Thesis submitted to the

UNIVERSITY OF MOHAMED BOUDIAF – MSILA

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
DEPARTMENT OF COMPUTER SCIENCE

In partial fulfillment of the requirements for the degree of

Master in **Computer science**

By

Sabir DEGHFEL

El aid Amin CHEKHABA

Title of the thesis

AI-Powered Web Platform for Automated Curriculum Vitae Evaluation and Job Matching

Under the supervision of

Madiha HALASSA

Composition of the jury

CHALABI Baya	University of Msila	President
HALASSA Madiha	University of Msila	Reporter
ATTIR Azedine	University of Msila	Examiner

June, 2026

Abstract

Traditional recruitment processes rely on manual CV screening, which is time-consuming, inconsistent, and prone to overlooking qualified candidates. This thesis presents the design and implementation of an AI-powered web platform that automates curriculum vitae evaluation and job matching. The system extracts candidate skills from uploaded resumes using named entity recognition, then computes a semantic similarity score between each CV and a target job description using transformer-based sentence embeddings and cosine similarity. Recruiters are presented with a ranked list of applicants, enabling faster and more objective screening decisions. The platform is built around three independent components: a Node.js/Express back-end server, a React front-end application, and a Python/Flask AI inference engine. The implemented system was successfully deployed as a live web service and validated through interface testing across all three user roles: candidate, recruiter, and administrator.

Keywords: Recruitment; Artificial Intelligence; Natural Language Processing; CV Evaluation; Semantic Similarity; Job Matching

---

ملخص

تعتمد عمليات التوظيف التقليدية على فرز السير الذاتية يدوياً، مما يستلزم وقتاً طويلاً وينطوي على قدر كبير من عدم الاتساق واحتمال إغفال مرشحين مؤهلين. تقدم هذه المذكرة تصميم وتطوير منصة ويب مدعومة بالذكاء الاصطناعي تعمل على أتمتة تقييم السير الذاتية ومطابقتها مع عروض العمل. يستخلص النظام مهارات المترشحين من السير المرفوعة باستخدام تقنية التعرف على الكيانات المسمّاة، ثم يحسب درجة تشابه دلالية بين كل سيرة ووصف الوظيفة باستخدام التمثيلات المستندة إلى المحولات وتشابه جيب التمام. تتكون المنصة من ثلاثة مكونات مستقلة: خادم Node.js/Express، وتطبيق React، ومحرك ذكاء اصطناعي مبني على Python/Flask. جرى نشر النظام بنجاح كخدمة ويب حية والتحقق منه عبر اختبار الواجهات لجميع أدوار المستخدمين الثلاثة.

الكلمات المفتاحية: التوظيف، الذكاء الاصطناعي، معالجة اللغة الطبيعية، تقييم السيرة الذاتية، التشابه الدلالي، مطابقة الوظائف

Dedication

To my beloved family,

Words fall short of expressing how much you mean to me. You have been my foundation, my shelter, and my greatest source of strength. Every sacrifice you made, every prayer you offered, and every word of encouragement you gave carried me further than you will ever know. This achievement is, before anything else, yours.

To my friends,

To those who shared laughter and hardship with me, who believed in me even when I doubted myself, and who made the long days of this journey lighter and brighter, this work carries a piece of each of you.

This work is dedicated to all of you, with love and gratitude.

- DEGHFEL Sabir –

To My Family,

To the silent pillars of my life, whose unconditional love and unwavering belief have been the compass guiding me through every storm. This milestone is a reflection of your quiet sacrifices, your endless patience, and the peaceful haven you always provided when the world became overwhelming. I offer you this work as a small token of a debt of gratitude I can never truly repay.

To My Companions,

To the peers and close friends who turned the rigorous demands of this academic chapters into an era of shared triumphs. Thank you for the late-night intellectual debates, the collaborative spirit, and the genuine solidarity that transformed complex challenges into memorable milestones. You made the heavy days feel light and the victories worth celebrating.

- CHEKHABA El Aid Amin -

Acknowledgement

In the name of Allah, the Most Gracious, the Most Merciful. All praise is due to the Almighty for His divine grace, blessings, and for granting me the resilience and clarity of mind to bring this endeavor to fruition.

I owe a profound debt of gratitude to my esteemed supervisor, Dr. HALLASSA Madiha, for her academic rigor, insightful critiques, and empowering mentorship. Her high standards and constructive feedback profoundly shaped not only the direction of this project but also my growth as a researcher. I am deeply grateful for her patience and for constantly challenging me to push the boundaries of my capabilities.

My sincere appreciation goes to the faculty and staff of the Department of Computer Science. Thank you for cultivating an environment of intellectual curiosity and for equipping us with the foundational tools necessary to navigate the rapidly evolving technological landscape.

Finally, I would like to extend my heartfelt thanks to everyone who contributed to this journey, whether through a fresh perspective, a word of timely encouragement, or a moment of welcome respite during this intense chapter of my life. Your presence has been an invaluable asset.

- CHEKHABA El Aid Amin –

First and foremost, all praise and gratitude are due to Allah, the Almighty, for granting me the strength, patience, and wisdom to reach this milestone.

I would like to express my deepest and most sincere gratitude to my supervisor, Dr. HALLASSA Madiha, for her invaluable guidance, continuous support, and thoughtful advice throughout this journey. Her dedication and expertise have been a true source of inspiration and motivation.

I extend my heartfelt thanks to all the professors and staff of the Department of Computer Science for their knowledge, encouragement, and the academic foundation they have provided throughout my studies.

To my friends who walked beside me through every challenge and every victory, your presence, your help, and the memories we shared have made this experience truly unforgettable. I am deeply grateful for each one of you.

Finally, I wish to acknowledge everyone who contributed, directly or indirectly, to the completion of this work. Every kind word, every moment of support, and every shared memory has left a mark on this journey.

- DEGHFEL Sabir -

Table of Contents

List of Figures

List of Acronyms

Introduction

In this age of unprecedented technological proliferation, modern societies are experiencing rapid transformations driven by advances in information technology and computing systems. These developments have significantly improved the efficiency and organization of institutions and companies. As a result, digital information systems have become essential tools for managing data, optimizing operations, and supporting decision-making processes in today's fast-paced professional environment.

Before the widespread adoption of digital technologies, many administrative tasks were performed manually using paper-based documentation. In practice, organizations traditionally relied on manual screening of CVs to evaluate job applicants. However, this approach often requires considerable time and effort, especially when dealing with a large number of applications. For this reason, traditional recruitment methods may lead to inefficiencies and increase the risk of overlooking qualified candidates.

With recent advances in AI, ML, and NLP, it has become possible to develop intelligent systems capable of analyzing textual information automatically. These technologies enable automated extraction of relevant information from resumes and facilitate the evaluation of candidates based on job requirements.

In this context, the objective of our project is to design and implement an intelligent web platform for automated CV evaluation and job matching. This system aims to assist recruiters by providing a tool capable of analyzing candidate resumes and identifying the most suitable applicants for a given job position.

This thesis is organized into three main chapters. The first chapter provides an overview of recruitment systems, discusses their main challenges, and highlights the limitations of traditional approaches. The second chapter focuses on the system design using UML modeling techniques. Finally, the third chapter presents the implementation of the proposed platform and describes the main technologies used in its development.

CHAPTER ONE

BACKGROUND AND PROBLEM STATEMENT

INTRODUCTION

The aim of this chapter is to establish the theoretical and practical foundations necessary for understanding the problem addressed in this work. We begin by presenting the recruitment process and the role of Human Resource Management, then identify the main actors involved. Following this, we examine the advantages that digital technologies bring to recruitment and expose the well-documented limitations of traditional approaches. The chapter then provides a detailed study of the key enabling technologies, namely Artificial Intelligence, Machine Learning, Natural Language Processing, and Deep Learning, with a particular focus on the methods that underpin the proposed system: TF-IDF-based text representation, skills extraction, and semantic similarity through sentence embeddings. We close by defining the project objectives and describing the proposed solution.

1. RECRUITMENT PROCESS

To understand what problem we are trying to solve, we first need a clear picture of what recruitment actually involves. Several well-known authors have defined it from different angles:

"Recruitment is the process of finding and engaging the people the organization needs.[1]*"*

– Armstrong M. (2014) –

"Recruitment is the process of discovering potential candidates for actual or anticipated organizational vacancies. [2]*"*

– Dessler G. (2013) –

These two definitions capture something slightly different but complementary. Armstrong focuses on engagement, meaning the active effort to bring the right people in. Dessler emphasizes discovery, the idea that recruitment is often a search for people who do not yet know about the opportunity. Together, they point to recruitment as a structured activity: identifying what the organization needs, finding people who might meet that need, and selecting among them.

In practice, this unfolds through a fairly consistent sequence of steps. It begins with job analysis, where the organization defines exactly what skills, qualifications, and responsibilities are associated with the position. The vacancy is then communicated externally through job postings on relevant platforms. Applications come in and are screened, usually against some predefined criteria, and the most promising candidates are invited to interview. A final selection decision follows, and the process ends with an offer.

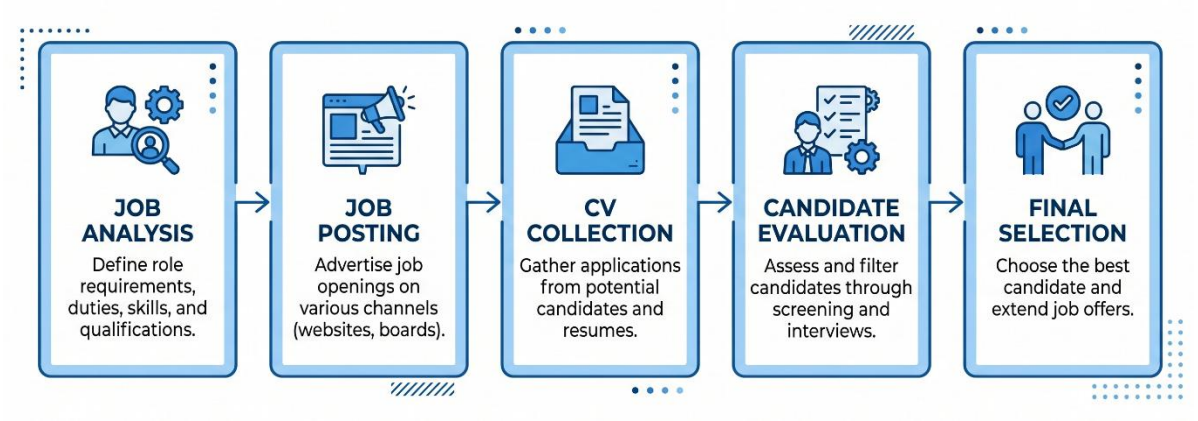

Figure 1.1: Recruitment Process Flow Diagram

Each stage of this process involves a significant amount of information exchange and decision-making. As the volume of applicants grows, a common scenario in today's competitive job market, managing this information manually becomes increasingly challenging. This observation forms the central motivation for introducing intelligent technologies into the recruitment workflow.

2. HUMAN RESOURCE MANAGEMENT

Human Resource Management (HRM) encompasses all organizational practices directed at managing the employment relationship in a way that enables the firm to achieve its strategic objectives while simultaneously developing the capabilities of its workforce. It involves planning human resource needs, organizing staffing activities, supervising
performance, and ensuring employee development over time.

Within HRM, the recruitment function occupies a particularly critical position. As Dessler (2013) [2] observes, it is through recruitment that organizations build the human capital necessary to sustain competitive advantage. The HR department acts as the custodian of this process:

It coordinates job postings, manages application pipelines, evaluates candidate profiles, and facilitates the final hiring decision. Once candidates are hired, the HR department also oversees their integration into the organization and monitors their ongoing performance.

Given the strategic importance of recruitment, any inefficiency in this process has direct consequences for organizational performance. This is why the modernization of recruitment practices particularly through the application of intelligent computational systems has become an active area of research and development in recent years.

3. ACTORS OF THE RECRUITMENT PROCESS

3.1. Organization (Recruiter)

The organization is the entity that initiates the recruitment process. It identifies workforce needs, defines the required skills and qualifications for a given position, and sets the conditions of employment. The organization is responsible for communicating job vacancies through appropriate channels and for making the final selection decision based on the information collected throughout the process.

3.2. Candidates (Job Seekers)

Candidates are individuals who apply for job positions within the organization. They submit their CVs and provide relevant information about their qualifications, skills, and experience. Candidates participate in the selection process in order to obtain employment.

3.3. HR Department

The HR department serves as the operational backbone of the recruitment process. It manages the flow of applications, coordinates screening activities, communicates with candidates, and ultimately presents shortlisted profiles to decision-makers within the organization. In the context of intelligent recruitment systems, the HR department transitions from a manual screening role to a supervisory and validation role, where algorithmic recommendations are reviewed and acted upon.

4. ADVANTAGES OF USING TECHNOLOGY IN RECRUITMENT

The integration of digital technologies into recruitment has produced measurable improvements across several dimensions. From a time-efficiency standpoint, automated systems can process and pre-screen thousands of applications in a fraction of the time required by human reviewers. This acceleration is particularly valuable in high-volume recruitment contexts, where delays in processing can result in the loss of top candidates to competing employers.

Beyond speed, technology contributes to greater consistency in candidate evaluation. Human reviewers are susceptible to cognitive biases including confirmation bias, affinity bias, and fatigue-related errors that can compromise the fairness and accuracy of screening decisions. Algorithmic systems, when designed carefully, apply uniform criteria across all applications, thereby reducing the influence of irrelevant factors.

Technology also enhances the manageability of large application pools. Centralized digital platforms allow HR teams to store, search, and retrieve candidate data efficiently, facilitating both immediate hiring decisions and future talent pipeline management. Furthermore, AI-powered matching tools can identify non-obvious connections between candidate profiles and job requirements, surfacing candidates who might otherwise be overlooked by keyword-based search methods.

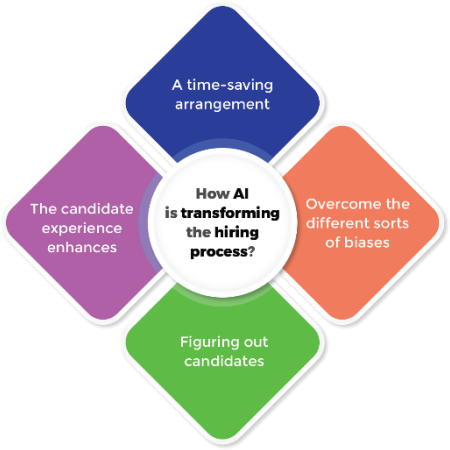

Figure 1.2: Benefits of AI in Recruitment

Collectively, these advantages make a compelling case for the adoption of technology-assisted recruitment, particularly in organizational contexts characterized by high applicant volumes and the need for rapid, accurate selection.

5. TRADITIONAL RECRUITMENT SYSTEMS LIMITATIONS

Despite the growing availability of digital tools, many organizations continue to rely on largely manual recruitment processes. This reliance introduces a range of well-documented limitations that motivated the development of the present system.

• Manual CV screening requires HR professionals to read and evaluate each application individually. When dealing with hundreds or thousands of applicants, this process is both time-consuming and cognitively taxing, often leading to superficial assessments.

• Subjectivity in evaluation is an inherent consequence of human judgment. Different recruiters may weigh the same qualifications differently, producing inconsistent outcomes across candidates and undermining the fairness of the process.

• Difficulty scaling to high application volumes is a structural limitation of manual methods. As the number of applications increases, the probability of qualified candidates being overlooked due to time pressure or fatigue rises proportionally.

• Poor matching between candidate profiles and job requirements is common when screening is performed using simple keyword search or visual inspection. Recruiters may miss semantically equivalent qualifications expressed in different terminology (e.g., "machine learning" vs. "statistical modeling").

• Absence of structured data makes it difficult to perform quantitative comparisons across candidates or to build reliable historical records for analytics and auditing purposes.

These limitations collectively justify the development of an intelligent, automated system capable of overcoming the constraints of manual recruitment.

6. ARTIFICIAL INTELLIGENCE AND RELATED TECHNOLOGIES

The proposed system draws on several interrelated fields within Artificial Intelligence (AI). This section provides a structured overview of these fields, with an emphasis on the concepts and techniques that are directly relevant to automated CV analysis and candidate-job matching.

6.1. Artificial Intelligence

Artificial Intelligence is a branch of computer science concerned with building systems that can perform tasks that, when performed by humans, require some form of intelligence, such as reasoning, learning, perception, and language understanding. The term "AI" is broad and encompasses numerous subfields, each addressing different aspects of intelligent behavior. For the purposes of this project, the most relevant subfields are Machine Learning (ML), Natural Language Processing(NLP), and Deep Learning (DL) [3].

6.2. Machine Learning

Machine Learning (ML) is a subfield of AI that focuses on the development of algorithms capable of learning patterns and making decisions from data without being explicitly programmed for each specific task. Rather than encoding domain knowledge through hand-crafted rules, ML systems derive knowledge from examples that is from labeled or unlabeled datasets. ML algorithms can be broadly categorized into three paradigms: supervised learning, where a model is trained on labeled input-output pairs; unsupervised learning, where the model discovers structure in unlabeled data; and reinforcement learning, where an agent learns by interacting with an environment and receiving feedback in the form of rewards or penalties. In the context of recruitment, ML techniques have been applied to a range of tasks, including Curriculum Vitae classification, skills extraction, candidate ranking, and attrition prediction. The ability of ML models to generalize from historical data makes them particularly well-suited to recruitment scenarios, where past hiring decisions can serve as training signal [3].

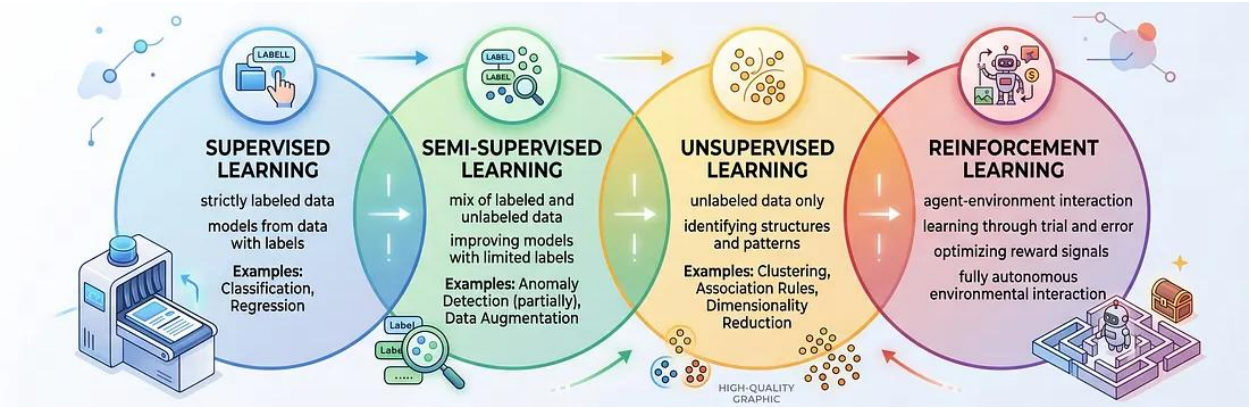

Figure 1.3: A spectrum of Machine Learning paradigms [34]

6.3. Natural Language Processing

Natural Language Processing (NLP) is the subfield of AI concerned with enabling computers to understand, interpret, and generate human language. Since Curriculum Vitae and job descriptions are inherently textual, NLP constitutes the core enabling technology for the present system.

NLP research has evolved considerably over the past two decades. Early approaches relied on hand-crafted linguistic rules and shallow statistical models, while contemporary NLP is dominated by neural architectures capable of learning rich linguistic representations directly from large text corpora. The following subsections describe two NLP techniques that are central to this project.

6.3.1. Text Representation: TF-IDF

Before any computational analysis can be performed on text, the text must first be converted into a numerical representation. One of the most established methods for this purpose is Term Frequency–Inverse Document Frequency (TF-IDF), a statistical measure that reflects the importance of a word within a specific document relative to a collection of documents (a corpus).

The TF-IDF weight of a term ***t*** in a document ***d*** is computed as:

$$\textbf{TF-IDF(t, d) = TF(t, d) × IDF(t)}$$

where **TF(t, d)** represents the frequency of term ***t*** within document ***d***, and **IDF(t) = log(N / df(t))** measures how rare the term is across the corpus of **N** documents, with **df(t)** denoting the number of documents containing ***t***. This formulation ensures that common words (e.g., "the", "and") receive **low** weights, while domain-specific terms that are informative for a given document receive **higher** weights [6].

In the recruitment context, TF-IDF can be applied to represent both CVs and job descriptions as weighted term vectors. A similarity measure, typically cosine similarity, can then be computed between these vectors to estimate the degree of lexical overlap between a candidate's profile and a job requirement. While TF-IDF is effective for detecting exact keyword matches, it does not capture semantic relationships between synonymous terms, a limitation that motivates the use of deep learning-based embeddings.

6.3.2. Information Extraction and Skills Detection

Information Extraction (IE) is the NLP task of identifying and structuring specific pieces of information from unstructured text. In the domain of CV analysis, IE is used to automatically identify and classify entities such as the applicant's name, contact details, educational qualifications, previous job titles, and technical skills.

Skills extraction, in particular, is a specialized form of Named Entity Recognition (NER) [7] in which the system identifies mentions of technical and soft skills within free text. This is a non-trivial problem because skill mentions vary widely in form and context. A candidate might describe proficiency in Python in a skills section, mention it implicitly in a project description, or refer to it by an alternative name (e.g., "scripting" or "automation"). Robust skills extractors must therefore go beyond simple pattern matching and leverage contextual understanding of the surrounding text.

The output of the skills extraction module in the present system is a structured list of competencies derived from each CV. These competencies are then compared against the skills required for a given job offer, enabling a more precise and interpretable matching process.

6.4. Deep Learning

Deep Learning (DL) is a specialized branch of Machine Learning that uses multi-layered artificial neural networks commonly referred to as deep neural networks, to learn hierarchical representations of data. The term "deep" refers to the number of layers in the network: each layer transforms its input into a progressively more abstract representation, allowing the model to capture complex patterns that shallow methods cannot [4] [5].

Deep learning has produced state-of-the-art results across a wide variety of tasks, including image recognition, speech processing, and natural language understanding. Its success in NLP tasks, in particular, has been transformative: deep models can now perform tasks such as question answering, machine translation, and semantic similarity with near-human accuracy.

6.4.1. Artificial Neural Networks

An Artificial Neural Network (ANN) is a computational model inspired by the structure and function of biological neural networks in the brain. It consists of interconnected processing units, called neurons or nodes, organized into layers: an input layer, one or more hidden layers, and an output layer [4].

Each connection between neurons is associated with a weight that is adjusted during training through a process called backpropagation. The network learns by minimizing a loss function.

A measure of the discrepancy between its predictions and the ground truth labels, using gradient-based optimization algorithms such as Stochastic Gradient Descent (SGD) or Adam [15].

The depth of a neural network that is the number of hidden layers, directly influences its capacity to model complex, non-linear relationships in the data. Networks with many layers (deep networks) are therefore better suited to tasks involving high-dimensional input, such as raw text or image data.

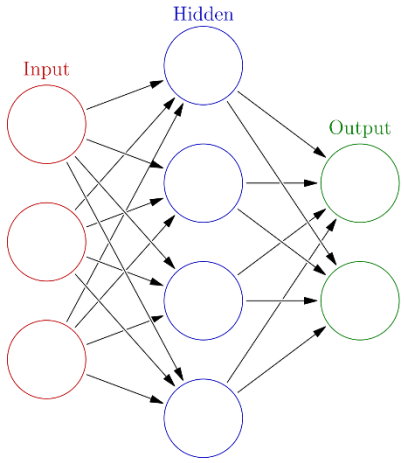

Figure 1.4: Simple Artificial Neural Network Architecture [35]

6.4.2. Word Embeddings and Sentence Embeddings

A key innovation that enabled deep learning to succeed in NLP tasks was the development of dense, low-dimensional vector representations of words, known as word embeddings. Unlike sparse representations such as one-hot encoding, word embeddings encode semantic relationships between words: words with similar meanings are mapped to nearby points in the vector space [10]. For example, the vectors for "engineer" and "developer" are closer to each other than either is to the vector for "accountant".

Building on this idea, sentence embeddings extend the concept to entire sequences of text. A sentence encoder maps a sentence (or a short document such as a Curriculum Vitae section) to a fixed-length dense vector that captures its overall meaning, regardless of its exact wording. The most powerful sentence encoders are based on the Transformer architecture, specifically on bidirectional pre-trained language models such as BERT, which process the entire input sequence simultaneously [8] [11] and capture long-range dependencies between words.

Sentence-BERT and its variants including the all-mpnet-base-v2 model family [9], fine-tune pre-trained transformer models on semantic similarity tasks to produce 768-dimensional sentence vectors that are highly effective for downstream applications such as semantic search, paraphrase detection, and document similarity.

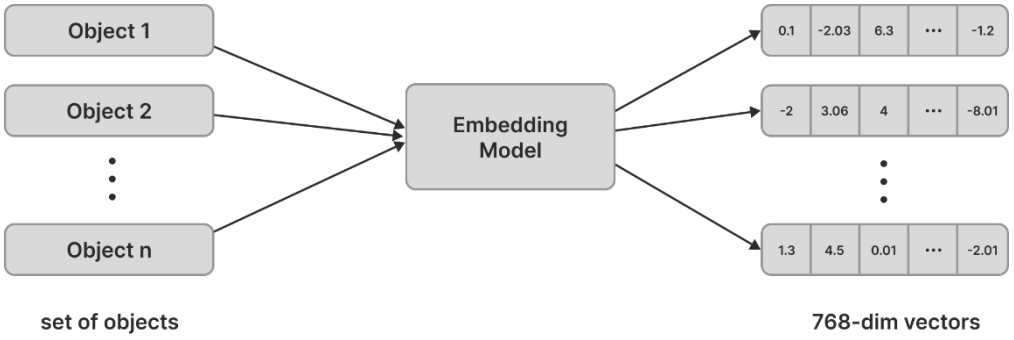

Figure 1.5: Simple Vectorization Phase Plan

In the present project, sentence embeddings are used to compare CV content with job descriptions at a deep semantic level, going beyond the surface-level lexical matching offered by TF-IDF.

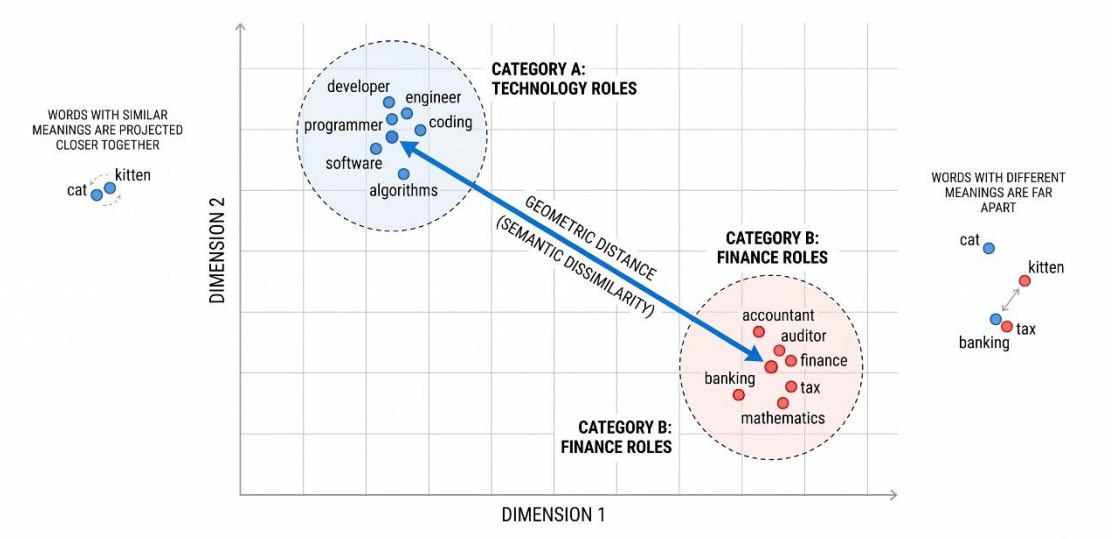

Figure 1.6: Visualizing Word Embeddings in 2D vector space

6.4.3. Semantic Similarity as a Matching Problem

In the context of this project, the task of determining whether a candidate is suitable for a given job can be recast as a semantic similarity problem: given a CV and a job description, how semantically similar are they? This framing is appropriate because candidate suitability is not simply a matter of keyword overlap, but rather of conceptual alignment between the candidate's profile and the job requirements.

Semantic similarity between two textual units can be quantified using the cosine similarity between their corresponding embedding vectors:

$$\text{cosine\_sim}(A, B) = (A \cdot B) / (||A|| \times ||B||)$$

where A and B are the embedding vectors of the CV and the job description, respectively. A score close to 1 indicates high semantic similarity (strong candidate-job alignment), while a score close to 0 indicates low similarity. This score, combined with the output of the skills extraction module, forms the basis of the candidate ranking system in the proposed platform.

This approach is closely related to classification tasks in machine learning, where the goal is to assign an input to one of several predefined categories. In recruitment, one can frame shortlisting as a binary classification task: for each (candidate, job) pair, predict whether the candidate should be shortlisted (positive class) or not (negative class). The matching score derived from semantic similarity can serve as a continuous-valued feature for this classifier, allowing the system to produce ranked lists of candidates ordered by predicted suitability.

6.4.4. Classification

Classification is one of the most fundamental tasks in supervised machine learning. It consists of assigning an input instance to one of a set of predefined discrete categories based on a learned mapping between input features and output labels. Classification problems are broadly divided into binary classification, which involves distinguishing between two mutually exclusive classes, and multi-class classification, which extends this to three or more categories. In deep learning, classification is typically performed by a neural network whose final layer applies a softmax activation function, producing a probability distribution over the possible classes [5]. In the context of the proposed system, the task of determining whether a candidate is suitable for a given job position is framed as a binary classification problem, where the matching score derived from semantic similarity serves as the primary feature driving the decision.

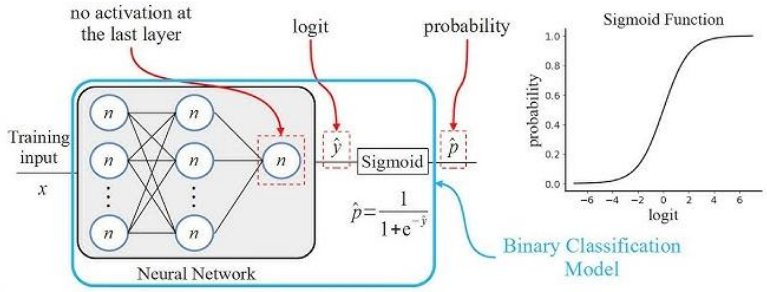

Figure 1.7: An architecture of ANN for binary classification

6.4.5. Text Classification

Text classification is a specialized application of classification in which the input instances are textual documents and the objective is to assign each document to one or more predefined categories. Early approaches relied on traditional machine learning algorithms combined with sparse representations such as TF-IDF. The advent of deep learning transformed the field, with transformer-based models such as BERT establishing state-of-the-art results by capturing rich bidirectional contextual representations of text [8]. In the proposed system, text classification drives the automated evaluation pipeline: candidate CVs are analyzed with respect to job descriptions, and a continuous matching score is produced to support shortlisting decisions, bridging semantic similarity with the classification framework.

7. RELATED WORKS

Before presenting the objectives of the proposed system, it is worthwhile to examine existing platforms that address the problem of connecting job seekers with employers, particularly within the Algerian market, in order to identify their limitations and the gaps that motivate the present work.

Emploitic is the leading recruitment platform in Algeria [30], dedicated to enabling companies to find the talent necessary for their growth and helping individuals build their professional future. The platform offers access to thousands of job listings categorized by function, sector, and region, and provides employer branding tools alongside a large and continuously growing candidate database. It is the most widely adopted employment portal in the country and constitutes the primary reference point for online recruitment in Algeria.

Despite its wide adoption, Emploitic operates as a traditional job board. The evaluation of candidates relies entirely on manual browsing by the recruiter, who must read through each application individually to assess its relevance. This process is time-consuming, inconsistent, and increasingly impractical as the volume of applications grows. There is no automated analysis of candidate profiles and no intelligent ranking of applicants based on their actual suitability for a given position.

Other platforms present in the Algerian market, such as EmploiPartner [31] and Option Carrière [32], follow a similar model and are subject to the same limitations. On the international scale, platforms such as LinkedIn [28] and Indeed [29] have introduced recommendation features, but these remain insufficient for the specific needs and context of the Algerian job market.

The proposed system addresses these limitations by introducing an intelligent layer into the recruitment process that automates candidate evaluation and ranking, reducing the manual effort required from recruiters and improving the overall quality and fairness of the selection process.

Figure 1.8: Related Works Companies Logo

8. OBJECTIVES AND GOALS

The primary objective of this project is to design and implement an intelligent web-based platform that automates the recruitment pipeline, with particular emphasis on CV analysis and candidate-job matching. The system is intended to provide HR professionals with a reliable, scalable, and interpretable tool that reduces manual workload while improving the quality of hiring decisions.

More specifically, the goals of the project are as follows:

- Automate the extraction of structured information from unstructured CV documents, including skills, educational qualifications, and professional experience, using NLP and IE techniques.
- Implement TF-IDF-based and semantic embedding-based approaches to quantify the similarity between candidate profiles and job descriptions.
- Develop a candidate ranking module that generates a matching score for each (candidate, job) pair and presents the results in a clear, actionable format for recruiters.
- Handle large volumes of applications efficiently through a scalable backend architecture.
- Provide a user-friendly interface that allows recruiters to publish job offers and candidates to upload their CVs in standard formats (PDF, DOCX).

9. PROPOSED SOLUTIONS

To address the limitations identified above and achieve the stated objectives, this project proposes the development of a full-stack intelligent recruitment platform. The system integrates three main processing layers: a web application front-end, a RESTful back-end service, and an AI inference engine.

The AI engine, implemented in Python, is responsible for performing all NLP and deep learning computations. It handles skills extraction using entity recognition techniques, computes TF-IDF representations of CV and job description texts, and generates dense sentence embeddings using a pre-trained transformer model. The resulting vectors are compared using cosine similarity, and the scores are aggregated into a final matching index for each candidate.

The back-end, built with Node.js and Express, manages user authentication, job offer and CV storage (via MongoDB), and API communication with the Python AI engine (via Flask). The front-end provides separate interfaces for recruiters and candidates, allowing the former to post jobs and review ranked candidate lists, and the latter to register and submit their CVs.

This architecture was designed with scalability and modularity in mind. The AI engine is decoupled from the web server, allowing each component to be independently updated or replaced as better models become available. The use of a document-oriented database (MongoDB) ensures flexibility in handling the heterogeneous structure of real-world CV data.

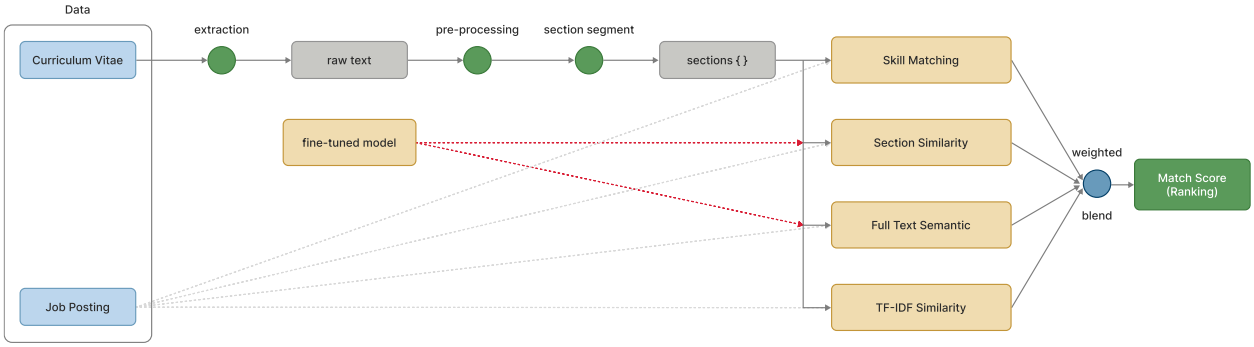

Figure 1.9: Text Classification Pipeline for Matching Flow

CONCLUSION

This chapter has established the theoretical and contextual foundations of the present work. We reviewed the recruitment process, the role of HRM, and the key actors involved, before highlighting the significant limitations of traditional, manual approaches. We then surveyed the enabling technologies: AI, ML, NLP, and Deep Learning, with a particular focus on TF-IDF text representation, skills extraction, semantic similarity via sentence embeddings, and the conceptual link to text classification. Finally, we stated the objectives of the project and described the architecture of the proposed solution.

These foundations will inform the design decisions presented in the following chapter, which is dedicated to the system architecture and UML modeling of the proposed platform.

CHAPTER TWO

SYSTEM ARCHITECTURE AND DESIGN

INTRODUCTION

Having established the theoretical foundations and defined the objectives of the proposed system in the preceding chapter, this chapter focuses on the software engineering modeling of the intelligent recruitment platform. Modeling constitutes a fundamental step in the software development process, as it allows the designer to represent the system at different levels of abstraction before any line of code is written. In this chapter, we follow a structured modeling approach organized into three complementary perspectives. We begin with a requirements specification phase, in which we identify the system actors and define both the functional and non-functional requirements. We then proceed to structural modeling, where we use a class diagram to describe the data structure of the system and the relationships between its entities. This is followed by functional modeling through use case diagrams, which capture what the system does and for whom. Finally, we present the behavioral modeling of the system through activity diagrams, which describe the logic flow of the main processes, and sequence diagrams, which illustrate the interactions between system components during key operations. All diagrams in this chapter are designed following UML notation [33].

1. REQUIREMENTS SPECIFICATION

Requirements specification is the first and most critical phase of the software design process. Its purpose is to precisely define what the system must do, who it must serve, and under what quality constraints it must operate. This phase directly conditions all subsequent design and implementation decisions.

1.1. Identification of System Actors

An actor represents any entity that interacts with the system from the outside. Based on the analysis conducted in the previous chapter, three main actors have been identified for the proposed platform:

Recruiter: A professional or organizational representative who uses the platform to publish job offers and consult the results of the automated candidate evaluation. The recruiter is the primary consumer of the AI matching output.

Candidate: An individual seeking employment who registers on the platform, submits their CV, and applies to published job offers. The candidate is the primary subject of the automated evaluation process.

Administrator: A privileged actor responsible for managing the platform, overseeing user accounts, and ensuring the proper functioning of the system. The administrator does not participate in the recruitment process itself but maintains the integrity of the platform.

1.2. Functional Requirements

Functional requirements describe the specific behaviors and services the system must provide to each actor.

As a Recruiter, the system must allow the user to:

- Register and authenticate using an email and password
- Create, edit, and delete job offers with a title, description, and required skills list
- View the list of candidates who applied to each published position
- Consult the matching score and extracted skills associated with each application
- Update the review status of each application
- Manage their profile information

As a Candidate, the system must allow the user to:

- Register and authenticate using an email and password
- Complete and update their personal profile
- Upload a CV in PDF or DOCX format
- Browse and search published job offers
- Submit an application to a selected job offer
- Track the status of submitted applications and view their matching scores

As an Administrator, the system must allow the user to:

- Authenticate through a secure dedicated login mechanism
- View and manage all registered recruiter and candidate accounts
- Deactivate or delete accounts and job offers that violate platform policies
- Consult platform-level statistics including user counts and application volumes

1.3. Non-Functional Requirements

Non-functional requirements define the quality attributes the system must satisfy regardless of specific functional behavior. The following non-functional requirements have been identified as essential for the proposed platform:

Performance: The system must process CV matching requests and return results within an acceptable response time, even under concurrent user load.

Security: All sensitive data, including user passwords and personal information, must be protected. Authentication must be enforced through JWT token validation, and role-based access control must prevent unauthorized operations.

Scalability: The architecture must be designed in a modular fashion so that individual components, particularly the AI inference engine, can be scaled independently as user demand grows.

Usability: The interface must be intuitive and accessible to users with no technical background, particularly recruiters and candidates who interact with the platform on a daily basis.

Maintainability: The codebase must be organized following separation of concerns principles, making it straightforward to update, debug, or extend individual components without affecting the rest of the system.

2. STRUCTURAL MODELING

Structural modeling describes the static dimension of the system. It focuses on what the system knows, the data it manages, the entities it defines, and the relationships that exist between them. In UML, the class diagram is the standard notation for expressing this structural view.

2.1. Class Diagram

The class diagram presented in **Figure 2.1** defines the main entities of the platform, their attributes, their methods, and the relationships that connect them.

The **User** class serves as the base class for all registered users of the platform. It holds the attributes shared across all roles: a unique identifier, an email address, a hashed password, a full name, a role type enumeration (Recruiter, Candidate, Admin), and a registration date. The User class is specialized through inheritance into three subclasses: Recruiter, Candidate, and Administrator.

The **Recruiter** class inherits from User and adds attributes specific to the recruiter profile, including the company name and a professional description. It is associated with a collection of

JobOffer instances through a one-to-many relationship, reflecting the fact that one recruiter may publish multiple job offers.

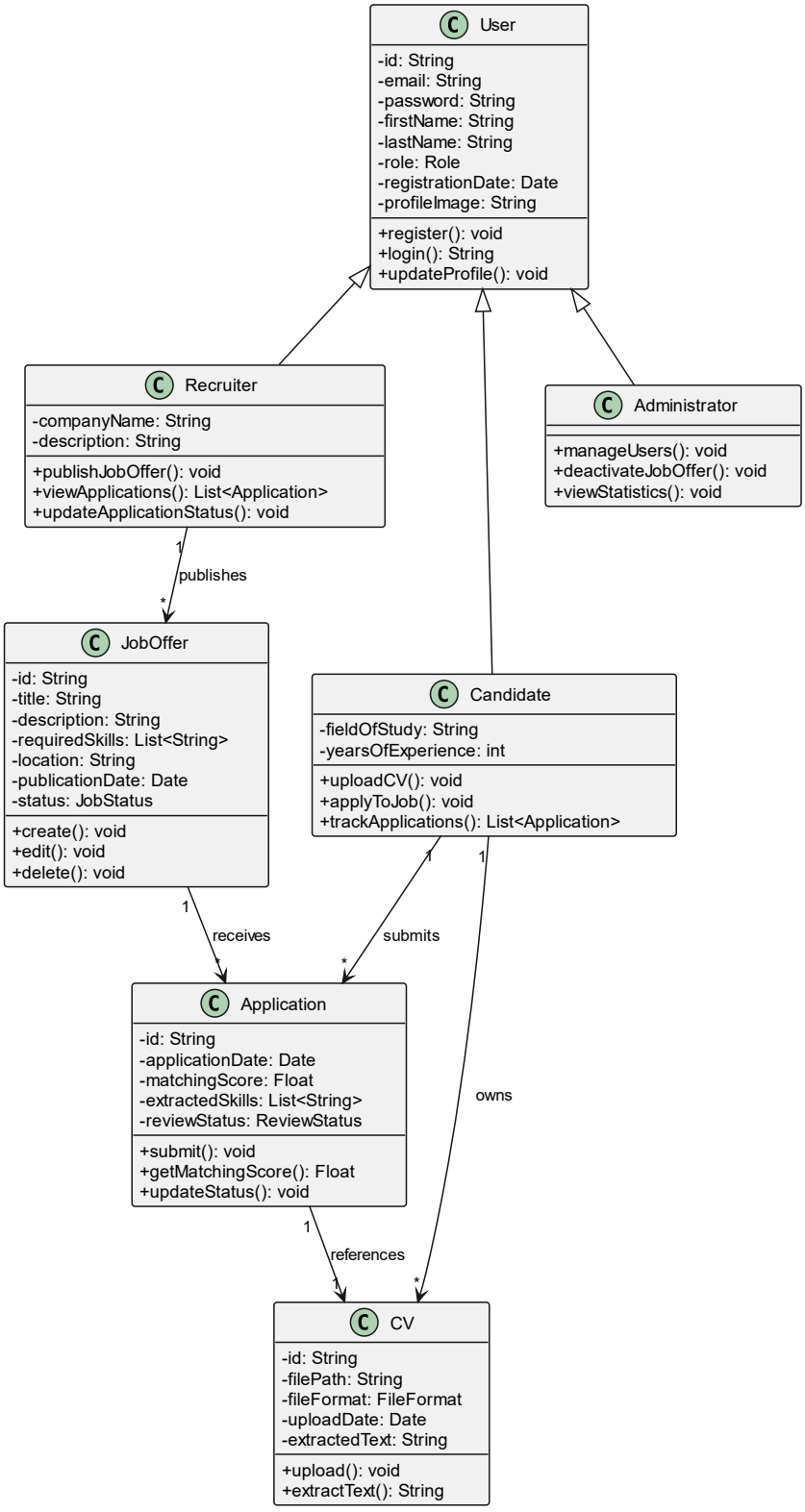

Figure 2.1: Class Diagram

The **Candidate** class inherits from User and carries additional attributes such as field of study, years of experience, and a reference to an uploaded CV document. A Candidate may submit multiple Applications, establishing a one-to-many relationship between Candidate and Application.

The **JobOffer** class represents a job position published on the platform. Its attributes include the job title, a textual description, a list of required skills, the publication date, the location, and the current status of the offer. Each JobOffer is linked to exactly one Recruiter and may receive multiple Applications, resulting in a one-to-many relationship with the Application class.

The **Application** class models the act of a candidate applying to a specific job offer. It stores a reference to the submitted CV document, the matching score computed by the AI engine, the list of skills extracted from the CV, the application date, and the review status assigned by the recruiter. Each Application is associated with exactly one Candidate and one JobOffer.

The **CV** class represents the document submitted by a candidate. It stores the file path, the upload date, the file format, and the raw extracted text content used by the AI engine for analysis.

2.2. Relational Database Model

The relational database model of the platform was designed based on the presented class diagram previously.

The relational schema of the system is presented as follows:

- **User** (id_u, firstName, lastName, email, password, role, registrationDate, profileImage)
- **Recruiter** (id_u*, companyName, description)
- **Candidate** (id_u*, fieldOfStudy, yearsOfExperience)
- **Administrator** (id_u*)
- **JobOffer** (id_j, title, description, requiredSkills, location, publicationDate, status, id_u**(recruiter id)*)
- **CV** (id_cv, filePath, fileFormat, uploadDate, extractedText, id_u**(candidate_id)*)
- **Application** (id_app, applicationDate, matchingScore, extractedSkills, reviewStatus, id_j*, id_u**(candidate_id)*, id_u**(recruiter id)*)

attribute : Primary Key

attribute : Foreign Key

attribute*: Primary Key + Foreign Key

3. FUNCTIONAL MODELING

Functional modeling describes the behavioral dimension of the system from the user's perspective. It focuses on what the system does and for whom. In UML, use case diagrams are the standard notation for capturing this view. Each use case diagram presents one actor and the set of functionalities available to them, along with the relationships between use cases such as include and extend.

3.1. Use Case Diagram

3.1.1. Authentication

Authentication serves as the foundational security layer of the system, verifying the identities of users before granting access to restricted features and data. This process ensures that system interactions are secure and that users are properly authorized based on their assigned roles. **Figure 2.2** illustrates the core use case diagram for the authentication process, mapping out the standard login and register workflows.

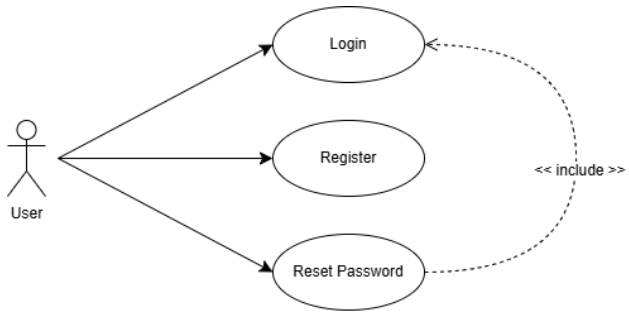

Figure 2.2: Authentication Use Case Diagram

To maintain clarity and avoid unnecessary redundancy throughout this document, the authentication mechanism is established here as a universal prerequisite. Consequently, subsequent actor-specific use case diagrams will omit the authentication use case, as its execution is implicitly assumed for all protected system interactions.

3.1.2. Recruiter Use Cases

The use case diagram presented in **Figure 2.3** captures the full set of interactions available to the Recruiter actor. The Recruiter must first authenticate through the login use case before accessing any protected functionality. Once logged in, the Recruiter can manage job offers through three related use cases: create a job offer, edit a job offer, and delete a job offer, all of which extend the manage job offers use case. The Recruiter can also consult the list of applications received for each offer and, for each application, view the matching score and extracted skills returned by the AI engine. Finally, the Recruiter can update the review status of any application and manage their profile information.

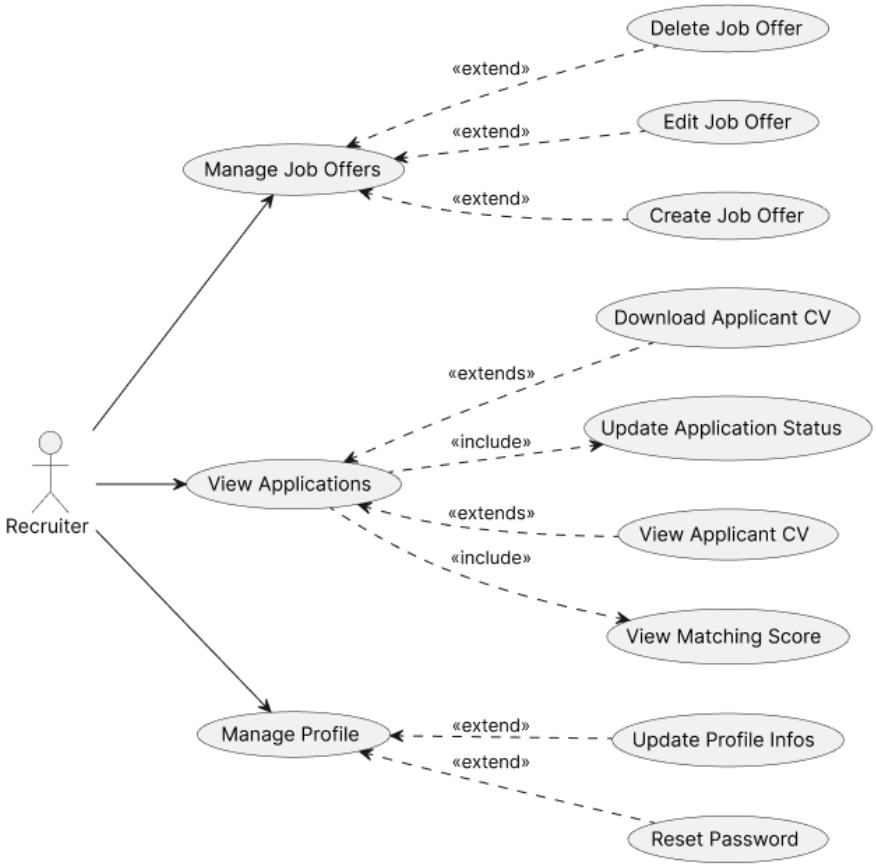

Figure 2.3: Recruiter Use Case Diagram

3.1.3. Candidate Use Cases

Figure 2.4 presents the use case diagram for the Candidate actor. The Candidate can register on the platform through the registration use case, which includes an email verification step. After logging in, the Candidate can complete their profile and upload a CV document. The core use case available to the Candidate is applying to a job offer, which includes browsing available offers and selecting one, then submitting an application with an attached CV. The Candidate can also track the status of previously submitted applications and consult the matching score assigned to each one.

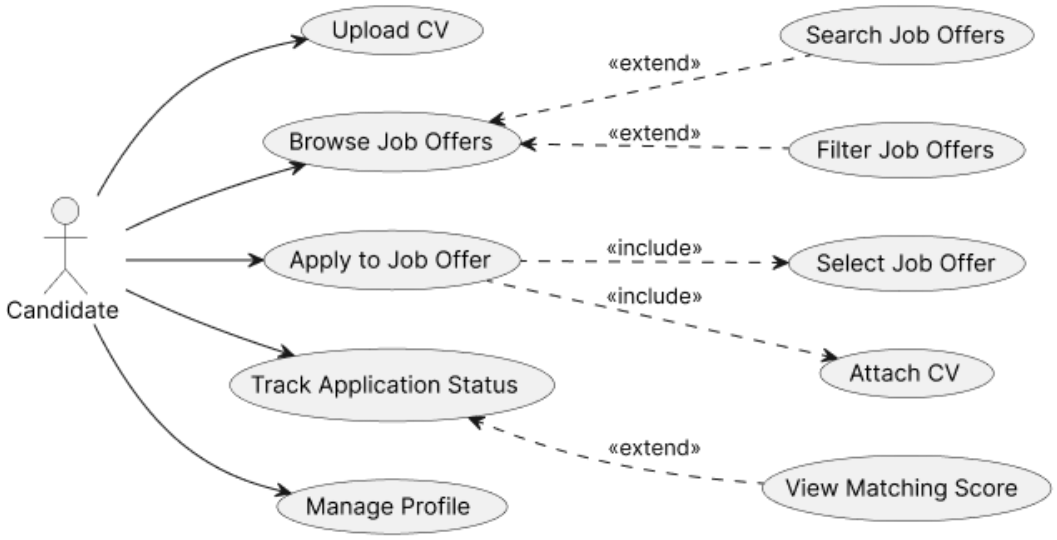

Figure 2.4: Candidate Use Case Diagram

3.1.4. Administrator Use Cases

Figure 2.5 illustrates the use case diagram for the Administrator actor. The Administrator authenticates through a dedicated login mechanism separate from the regular user login. Once authenticated, the Administrator can view the full list of registered recruiters and candidates, edit or delete any user account, and deactivate or remove job offers that violate platform policies. The Administrator can also consult platform-level statistics covering user registrations, job offer counts, and application volumes.

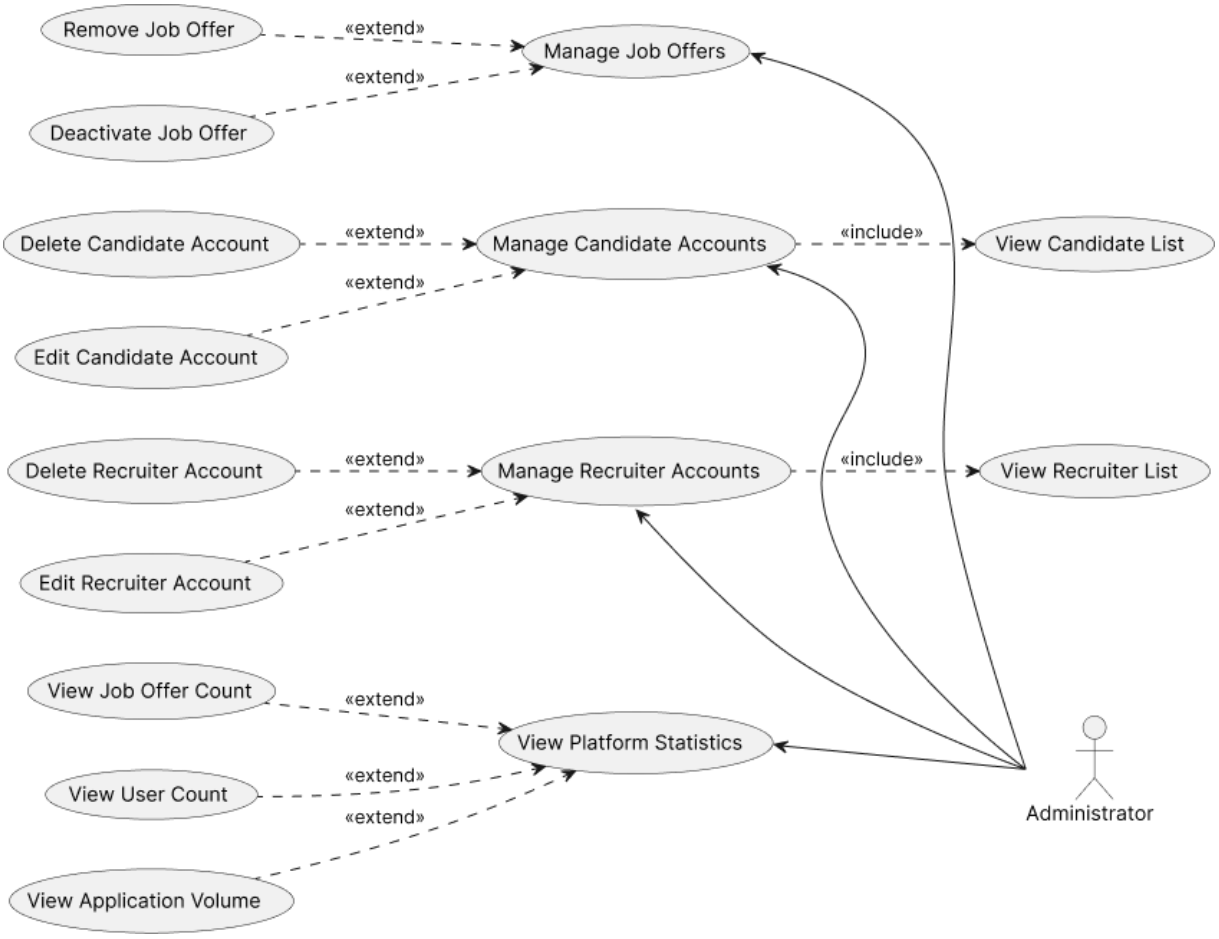

Figure 2.5: Administrator Use Case Diagram

4. SYSTEM ARCHITECTURE AND COMPONENT DESIGN

Beyond the abstract modeling of the system, it is essential in a software engineering context to describe the concrete technical structure of the platform. This section presents the design of the two main software components of the system: the back-end server, which handles business logic, data persistence, and communication with the AI inference engine, and the end-users application, which provides the interface through which recruiters and candidates interact with the platform.

4.1. Back-end Server Design

The back-end server constitutes the core processing layer of the platform. Built on Node.js with the Express framework, it acts as the central mediator between the front-end application, the MongoDB database, and the AI inference engine. It is internally organized into four components that operate in a layered fashion: the router, the middleware layer, the controller layer, and the database access layer.

4.1.1. Back-end Server Design

The router module is responsible for receiving incoming HTTP requests and directing them to the appropriate controller functions based on the request method and URL path. The API is organized into two main groups. Public endpoints are accessible without authentication and include the registration and login routes. Protected endpoints require a valid JWT token and are further subdivided by user role into recruiter, candidate, and administrator endpoint groups.

Recruiter endpoints support the creation, editing, and deletion of job offers, as well as the retrieval of application lists with their associated matching scores. Candidate endpoints handle CV upload, profile management, and application submission and tracking. Administrator endpoints provide access to user account management and platform monitoring functionalities.

4.1.2. Business Logic and Controller Layer

The controller layer encapsulates the business logic of the application. Upon receiving a validated request from the router, the appropriate controller function is invoked. It processes the incoming data, applies the relevant business rules, and coordinates the necessary interactions with the database and the AI inference engine. In the context of CV evaluation, the controller is responsible for retrieving the job description from the database, forwarding both the CV content and the job description to the AI inference engine via an HTTP request, receiving the computed matching score and extracted skills, and persisting the results as a new Application record. The controller then formats and returns the appropriate HTTP response to the client.

4.1.3. Middleware and Request Processing

Middleware functions intercept incoming requests before they reach the controller layer, enabling cross-cutting concerns to be handled in a modular and reusable manner. The middleware stack of the proposed system includes four components. The JWT Verifier validates the authentication token included in the HTTP request header and rejects any request that does not carry a valid token. The Role Guard ensures that authenticated users can only access endpoints that correspond to their assigned role, preventing unauthorized cross-role operations. The Input Validator verifies that all required fields are present and correctly formatted in the request body, returning descriptive error messages when validation fails. The File Parser handles multipart form data submitted during CV upload operations, extracting the binary file content and making it available to the controller for downstream processing.

4.1.4. Database Access Layer

All interactions with the MongoDB database are handled through a dedicated database access layer that abstracts the specifics of the underlying storage system. This layer exposes a clean and consistent interface for performing create, read, update, and delete operations on the User, JobOffer, Application, and CV collections. By centralizing all database communication in this layer, the system decouples the business logic from the database implementation, reduces the risk of injection vulnerabilities, and simplifies the process of switching to a different database system in future versions of the platform.

4.2. End-users Application Design

4.2.1. Application Pages and Navigation Structure

The end-users application is organized into public pages, accessible to unauthenticated visitors, and role-specific private pages, accessible only after successful login. Public pages include the platform landing page, the user registration form, and the login screen. Once authenticated as a Recruiter, the user gains access to a job offer management dashboard for creating and editing positions, an application review interface for consulting ranked candidate lists, and a profile management page. Once authenticated as a Candidate, the user gains access to a profile and CV management page, a job browsing and search interface, an application submission page, and an application tracking dashboard where they can monitor the status and matching score of each submitted application.

5. BEHAVIORAL MODELING

Behavioral modeling captures the dynamic dimension of the system. While functional modeling describes what the system does and structural modeling describes what data it manages, behavioral modeling describes how the system behaves when events occur. Two complementary UML diagram types are used in this section. Activity diagrams describe the step-by-step logic of the main processes, including decision points and alternative flows.

5.1. Activity Diagrams

5.1.1. CV Submission and Automated Evaluation

Figure 2.6 presents the activity diagram for the CV submission and automated evaluation process. The process begins when an authenticated candidate selects a job offer and initiates an application. The system first checks whether the candidate has already submitted an application for the same offer if so, a notification is displayed and the process terminates. Otherwise, the candidate proceeds to upload their CV document. The system validates the file format, if validation fails, the candidate is prompted to upload a corrected file. Upon successful validation, the system forwards the CV text and the job description to the AI inference engine. The engine extracts the candidate's skills and computes a semantic similarity score between the CV and the job description. The system stores the application record with the computed results, updates the candidate's application dashboard, and sends a notification to the recruiter informing them of the new submission.

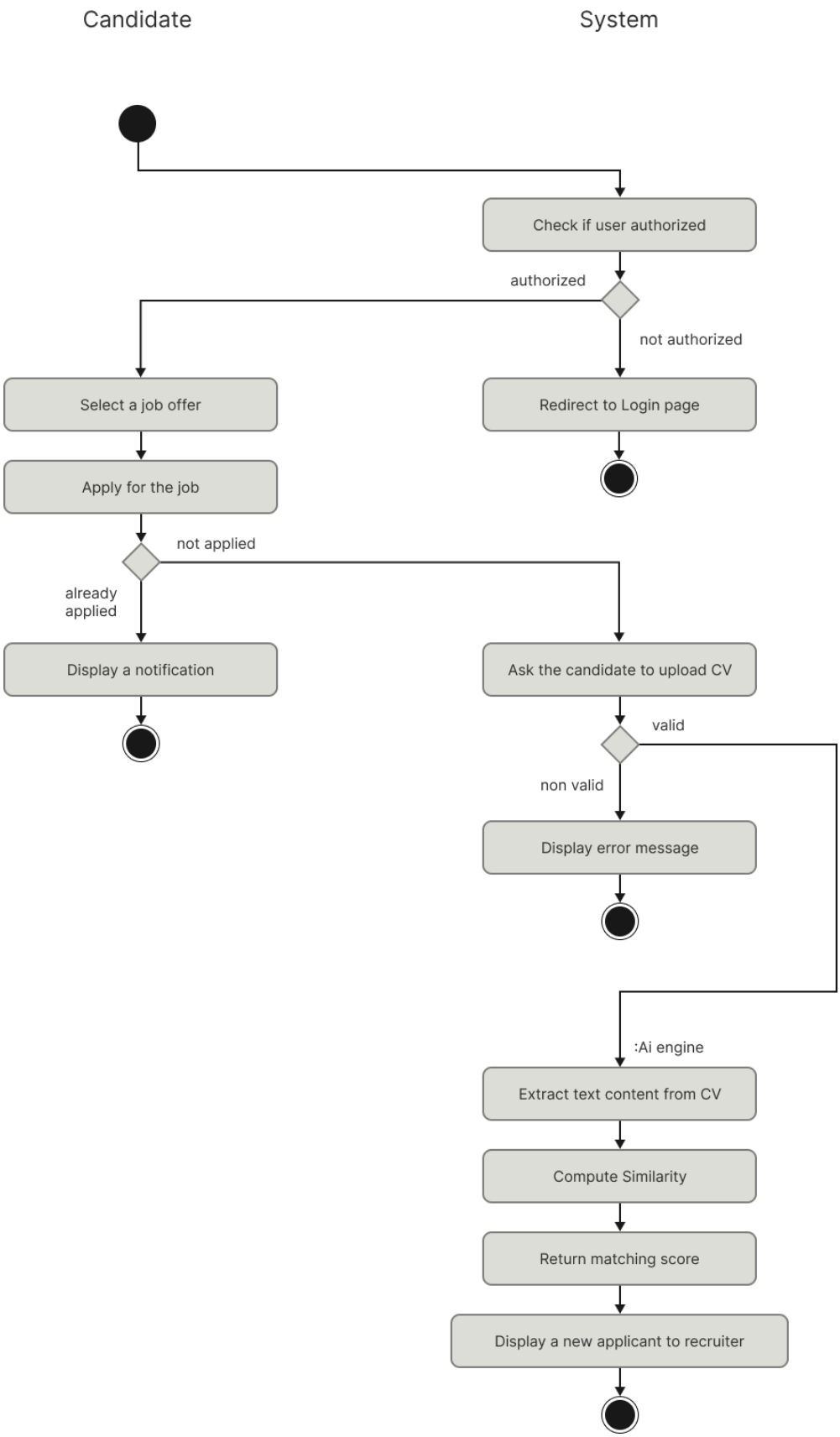

Figure 2.6: CV Submission and Automated Evaluation Activity Diagram

5.1.2. Job Offer Publication Process

Figure 2.7 illustrates the activity flow for publishing a new job offer. The recruiter navigates to the job management page and selects the option to create a new position. The system presents a form requesting the job title, description, required skills, and location. The recruiter fills in the form and submits it. The system performs input validation; if any required field is missing or incorrectly formatted, an error message is displayed and the recruiter is invited to correct the input before re-submitting. Upon successful validation, the job offer is stored in the database, assigned a published status, and made immediately visible to all registered candidates on the platform.

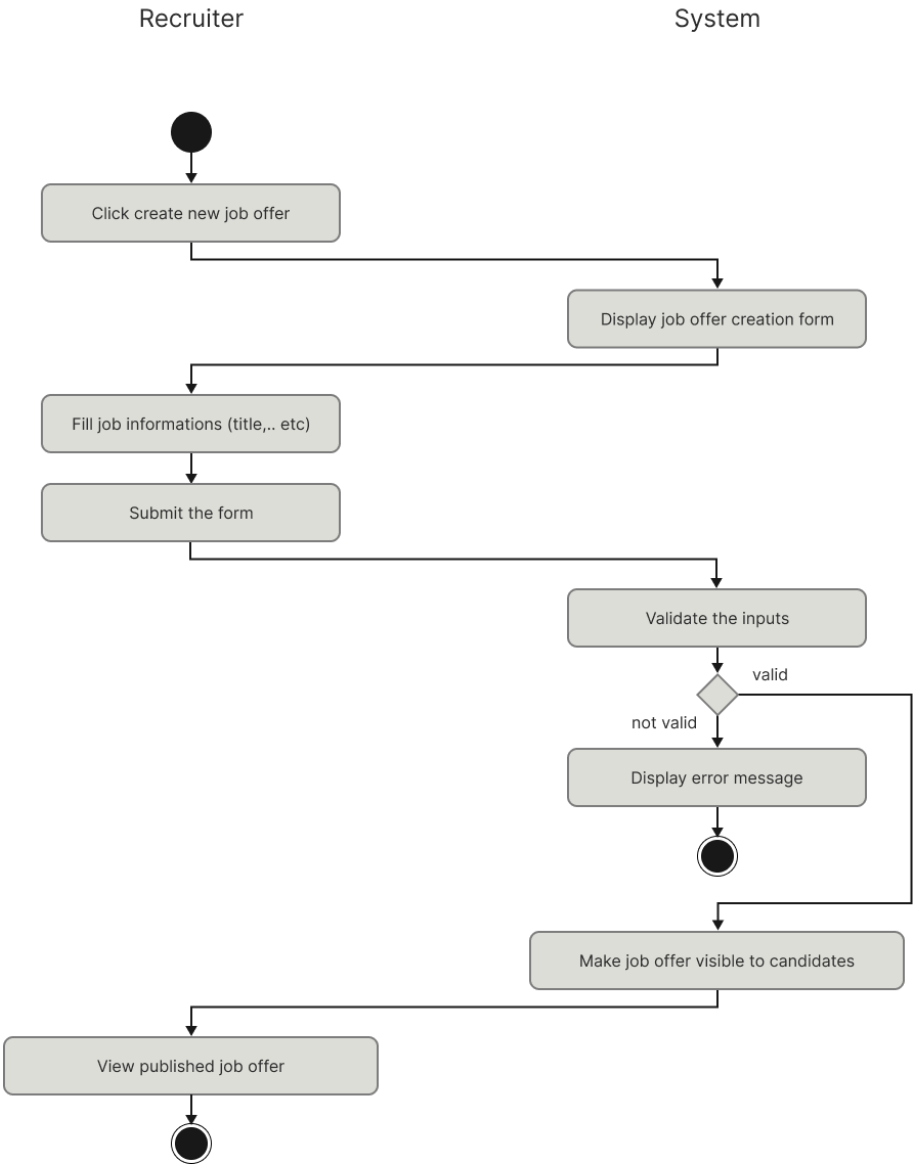

Figure 2.7: Job Offer Publication Process Activity Diagram

CONCLUSION

This chapter has presented a comprehensive software engineering modeling and design of the proposed intelligent recruitment platform. The requirements specification phase allowed us to precisely define the system actors and establish both the functional and non-functional requirements that govern the platform's expected behavior. The structural modeling phase produced a class diagram capturing the data entities of the system and the object-oriented relationships between them. The functional modeling phase described the interactions available to each user role through use case diagrams with include and extend relationships. The system architecture and component design section provided a concrete technical description of the back-end server and the end-users application, detailing the routing structure, business logic, middleware pipeline, database access layer, and navigation organization. Finally, the behavioral modeling phase utilized activity diagrams to describe the decision logic and operational workflows of the main processes.. Together, these modeling artifacts constitute a rigorous and complete blueprint of the system, directly informing the implementation and deployment decisions presented in the following chapter.

CHAPTER THREE

SYSTEM IMPLEMENTATION AND DEPLOYMENT

INTRODUCTION

Having established the theoretical foundations in Chapter One and produced a complete software engineering model of the system in Chapter Two, this chapter is dedicated to the concrete realization of the proposed intelligent recruitment platform. The objective is to document the transition from design artifacts to a fully functional and deployable software system. We begin by presenting the development environment and the tools used throughout the implementation process. We then describe the technology stack adopted for each component of the system, justifying the key choices made. The third section covers the implementation of the system's main components, namely the back-end server, the front-end application, and the AI inference engine, highlighting the most significant technical aspects of each. The fourth section presents the application through a collection of screenshots illustrating the key interfaces available to each user role. Finally, the chapter concludes with a description of the deployment architecture and the services used to make the platform accessible online.

1. DEVELOPMENT ENVIRONMENT AND TOOLS

The quality and efficiency of a software development process depend not only on the technologies used but also on the tools and environment that support the development workflow. This section presents the tools adopted throughout the implementation of the proposed platform.

The development of the platform was carried out using the following tools:

- **Google Antigravity:** An agent-first IDE developed by Google that forks the open-source Visual Studio Code architecture. It was chosen as the primary development environment to leverage autonomous AI engineering agents operating across the editor, terminal, and browser. The platform's deep context learning and multi-surface automation were used extensively to plan, implement, and debug the application’s codebase.

- **Postman:** A collaboration platform for API development and testing. Postman was used to design, test, and validate all back-end API endpoints before integrating them with the front-end application. It allowed the team to verify request and response structures, test authentication flows, and simulate edge cases in a controlled environment [27].
- **Google Chrome DevTools:** The built-in developer tools of the Google Chrome browser were used for front-end debugging, network request inspection, and performance profiling during the development of the React application.
- **Git:** A distributed version control system used to track all changes made to the source code throughout the development lifecycle. Git enabled the team to maintain a clean history of modifications, manage feature branches, and revert to stable versions when needed [25].
- **GitHub:** A cloud-based hosting platform for Git repositories. GitHub was used as the central remote repository for all project components, providing a collaborative workspace and serving as a backup for the entire codebase [26].

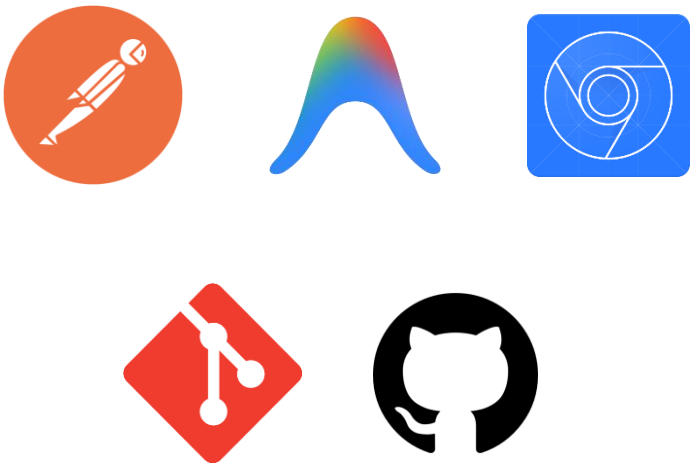

Figure 3.1: Development Environment Tools

2. TECHNOLOGY STACK

The technology stack of a software system refers to the set of programming languages, frameworks, libraries, and services used to build and run it. The stack adopted for the proposed platform was selected based on criteria of maturity, community support, performance, and suitability for the specific requirements of each component.

2.1. Back-end Technologies

The back-end server was built using the following technologies:

Node.js: A JavaScript runtime built on Chrome's V8 engine. Node.js was chosen for its non-blocking, event-driven architecture, which makes it highly suitable for building scalable network applications that handle multiple concurrent requests efficiently [16].

Express.js: A minimal and flexible Node.js web application framework. Express was used to define the API routing structure, register middleware functions, and organize the controller logic. Its simplicity and extensibility made it an ideal choice for building a RESTful API [17].

MongoDB: A document-oriented NoSQL database system. MongoDB was selected for its flexible schema design, which accommodates the evolving data structures of a recruitment platform, and for its native support for JSON-like documents, which aligns naturally with the JavaScript ecosystem of the back-end [18].

Figure 3.2: Back-end Technology Stack

2.2. Front-end Technologies

The end-users application was developed using the following technologies:

- **React:** A JavaScript library for building user interfaces, developed and maintained by Meta. React was chosen for its component-based architecture, which promotes code reusability and modularity, and for its virtual DOM mechanism, which ensures efficient and performant UI updates in response to data changes [19].
- **Vite:** A modern front-end build tool that provides an extremely fast development server and optimized production builds. Vite was used as the build tool for the React application, significantly improving the development experience through near-instant hot module replacement [20].
- **Tailwind CSS:** A utility-first CSS framework that enables developers to build custom user interfaces directly in the markup without writing custom stylesheets. Tailwind was used to style all components of the front-end application, ensuring visual consistency and rapid UI development [21].
- **Axios:** A promise-based HTTP client for JavaScript. Axios was used to handle all communication between the front-end and the back-end API, providing a clean and consistent interface for sending requests and handling responses, including automatic JSON serialization and error handling [22].

Figure 3.3: Front-end Technology Stack

2.3. AI Engine Technologies

The AI inference engine was developed as an independent microservice using the following technologies:

- **Python:** A high-level, general-purpose programming language widely adopted in the field of machine learning and natural language processing. Python was chosen as the implementation language of the AI engine for its rich ecosystem of scientific libraries and its native support for NLP frameworks [24].
- **Flask:** A lightweight Python web framework used to expose the AI engine's functionalities as a RESTful HTTP service. Flask was chosen for its simplicity and minimal overhead, allowing the AI engine to be developed and deployed as a standalone microservice [23].
- **spaCy:** An industrial-strength NLP library for Python. spaCy was used for named entity recognition and skills extraction from raw CV text, leveraging its pre-trained language models to identify and classify relevant entities with high accuracy and speed [13].
- **Sentence Transformers:** A Python library built on top of the Hugging Face Transformers framework. Sentence Transformers was used to generate dense vector embeddings from both the CV text and the job description, enabling semantic similarity computation beyond simple keyword matching [9] [14].
- **Scikit-learn:** A machine learning library for Python. Scikit-learn was used for computing cosine similarity between the sentence embeddings of the CV and the job description, producing the final matching score returned to the back-end [12].

Figure 3.4: AI Engine Technology Stack

3. SYSTEM IMPLEMENTATION

This section presents the most significant technical aspects of the implementation of each major component of the platform. Rather than reproducing the entire codebase, we focus on the key implementation decisions and the most representative code excerpts that illustrate how the design choices described in Chapter Two were concretely realized.

3.1. Back-end Server Implementation

The back-end server was structured following the layered architecture defined in the design chapter, organizing the codebase into distinct modules for routing, middleware, controllers, and database access.

The API routing layer defines all endpoints grouped by resource and role. Each route file registers the applicable middleware functions before delegating to the appropriate controller. Protected routes require the JWT verification middleware to run first, followed by the role guard middleware, ensuring that authentication and authorization are enforced consistently across all sensitive endpoints.

The controller layer implements the business logic for each operation. The most technically significant controller is the application controller, which orchestrates the full CV matching pipeline. Upon receiving a valid application request, the controller extracts the text content from the uploaded CV file, retrieves the corresponding job description from the database, and forwards both to the AI inference engine via an HTTP POST request. The matching score and extracted skills returned by the engine are then stored as part of the new Application document in MongoDB.

The middleware layer provides reusable request processing functions. The JWT verification middleware decodes the token from the Authorization header, verifies its signature using the server's secret key, and attaches the decoded user payload to the request object for downstream use. The role guard middleware reads the user's role from the decoded payload and rejects requests from users whose role does not match the required access level for the requested endpoint.

3.2. Front-end Application Implementation

The front-end application was structured as a React single-page application with a clear separation between pages, reusable components, and service modules responsible for API communication.

The application uses React Router to implement client-side navigation between pages. Route protection is enforced through a custom PrivateRoute component that reads the JWT token from local storage and verifies its presence before rendering protected pages. If no valid token is found, the user is automatically redirected to the login page.

The state management of the application relies on React's built-in hooks, primarily useState and useEffect, for managing local component state and handling asynchronous data fetching. For globally shared state such as the authenticated user's profile and role, the React Context API was used to provide a centralized and accessible data store across the component tree without introducing additional external dependencies.

The CV upload and application submission flow represents the most complex front-end interaction. The component manages a multi-step form that first prompts the candidate to confirm the job offer details, then presents a file upload interface. Upon submission, the component sends a multipart form data request to the back-end using Axios and displays a loading indicator while awaiting the matching score response from the server.

3.3. AI Engine Implementation

The AI inference engine was implemented as a standalone Flask application exposing a single POST endpoint that accepts a CV text and a job description and returns the computed matching score along with the list of extracted skills.

The skills extraction module uses spaCy's named entity recognition pipeline, augmented with a custom skills vocabulary, to identify technical and professional skills mentioned in the CV text. The extracted entities are filtered and normalized to produce a clean list of skills that is returned alongside the matching score.

The semantic similarity module converts both the CV text and the job description into dense vector representations using a pre-trained Sentence Transformer model. The cosine similarity between the two vectors is then computed using scikit-learn and normalized to a score between zero and one. A score close to one indicates a high degree of semantic alignment between the candidate's profile and the job requirements, while a score close to zero indicates a significant mismatch.

The Flask application is configured to run as an independent service, decoupled from the main back-end server. This architectural decision ensures that the AI engine can be updated, re-trained, or replaced independently without requiring any changes to the rest of the platform.

4. APPLICATION PRESENTATION

This section presents the proposed platform through a series of screenshots illustrating the key interfaces available to each user role. The screenshots were captured from the deployed version of the application and represent the final state of the implemented system.

4.1. Candidate Interface

The candidate interface provides a seamless experience for job seekers to register, manage their profile, browse job offers, and submit applications.

Registration and Login Page: The registration page presents a clean form collecting the candidate's full name, email address, password, and role selection. Input validation is enforced in real time, providing immediate feedback for incorrectly formatted fields before submission.

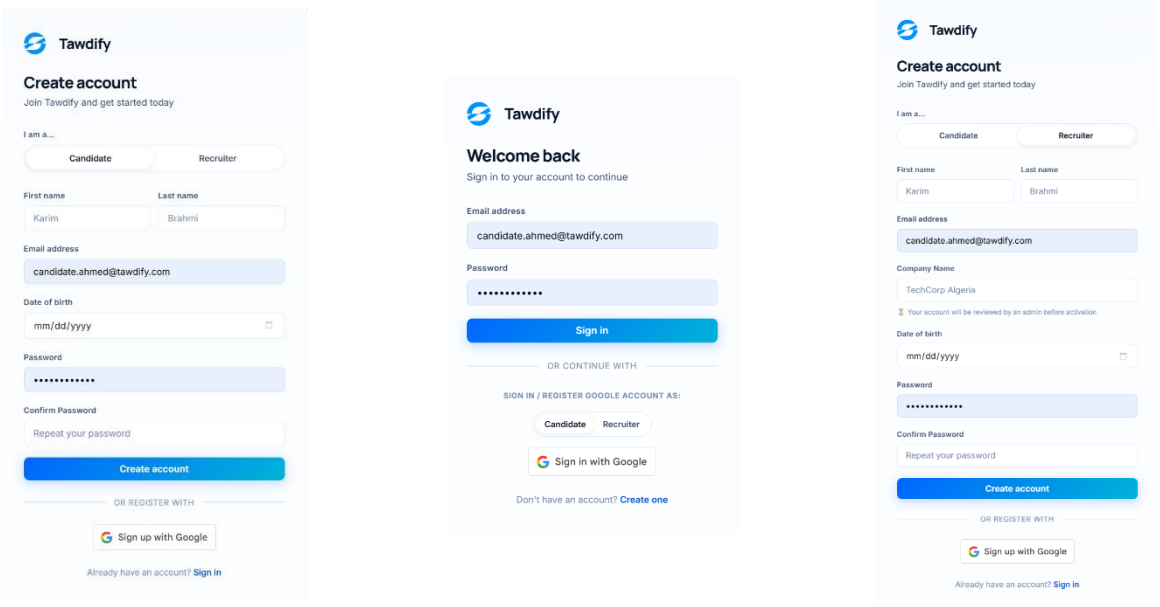

Figure 3.5: Registration and Login Page

Job Browsing Page: The job browsing page displays all currently published job offers in a card-based layout. Each card presents the job title, required skills, location, and publication date. A search bar and filter options allow candidates to narrow down the displayed offers by keyword or skill.

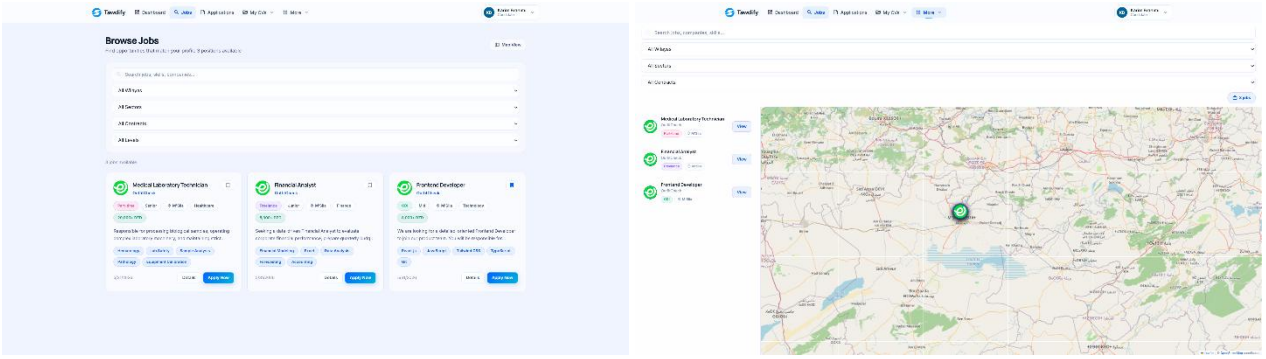

Figure 3.6: Job Browsing Pages

CV Upload and Application Page: The application page guides the candidate through a two-step process: confirming the selected job offer and uploading their CV document. Upon successful submission, the computed matching score is displayed to the candidate alongside a confirmation message.

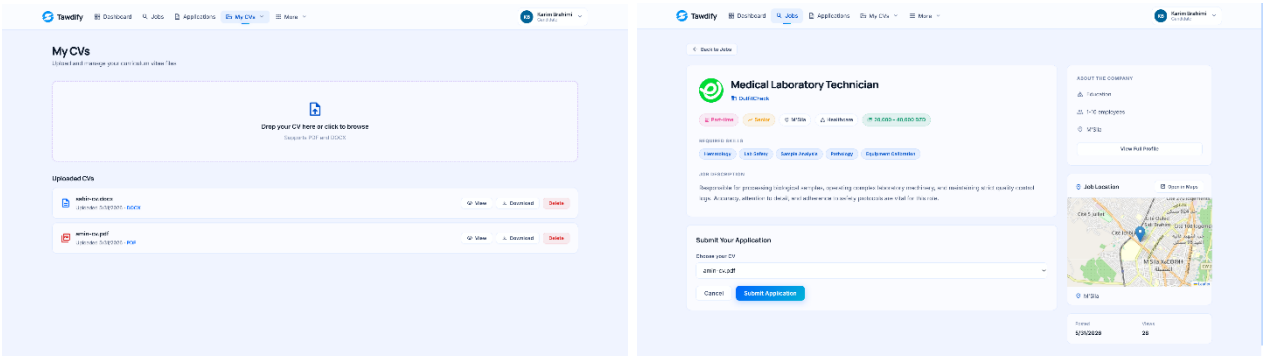

Figure 3.7: CV Upload and Application Pages

Application Tracking Dashboard: The application tracking dashboard presents a table of all submitted applications, showing the job title, application date, matching score, and current review status for each entry.

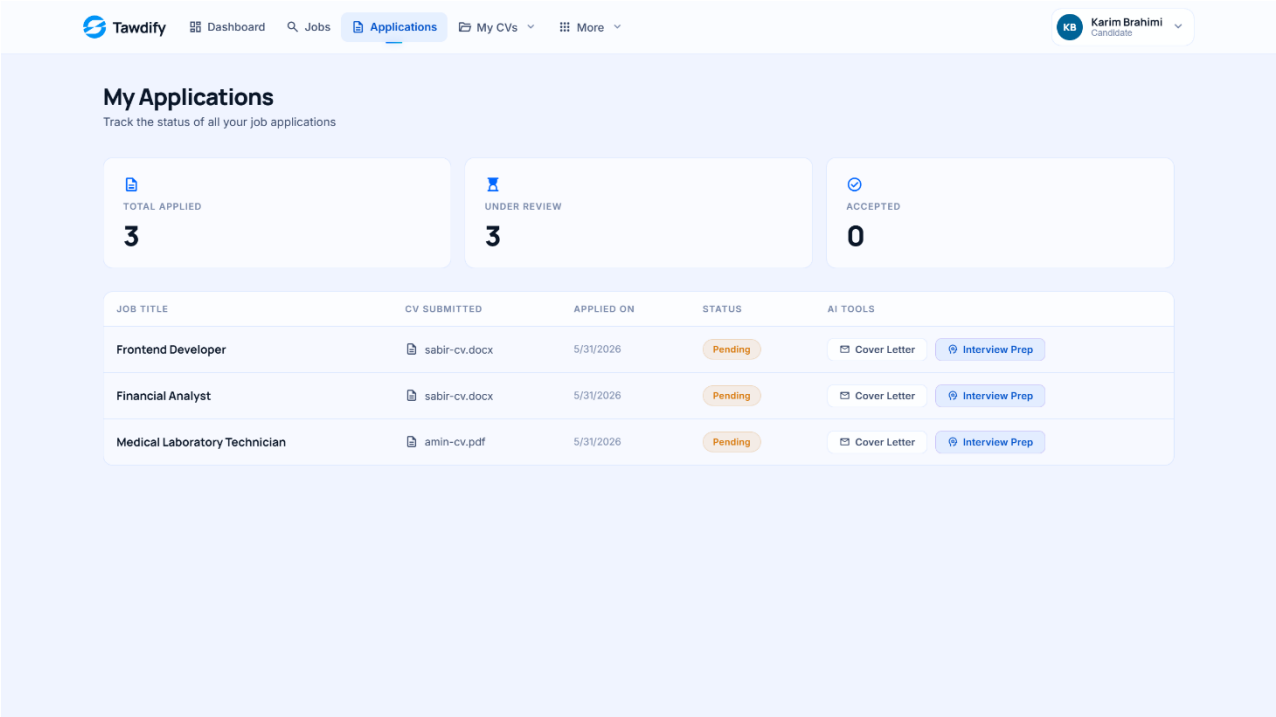

Figure 3.8: Application Tracking Dashboard

4.2. Recruiter Interface

The recruiter interface provides tools for managing job offers and reviewing ranked candidate applications.

Job Offer Management Page: The job offer management page displays all job offers published by the recruiter in a clean card format. Each card presents the job title, a short description preview, required skills, and key tags (like location, contract type, and live status). Action buttons at the bottom allow the recruiter to view applicants, edit, or delete each offer.

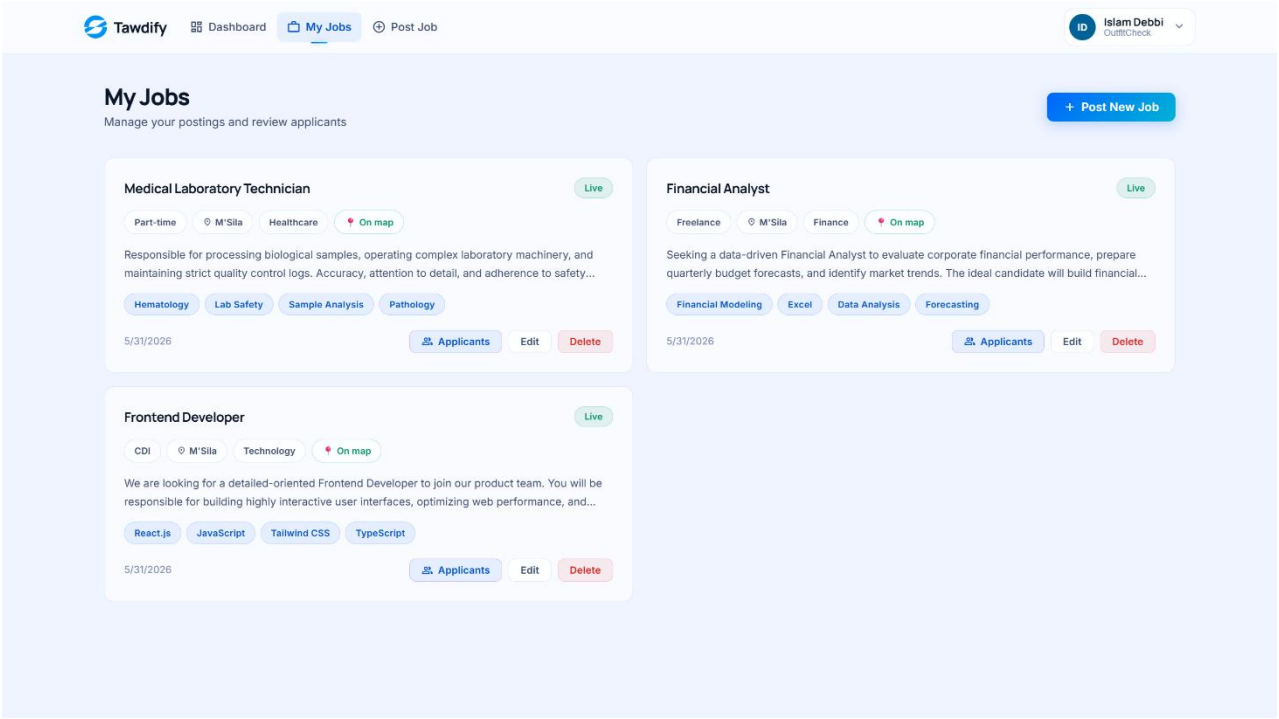

Figure 3.9: Job Offer Management Page

Candidate Ranking Page: The candidate ranking page presents the full list of applications received for a selected job offer, sorted by matching score in descending order. For each application, the recruiter can view the candidate's name, extracted skills, matching score, and current review status, and update the status to shortlisted or rejected.

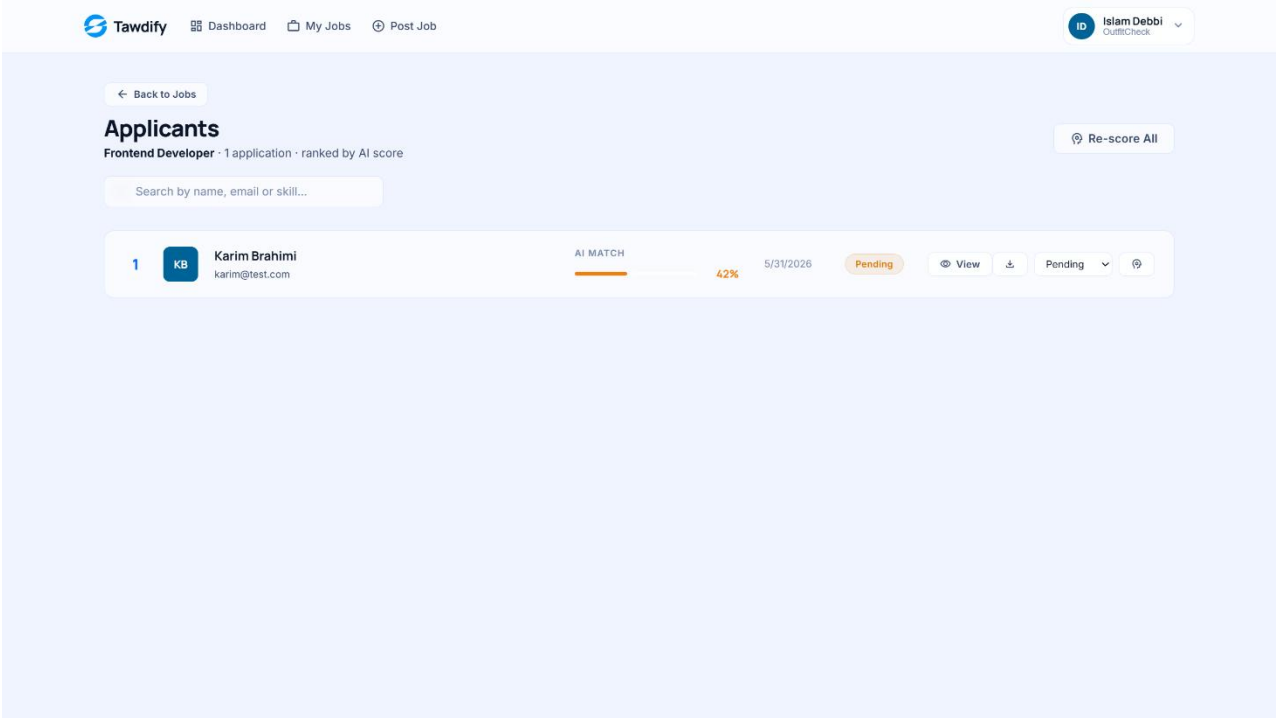

Figure 3.10: Candidate Ranking Page

4.3. Administrator Interface

The administrator interface provides a centralized control panel for platform management.

Dashboard Overview Page: The dashboard home page presents a summary of the platform's current state, including the total number of registered candidates, registered recruiters, published job offers, and submitted applications displayed as metric cards.

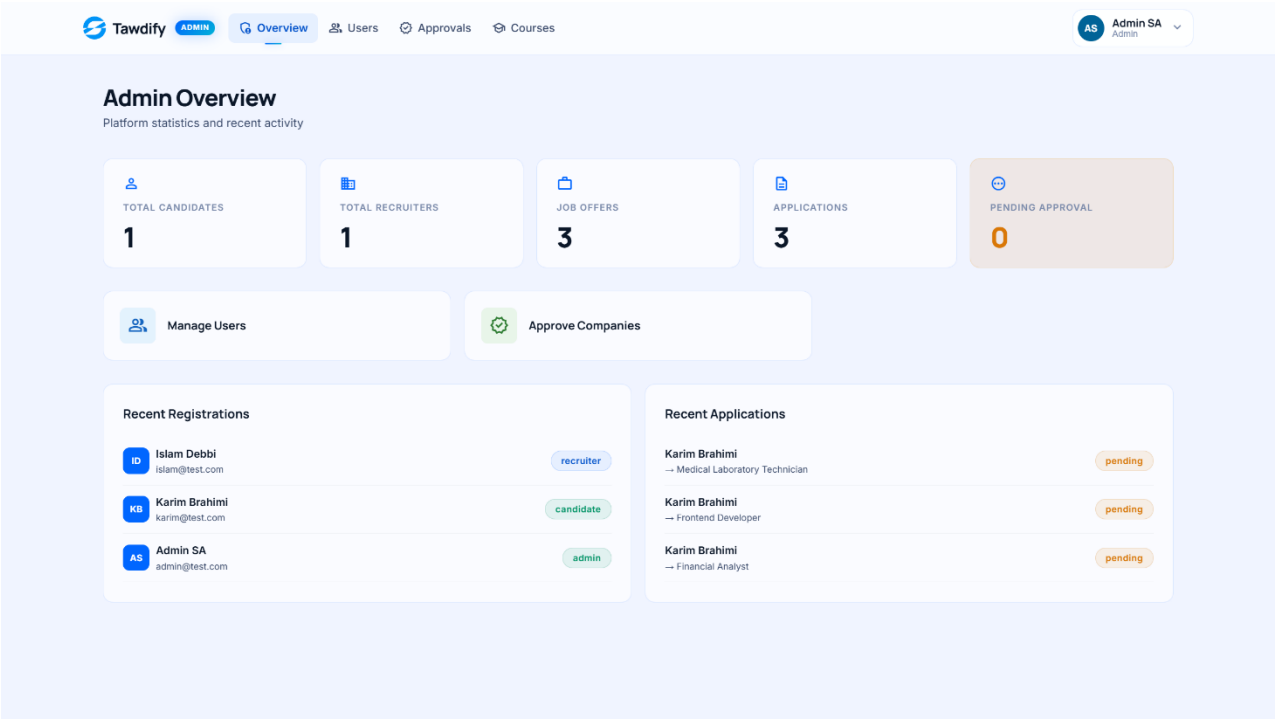

Figure 3.11: Dashboard Overview Page

User Management Page: The user management page presents a full tabular listing of all registered users, filterable by role. Each row displays the user's name, email address, role, and registration date, with action buttons for editing and deleting accounts.

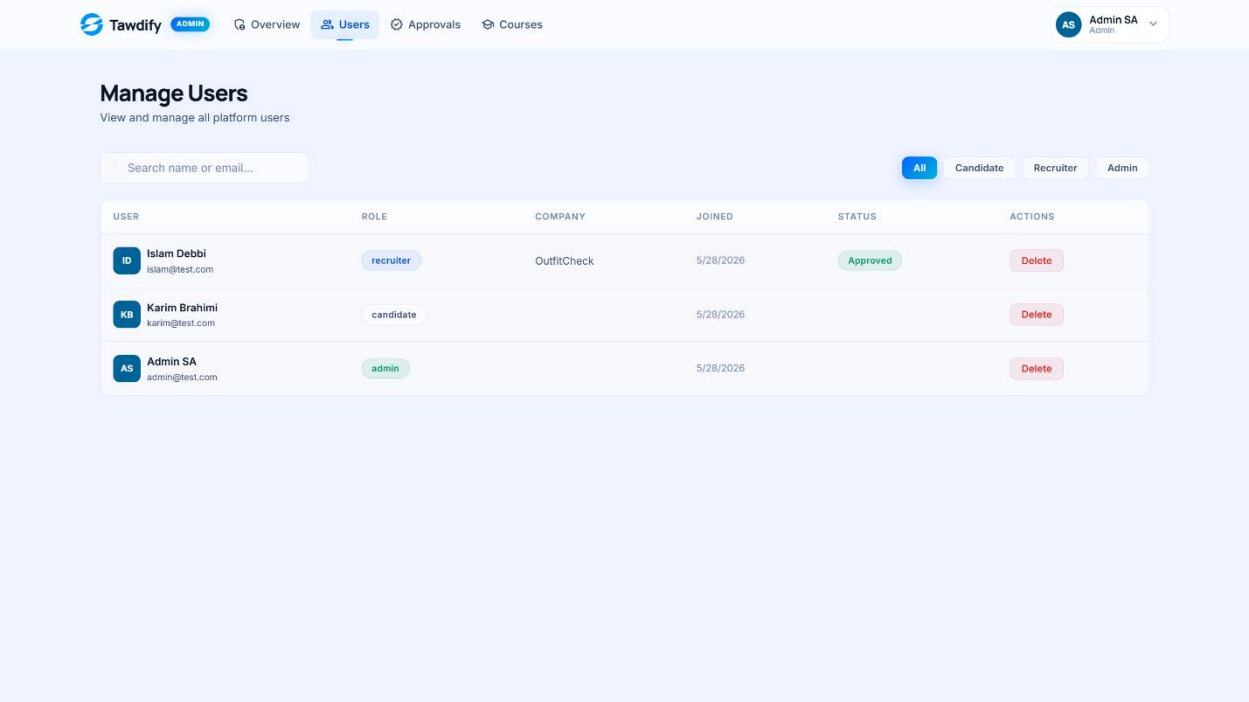

Figure 3.12: AI Engine Technology Stack

CONCLUSION

This chapter has documented the complete implementation and deployment of the proposed intelligent recruitment platform. The development environment and tools section presented the editors, testing platforms, and version control systems used throughout the implementation process. The technology stack section justified the selection of each technology adopted for the back-end server, the front-end application, and the AI inference engine, covering Node.js, Express, MongoDB, React, Python, Flask, spaCy, and Sentence Transformers among others. The application presentation section illustrated the realized interfaces through figures covering all three user roles. Together, the three chapters of this report constitute a complete software engineering documentation of the proposed system, from theoretical grounding and requirements analysis through design modeling to concrete implementation.

Conclusion

This thesis presented the design and implementation of an AI-powered web platform for automated curriculum vitae evaluation and job matching. The work was motivated by the recognized limitations of traditional recruitment methods, which are time-consuming, inconsistent, and poorly suited to the growing volume of job applications in today's professional environment.

The first chapter established the theoretical foundations of the project, covering the recruitment process, its main challenges, and the key AI technologies involved, namely natural language processing, named entity recognition, sentence embeddings, and semantic similarity. The second chapter produced a complete software engineering model of the system through UML-based requirements specification, structural, functional, and behavioral diagrams. The third chapter documented the full implementation using Node.js, Express, MongoDB, React, Python, Flask, spaCy, and Sentence Transformers, and validated the platform through interface testing across all three user roles.

The results confirm that the proposed system successfully automates skills extraction from unstructured resumes and computes reliable semantic matching scores between candidate profiles and job descriptions, providing recruiters with an objective and efficient screening tool.

As future perspectives, the platform could be enhanced with supervised matching models trained on labeled recruitment data, multilingual support for Arabic and French to better serve the Algerian job market, and additional features such as interview scheduling and analytics dashboards.

Overall, this project demonstrates that modern AI techniques can be integrated into a practical and deployable web platform that brings real value to the recruitment process, while laying a solid foundation for further research and development in intelligent human resource systems.

REFERENCES

[1] Book — M. Armstrong, Armstrong's Handbook of Human Resource Management Practice, 13th ed. London: Kogan Page, 2014.

[2] Book — G. Dessler, Human Resource Management, 13th ed. Upper Saddle River, NJ: Pearson, 2013.

[3] Book — S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 4th ed. Hoboken, NJ: Pearson, 2020.

[4] Book — I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA: MIT Press, 2016.

[5] Book — C. M. Bishop, Pattern Recognition and Machine Learning. New York: Springer, 2006.

[6] Journal Article — G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," Information Processing & Management, vol. 24, no. 5, pp. 513–523, 1988.

[7] Journal Article — D. Nadeau and S. Sekine, "A survey of named entity recognition and classification," Lingvisticae Investigationes, vol. 30, no. 1, pp. 3–26, 2007.

[8] Journal Article — J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proc. NAACL-HLT, Minneapolis, MN, 2019, pp. 4171–4186.

[9] Conference Paper — N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using siamese BERT-networks," in Proc. EMNLP-IJCNLP, Hong Kong, 2019, pp. 3982–3992.

[10] Conference Paper
T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in Advances in Neural Information Processing Systems, vol. 26, 2013, pp. 3111–3119.

[11] Conference Paper
A. Vaswani et al., "Attention is all you need," in Advances in Neural Information Processing Systems, vol. 30, 2017, pp. 5998–6008.

[12] Journal Article
F. Pedregosa et al., "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[13] Webpage
M. Honnibal and I. Montani, "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing," 2017. [Online]. Available: https://spacy.io

[14] Conference Paper
T. Wolf et al., "Transformers: State-of-the-art natural language processing," in Proc. EMNLP: System Demonstrations, 2020, pp. 38–45.

[15] Conference Paper
D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in Proc. ICLR, San Diego, CA, 2015.

[16] Webpage
Node.js Foundation, "Node.js," 2009. [Online]. Available: https://nodejs.org

[17] Webpage
T. J. Holowaychuk, "Express.js: Fast, unopinionated, minimalist web framework for Node.js," 2010. [Online]. Available: https://expressjs.com

[18] Webpage
MongoDB, Inc., "MongoDB manual," 2009. [Online]. Available: https://www.mongodb.com/docs

[19] Webpage
Meta Platforms, Inc., "React: A JavaScript library for building user interfaces," 2013. [Online]. Available: https://react.dev

[20] Webpage E. You, “Vite: Next generation frontend tooling,” 2020. [Online]. Available: https://vitejs.dev

[21] Webpage A. Wathan, “Tailwind CSS: A utility-first CSS framework,” 2017. [Online]. Available: https://tailwindcss.com

[22] Webpage M. Zhu, “Axios: Promise based HTTP client for the browser and Node.js,” 2014. [Online]. Available: https://axios-http.com

[23] Webpage A. Ronacher, “Flask: A lightweight WSGI web application framework,” 2010. [Online]. Available: https://flask.palletsprojects.com

[24] Webpage Python Software Foundation, “Python language reference,” version 3.x. [Online]. Available: https://www.python.org

[25] Webpage Git Project, “Git: Distributed version control system,” 2005. [Online]. Available: https://git-scm.com

[26] Webpage GitHub, Inc., “GitHub: Where the world builds software,” 2008. [Online]. Available: https://github.com

[27] Webpage Postman, Inc., “Postman API platform,” 2012. [Online]. Available: https://www.postman.com

[28] Webpage LinkedIn Corporation, “LinkedIn: Professional networking platform,” 2003. [Online]. Available: https://www.linkedin.com

[29] Webpage Indeed, Inc., “Indeed: Job search platform,” 2004. [Online]. Available: https://www.indeed.com

[30] Webpage — Emploitic, “Emploitic: Algerian recruitment platform,” [Online]. Available: https://www.emploitic.com

[31] Webpage — EmploiPartner, “EmploiPartner: Algerian employment portal,” [Online]. Available: https://www.emploipartner.com

[32] Webpage — Option Carrière, “Option Carrière: Algerian job search platform,” [Online]. Available: https://www.optioncarriere.com

[33] Webpage — Object Management Group, Unified Modeling Language Specification, version 2.5.1. Needham, MA: OMG, 2017.

[34] Image — T. Madhushan, "Beyond the big three: The expanded landscape of machine learning paradigms," *Medium*, Apr. 21, 2026. [Online]. Available: https://medium.com/@th4rinonline/beyond-the-big-three-the-expanded-landscape-of-machine-learning-paradigms-7dec23f8efe9.

[35] Image — Wikipedia, "Neural network (machine learning)," *Wikipedia, The Free Encyclopedia*, [Online]. Available: https://en.wikipedia.org/wiki/Neural_network_(machine_learning).