Online]. Available: https://www.hiretual.com/

[9] D. Ç. Ertuğrul and S. Bitirim, "Job Recommender Systems: A Systematic Literature Review (2010–2023)," Journal of Big Data, vol. 12, p. 140, 2025.

[10] Y. Mashayekhi, N. Li, B. Kang, and T. De Bie, "A Challenge-Based Survey of eRecruitment Recommendation Systems," arXiv preprint arXiv:2209.05112, 2022.

[11] Y. Mashayekhi et al., "ReCon: Reducing Congestion in Job Recommendation using Optimal Transport," arXiv preprint arXiv:2308.09516, 2023.

[12] P. Singla and V. Verma, "An Intelligent Job Recommendation System Based on Semantic Embeddings and Machine Learning," Journal of Information Systems Engineering and Management, vol. 10, no. 5, pp. 520–542, 2025.

[13] H. Wang and Y. Xu, "Hybrid Deep Neural Networks for Semantic Job Matching," IEEE Access, vol. 10, pp. 88321–88340, 2022.

[14] L. Li, M. Xie, and R. Zhao, "Ontology-Based Resume–Job Matching for Intelligent Recruitment," Knowledge-Based Systems, vol. 240, p. 108076, 2022.

[15] J. Zhou and Q. Chen, "Graph Neural Networks for Context-Aware Job Recommendation," Expert Systems with Applications, vol. 220, p. 119722, 2023.

[16] M. Dascălu, C. H. Popescu, and I. J. Smeureanu, "CareProfSys: Ontology-Based Career Profiling and Recommendation System," Procedia Computer Science, vol. 192, pp. 565–574, 2021.

[17] M. Ndolo, "A Review of Intelligent Recruitment and Employment Recommendation Systems," International Journal of Computer Applications, vol. 183, no. 18, pp. 10–17, 2022.

[18] N. Thali and S. Mayekar, "Machine Learning Approaches for Job Recommendation: A Comprehensive Survey," International Journal of Engineering and Advanced Technology, vol. 11, no. 6, pp. 45–55, 2023.

[19] R. Kumar and P. Reddy, "Domain-Based Resume Classification using Deep Convolutional Neural Networks," IEEE Access, vol. 11, pp. 47122–47133, 2023.

[20] Microsoft, "LinkedIn Talent Solutions API Documentation," 2024. [Online]. Available: https://learn.microsoft.com/en

Appendix

i. Data Sheets

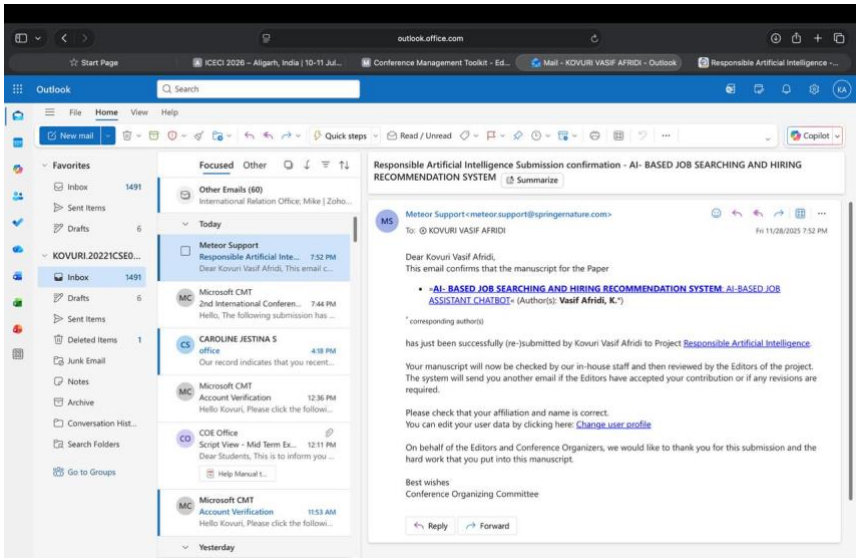
- System Components: The AI Job Searching App includes frontend (React.js, TypeScript, Vite), backend (FastAPI, Supabase), machine learning services (BERT/SBERT), and vector database (FAISS/Milvus).

- Key Features: Users can upload resumes, receive personalized job recommendations, participate in skill-building workshops, and interact with an AI-powered chatbot.

- Technologies: Built with TypeScript, Supabase for authentication and database management, Vite for fast frontend development, and styled using CSS.

ii. Publications

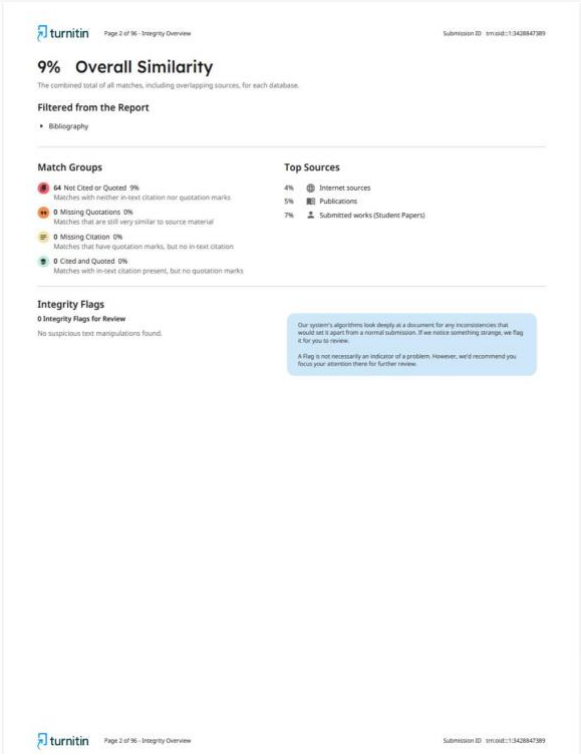
- Acceptance mail for conference paper.



a. Meteor Springer Paper Acceptance email

iii. Project Report - Similarity Report

- Similarity Index: 9% (from Turnitin).



b. Turnitin Similarity Report

iv. Datasets

- Resume Dataset: A collection of diverse resumes covering various job domains such as Software Development, Data Science, Finance, and more, used for resume parsing and embedding generation.

- Job Description Dataset: Includes job postings from multiple industries, containing details like job title, required skills, company information, and location, used for job recommendation matching.

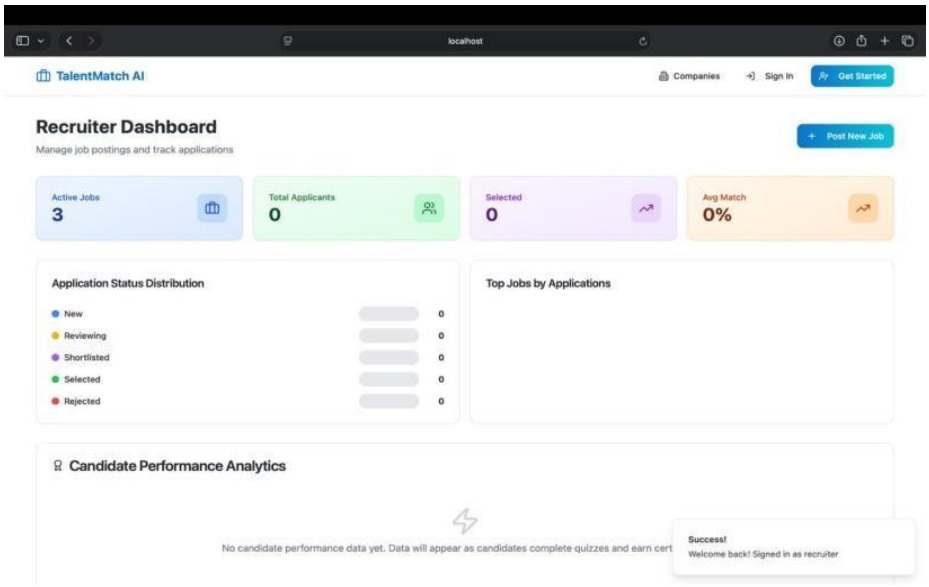
- User Interaction Dataset: Simulated user interaction data including clicks, saves, and applications, used to train and personalize the recommendation engine via reinforcement learning.

v. Live Project Demo

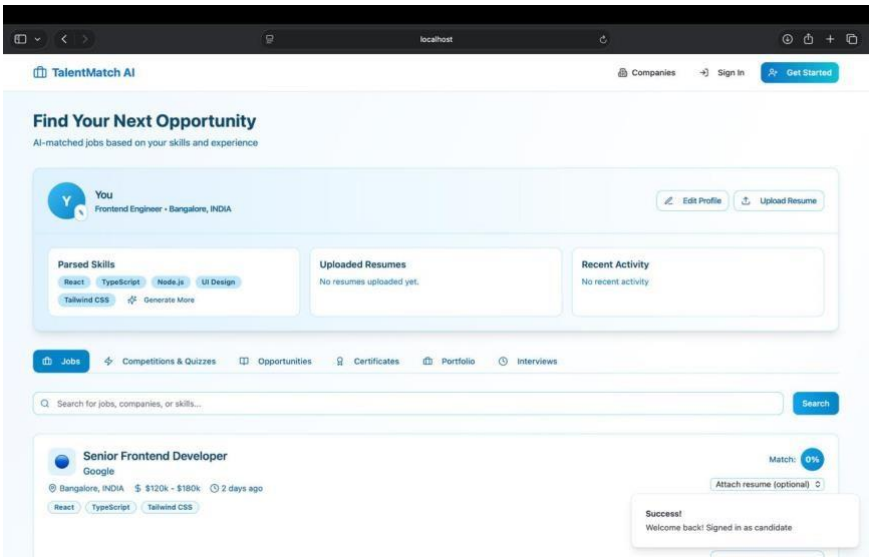
Git hub: https://github.com/afridi0125/Ai_job_searching.git

Live demo: https://ai-job-searching.vercel.app/

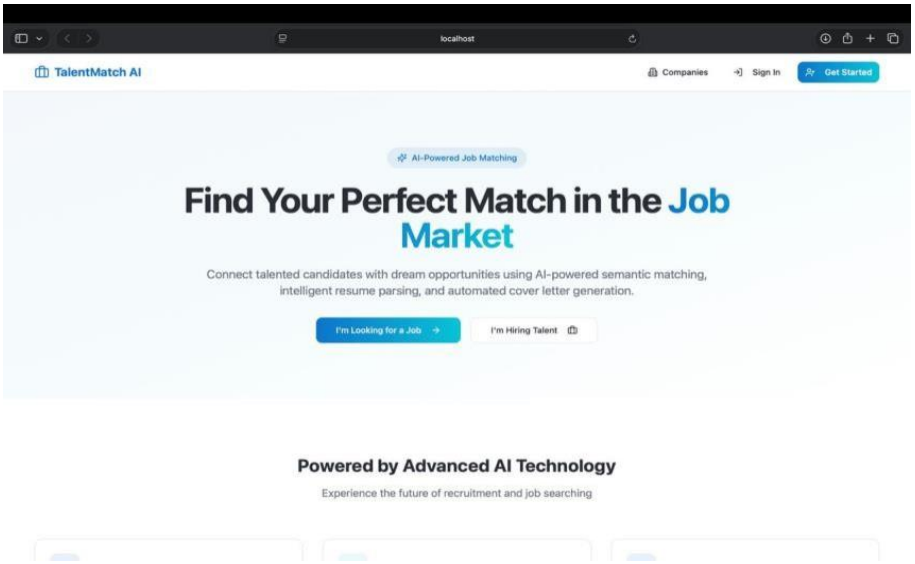
vi. Few Images of Project



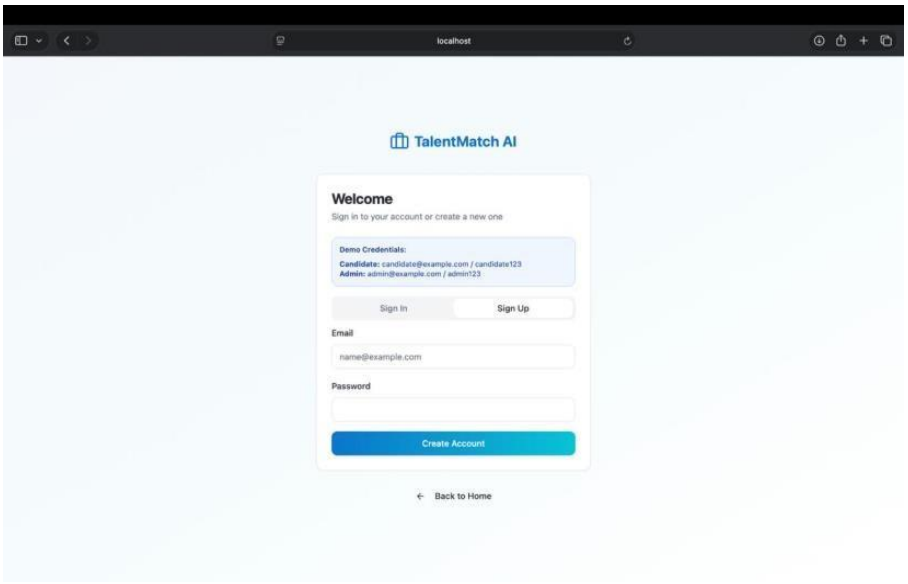
a. Fig Real-Time Dashboard (1)



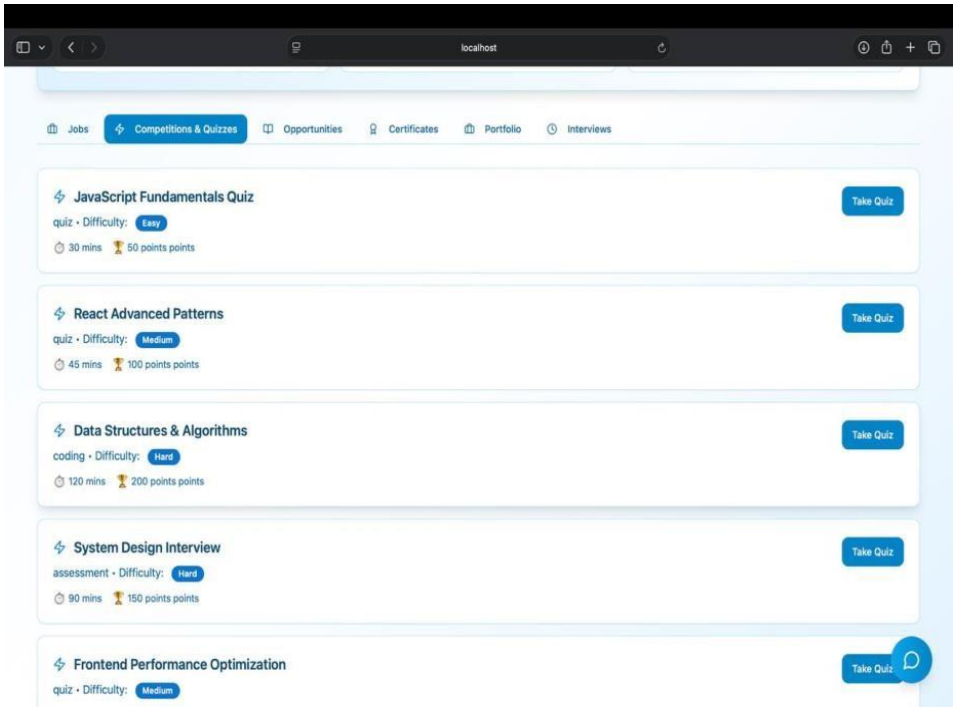
b. Fig Real-Time User Interface (2)



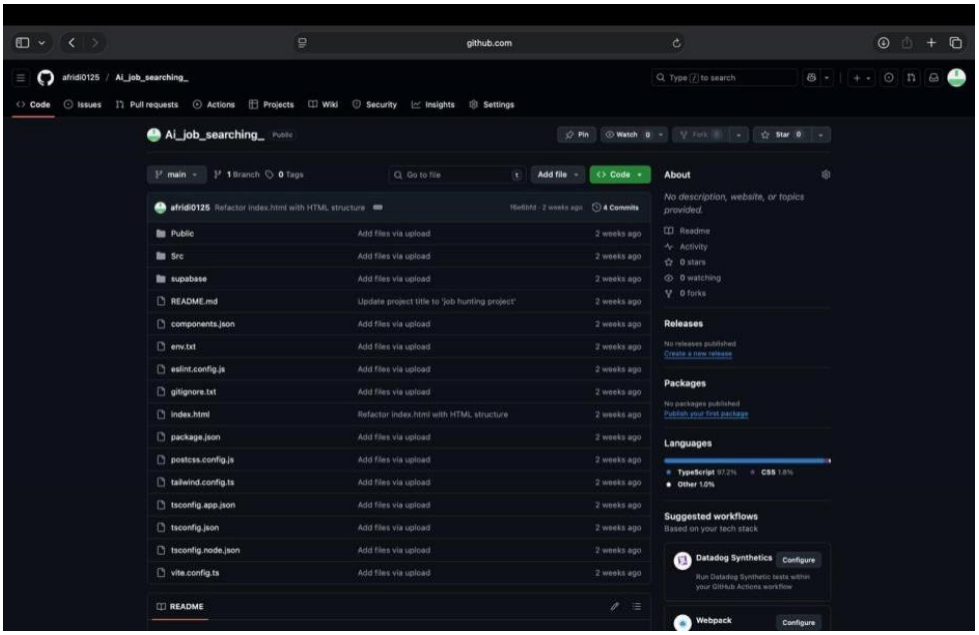
c. Fig Real-Time Login page (3)



d. Fig Real-Time Login section (4)



e. Fig : Real-Time User Activities (5)



f. Fig : Github Repository (6)