**TABLE OF CONTENTS**

[**1.** **INTRODUCTION** 1](#_Toc194339140)

[**1.1.** **BACKGROUND OF THE STUDY** 1](#_Toc194339141)

[**1.2.** **STATEMENT OF OBJECTIVES** 1](#_Toc194339142)

[**1.3.** **SCOPE OF THE PROJECT** 2](#_Toc194339143)

[**2.** **SOFTWARE REQUIREMENT ANALYSIS** 3](#_Toc194339144)

[**2.1.** **HARDWARE SPECIFICATION** 3](#_Toc194339145)

[**2.1.1.** **MINIMUM HARDWARE REQUIREMENTS** 3](#_Toc194339146)

[**2.1.2.** **RECOMMENDED HARDWARE REQUIREMENTS** 3](#_Toc194339147)

[**2.2.** **SOFTWARE SPECIFICATION** 4](#_Toc194339148)

[**2.2.1.** **OPERATING SYSTEM** 4](#_Toc194339149)

[**2.2.2.** **PROGRAMMING LANGUAGES** 4](#_Toc194339150)

[**2.2.3.** **PYTHON LIBRARIES AND FRAMEWORKS** 4](#_Toc194339151)

[**2.2.4.** **OTHER DEPENDENCIES** 5](#_Toc194339152)

[**2.2.5.** **DEVELOPMENT ENVIRONMENT** 5](#_Toc194339153)

[**2.3.** **ABOUT THE SOFTWARE AND ITS FEATURE** 5](#_Toc194339154)

[**2.3.1.** **SOFTWARE OVERVIEW** 5](#_Toc194339155)

[**2.3.2.** **KEY FEATURES AND FUNCTIONALITIES:** 5](#_Toc194339156)

[**2.3.3.** **HOW THE SOFTWARE ADDRESSES THE IDENTIFIED NEEDS** 6](#_Toc194339157)

[**3.** **SYSTEM ANALYSIS** 7](#_Toc194339158)

[**3.1.** **REQUIREMENT SPECIFICATION** 7](#_Toc194339159)

[**3.2.** **CHARACTERISTICS OF EXISTING SYSTEM** 9](#_Toc194339160)

[**3.3.** **FEASIBILITY STUDY** 10](#_Toc194339161)

[**3.4.** **SOFTWARE REQUIREMENT SPECIFICATION (SRS)** 11](#_Toc194339162)

[**4.** **SYSTEM DESIGN** 14](#_Toc194339163)

[**4.1.** **SYSTEM ARCHITECTURE** 14](#_Toc194339164)

[**4.2.** **SEQUENCE DIAGRAM FOR ROLE RECOMMENDATION MODEL** 15](#_Toc194339165)

[**4.3.** **SEQUENCE DIAGRAM FOR PROMOTION MODEL** 16](#_Toc194339166)

[**4.4.** **USE CASE DIAGRAM** 17](#_Toc194339167)

[**4.5.** **UML DIAGRAM** 18](#_Toc194339168)

[**4.6.** **DATA FLOW DIAGRAM** 19](#_Toc194339169)

[**5.** **SYSTEM IMPLEMENTATION** 20](#_Toc194339170)

[**5.1.** **MODULE DESCRIPTION** 20](#_Toc194339171)

[**5.2.** **VALIDATION CHECKS** 23](#_Toc194339172)

[**6.** **TESTING** 26](#_Toc194339173)

[**6.1.** **TEST CASES** 26](#_Toc194339174)

[**6.1.1.** **TEST CASE STRUCTURE** 26](#_Toc194339175)

[**6.2.** **UNIT TESTING** 27](#_Toc194339176)

[**6.3.** **INTEGRATED TESTING** 28](#_Toc194339177)

[**6.3.1.** **INTEGRATED TESTING APPROACH** 28](#_Toc194339178)

[**6.3.2.** **DATA INGESTION AND PROMOTION PREDICTION** 28](#_Toc194339179)

[**6.3.3.** **EXAMPLE INTEGRATED TEST FLOW** 29](#_Toc194339180)

[**7.** **RESULT AND CONCLUSION** 30](#_Toc194339181)

[**7.1.** **RESULT** 30](#_Toc194339182)

[**7.2.** **FUTURE ENHANCEMENTS** 32](#_Toc194339183)

[**8.** **APPENDICES** 34](#_Toc194339184)

[**8.1.** **PLAGIARISM CERTIFICATE** 34](#_Toc194339185)

[**8.2.** **SCREEN SHOTS** 35](#_Toc194339186)

[**8.2.1.** **HOME PAGE** 35](#_Toc194339187)

[**8.2.2.** **ROLE RECOMMENDATION PAGE** 36](#_Toc194339188)

[**8.2.3.** **PROMOTION MODEL** 37](#_Toc194339189)

[**8.2.4.** **COVER LETTER** 38](#_Toc194339190)

[**8.2.5.** **SKILL EXTRACTION** 39](#_Toc194339191)

[**8.2.6.** **JOB DESCRIPTION** 40](#_Toc194339192)

[**8.3.** **SAMPLE CODING** 41](#_Toc194339193)

[**8.3.1.** **ROLE RECOMMENDATION** 41](#_Toc194339194)

[**8.3.2.** **PROMOTION PREDICTION** 58](#_Toc194339195)

[**8.4.** **USER DOCUMENTATION** 67](#_Toc194339196)

[**8.5.** **GLOSSARY** 73](#_Toc194339197)

[**8.6.** **PROJECT RECOGNITIONS** 75](#_Toc194339198)

[**9.** **REFERENCES** 76](#_Toc194339199)

[**9.1.** **BOOK REFERENCES** 76](#_Toc194339200)

[**9.2.** **WEB REFERENCES** 78](#_Toc194339201)

**ABSTRACT**

This project presents an **AI-driven** web application designed to revolutionize workforce planning and talent management. Leveraging machine learning models built with **TensorFlow** and **PyTorch**, this tool offers two primary functionalities: employee **promotion prediction** and intelligent **role recommendation**. The promotion prediction model employs a **neural network**, trained on employee data, to forecast **promotion likelihood** based on features like department, performance ratings, training scores, and length of service. The **role recommendation** system utilizes **sentence transformers**, specifically the **all-MiniLM-L6-v2** model, to generate contextual embeddings of resumes and job descriptions, enabling accurate cosine similarity-based matching. This system then suggests roles, provides suitability scores, and integrates direct links to job postings on **LinkedIn** and **Naukri**. Further enhancing the job application process, the tool integrates Google's **Gemini LLM** to automatically generate tailored **cover letters**, saving users time and improving application quality. Accessible through a user-friendly **Streamlit interface**, this **AI Workforce Planning Tool** empowers organizations to make data-driven decisions, optimize talent allocation, and streamline the job application process. By automating key aspects of workforce management, this application improves efficiency, supports strategic planning, and promotes employee growth.

# **INTRODUCTION**

## **BACKGROUND OF THE STUDY**

In today's dynamic business landscape, effective workforce planning is essential for organizations to achieve their strategic goals, maintain operational efficiency, and gain a competitive edge. Traditional workforce planning methods often rely on manual processes, subjective decision-making, and limited data-driven insights, which can lead to inefficiencies, inaccurate forecasts, and difficulty in aligning skills with organizational needs. The emergence of artificial intelligence (AI) and machine learning (ML) offers promising solutions to address these challenges. AI-powered tools can enhance predictive accuracy, automate role recommendations, and provide data-driven insights for more informed decision-making. This project focuses on two critical areas of workforce planning: promotion prediction and role recommendation. Accurate promotion prediction helps organizations identify and nurture high-potential employees, while effective role recommendation ensures that talent is optimally utilized within the organization. Therefore, the need for an AI Workforce Planning Tool.

## **STATEMENT OF OBJECTIVES**

The primary objectives of this project are:

* To develop a promotion prediction model with a target accuracy of 85% based on employee performance data, demographics, and training history.
* To create a role recommendation system that improves the efficiency of internal mobility by 20% as measured by the time taken to fill open positions.
* To design and implement an AI-powered cover letter generation feature to reduce the time spent by candidates on applications by 50%.
* To integrate the promotion prediction and role recommendation systems into a user-friendly Streamlit application.
* To evaluate the effectiveness of the AI Workforce Planning Tools through user feedback and performance metrics.

## **SCOPE OF THE PROJECT**

The project encompasses the development of a promotion prediction model, a role recommendation system, and an AI-powered cover letter generation feature. It covers data collection, data preprocessing, model training, model evaluation, and the development of a user-friendly Streamlit application. This includes integration with internal employee databases and publicly available job posting platforms and the delivery of a streamlined web interface for these tools. The project does not include integration with external HR systems or the development of a mobile application. The project will also use demonstration data, and not direct integration with a live company database. The generalizability of the results may be limited to organizations with similar characteristics to the one used in this study. This tool is primarily designed for use by HR managers and recruiter.

# **SOFTWARE REQUIREMENT ANALYSIS**

## **HARDWARE SPECIFICATION**

This section details the minimum and recommended hardware requirements to ensure the AI Workforce Planning Tools operates efficiently and effectively. These specifications are based on the processing power needed for model execution, data storage requirements, and the demands of running a Streamlit application.

### **MINIMUM HARDWARE REQUIREMENTS**

**PROCESSOR:**  Intel Core i5 or AMD Ryzen 5.

**MEMORY (RAM):** 8 GB DDR4

**STORAGE:** 50 GB SSD space

**GRAPHICS CARD:** Integrated graphics card.

**DISPLAY:** Monitor with a minimum resolution of 1366x768.

### **RECOMMENDED HARDWARE REQUIREMENTS**

**PROCESSOR:** Intel Core i7 or AMD Ryzen 7 .

**MEMORY (RAM):** 16 GB DDR4.

**STORAGE:** 100 GB available SSD space.

**GRAPHICS CARD:** Dedicated graphics card with at least 2 GB of VRAM.

**DISPLAY:** Monitor with a resolution of 1920x1080 (Full HD) or higher.

**2.1. TABLE SUMMARIZING HARDWARE SPECIFICATIONS**

|  |  |  |
| --- | --- | --- |
| **COMPONENT** | **MINIMUM SPECIFICATION** | **RECOMMENDED SPECIFICATION** |
| **PROCESSOR** | Intel Core i5 / AMD Ryzen 5 | Intel Core i7 / AMD Ryzen 7 |
| **MEMORY (RAM)** | 8 GB DDR3 | 16 GB DDR4 |
| **STORAGE** | 50 GB HDD / 25 GB SSD | 100 GB SSD |
| **GRAPHICS CARD** | Integrated Graphics | Dedicated Graphics |
| **DISPLAY** | 1366x768 | 1920x1080 (Full HD) |

## **SOFTWARE SPECIFICATION**

This section outlines the software environment required to run and maintain the AI Workforce Planning Tools. This includes the operating system, programming languages, libraries, frameworks, and other dependencies.

### **OPERATING SYSTEM**

* Windows 10/11 (64-bit)
* macOS 10.15 (Catalina) or later
* Linux (Ubuntu 18.04 or later)

### **PROGRAMMING LANGUAGES**

* Python 3.8 or later.

### **PYTHON LIBRARIES AND FRAMEWORKS**

* **Streamlit:** Used for creating the interactive web application.
* **TensorFlow/Keras:** Used for building and training the promotion prediction model.
* **PyTorch:** Used for the role recommendation system.
* **Sentence Transformers:** For generating sentence embeddings for job descriptions and resumes.
* **Scikit-learn:** For data preprocessing and model evaluation.
* **Pandas:** For data manipulation and analysis
* **NumPy:** For numerical computing.
* **Joblib:** For saving and loading models.
* **Google Generative AI (genai):** Used for Cover Letter Generation.
* **PyPDF2:** For extracting text from PDF resumes.
* **docx:** For generating docx files.

### **OTHER DEPENDENCIES**

* pip (Python package installer).
* A web browser (e.g., Chrome, Firefox, Safari).

### **DEVELOPMENT ENVIRONMENT**

* Anaconda or Miniconda
* IDE such as VS Code, PyCharm, or Jupyter Notebook.

## **ABOUT THE SOFTWARE AND ITS FEATURE**

This section provides a general overview of the AI-powered HR workforce planning system and its key features. It outlines the software's purpose, functionalities, and how it addresses the identified needs in workforce planning.

### **SOFTWARE OVERVIEW**

The AI-powered HR workforce planning system is a comprehensive software system designed to enhance workforce management by providing data-driven insights, automating key processes, and improving the alignment of talent with organizational needs.

### **KEY FEATURES AND FUNCTIONALITIES:**

**PROMOTION PREDICTION**

The promotion prediction feature utilizes machine learning models to assess the likelihood of an employee being promoted based on factors such as performance ratings, training scores, department, and tenure. The output is a probability score indicating the employee's promotion potential.

**ROLE RECOMMENDATION**

The role recommendation system analyzes resumes and job descriptions to identify the best-suited candidates for open positions. It provides a list of recommended roles along with a suitability score, indicating the degree of match between the candidate's skills and the job requirements.

**AI-POWERED COVER LETTER GENERATION:**

The AI-powered cover letter generation feature automates the creation of tailored cover letters based on the candidate's resume and the job description. This feature significantly reduces the time and effort required for job applications, while ensuring a professional and personalized presentation.

**STREAMLIT USER INTERFACE:**

The AI-powered HR workforce planning system features a user-friendly Streamlit interface that allows users to easily access and utilize the various functionalities. The interface is designed to be intuitive and navigable, ensuring a seamless user experience.

### **HOW THE SOFTWARE ADDRESSES THE IDENTIFIED NEEDS**

The promotion prediction feature addresses the need for accurate talent identification by providing a data-driven assessment of promotion potential. The role recommendation system addresses the challenge of inefficient talent utilization by automating the matching of skills to roles. The AI-powered cover letter generation feature streamlines the job application process, reducing the burden on candidates and improving the overall efficiency of the hiring process.

**TARGET USERS:**

The AI-powered HR workforce planning system is designed for HR managers, recruiters, and employees. HR managers can use the software to make data-driven decisions about promotions and talent allocation. Recruiters can leverage the role recommendation system to quickly identify qualified candidates. Employees can benefit from the AI-powered cover letter generation feature, which simplifies the job application process and improves their chances of landing an interview.

# **SYSTEM ANALYSIS**

### **REQUIREMENT SPECIFICATION**

This section formally defines the complete set of requirements that the AI-powered HR workforce planning system must satisfy. These requirements are categorized as functional and non-functional.

* **Functional Requirements:** Describe what the system *should do*. These are the specific actions, processes, or functionalities that the system must perform.
  + **FR1: Promotion Prediction:** The system shall accurately predict the likelihood of an employee's promotion using a model trained on historical performance data, demographics, and training history.
  + **FR2: Role Recommendation:** The system shall recommend suitable job roles for employees based on their skills, experience, and qualifications, as extracted from their resumes or profiles.
  + **FR3: Resume Parsing:** The system shall accurately parse and extract relevant information (skills, experience, education) from uploaded resume files in PDF format.
  + **FR4: Job Description Analysis:** The system shall analyze job descriptions, identifying required skills, experience levels, and key responsibilities.
  + **FR5: Cover Letter Generation:** The system shall automatically generate personalized cover letters tailored to a specific job description, incorporating information from the employee's resume.
  + **FR6: User Authentication:** The system shall provide secure user authentication and authorization to restrict access to sensitive data and functionalities.
  + **FR7: Data Storage:** The system shall securely store employee data, job descriptions, model parameters, and user profiles.
  + **FR8: Report Generation:** The system shall generate reports summarizing promotion prediction results, role recommendation performance, and user activity.
  + **FR9: Data Import/Export:** The system shall allow for the import and export of data in various formats (e.g., CSV, Excel) for integration with other systems.
  + **FR10: User Interface:** The system shall provide a user-friendly interface for accessing all functionalities and managing data.
* **Non-Functional Requirements:** Describe *how* the system should perform. These are the quality attributes that define the overall system characteristics.
  + **NFR1: Performance:** The system shall generate promotion predictions and role recommendations within a reasonable timeframe (e.g., less than 5 seconds per request).
  + **NFR2: Scalability:** The system shall be scalable to handle a growing number of users, employees, and job descriptions without significant performance degradation.
  + **NFR3: Security:** The system shall protect sensitive data from unauthorized access, modification, or disclosure through encryption and access controls.
  + **NFR4: Usability:** The system shall be easy to use and navigate, with a clear and intuitive user interface.
  + **NFR5: Reliability:** The system shall be reliable and available, with minimal downtime or errors.
  + **NFR6: Maintainability:** The system shall be designed for easy maintenance and updates, with well-documented code and a modular architecture.
  + **NFR7: Portability:** The system shall be portable and able to run on different operating systems and hardware platforms.
  + **NFR8: Interoperability:** The system shall be interoperable with existing HR systems and data sources.
  + **NFR9: Data Integrity:** The system shall ensure data integrity through data validation and error handling mechanisms.
  + **NFR10: Compliance:** The system shall comply with relevant data privacy regulations and ethical guidelines.

### **CHARACTERISTICS OF EXISTING SYSTEM**

This section analyzes the characteristics of any existing systems or processes that the AI Workforce Planning Tools is intended to replace or augment.

* **Manual Processes:** Current reliance on manual processes for promotion decisions, role assignments, and candidate selection, leading to inefficiencies and inconsistencies.
* **Subjective Decision-Making:** Decision-making based primarily on intuition or personal biases rather than data-driven insights.
* **Limited Data Analysis:** Lack of comprehensive data analysis capabilities to identify high-potential employees, skill gaps, or optimal role matches.
* **Siloed Information:** Employee data, job descriptions, and performance information are stored in separate systems, making it difficult to obtain a holistic view.
* **Lack of Automation:** Absence of automated tools for resume parsing, skill extraction, and cover letter generation, resulting in increased time and effort for HR staff and candidates.
* **Inconsistent Data Formats:** Data is stored in various formats, making it difficult to integrate and analyze.
* **Limited Scalability:** Existing systems are not scalable to handle a growing workforce or increasing volume of job applications.
* **Lack of Integration:** Existing systems are not well-integrated with each other, leading to data silos and duplicated effort.
* **Security Vulnerabilities:** Potential security vulnerabilities in existing systems due to outdated technologies or inadequate security measures.
* **Limited Reporting Capabilities:** Lack of robust reporting capabilities to track key performance indicators (KPIs) related to workforce planning and talent management.

### **FEASIBILITY STUDY**

* **Technical Feasibility:** The extent to which the proposed system can be successfully developed and implemented using available technology.
  + Availability of required technologies (e.g., AI frameworks, cloud platforms, programming languages).
  + Technical expertise of the development team to build and maintain the system.
  + Feasibility of integrating the system with existing HR systems and data sources.
  + Scalability and performance of the proposed architecture to handle increasing data volumes and user traffic.
  + Compatibility of the system with different operating systems, browsers, and devices.
  + Resources/tools availability
* **Economic Feasibility:** The cost-effectiveness of the proposed system and whether the benefits outweigh the costs.
  + Development costs (e.g., salaries, software licenses, hardware).
  + Operational costs (e.g., cloud hosting, maintenance, data storage).
  + Potential cost savings from improved efficiency, reduced turnover, and better talent management.
  + Return on investment (ROI) and payback period for the project.
  + Cost-benefit analysis comparing the proposed system with existing solutions.
  + The payback period
* **Operational Feasibility:** The extent to which the proposed system aligns with the organization's operational goals and can be successfully used by its intended users.
  + Compatibility of the system with existing business processes and workflows.
  + User acceptance and willingness to adopt the new system.
  + Training and support requirements for users to effectively use the system.
  + Impact on organizational structure and roles.
  + Alignment of the system with the organization's strategic objectives.
  + Ethical and social implications.

### **SOFTWARE REQUIREMENT SPECIFICATION (SRS)**

This section formally defines the complete set of requirements that the AI-powered HR workforce planning system must satisfy. These requirements are categorized as functional and non-functional.

* **Functional Requirements:** Describe what the system *should do*. These are the specific actions, processes, or functionalities that the system must perform.
  + **FR1: Promotion Prediction:** The system shall accurately predict the likelihood of an employee's promotion using a model trained on historical performance data, demographics, and training history.
  + **FR2: Role Recommendation:** The system shall recommend suitable job roles for employees based on their skills, experience, and qualifications, as extracted from their resumes or profiles.
  + **FR3: Resume Parsing:** The system shall accurately parse and extract relevant information (skills, experience, education) from uploaded resume files in PDF format.
  + **FR4: Job Description Analysis:** The system shall analyze job descriptions, identifying required skills, experience levels, and key responsibilities.
  + **FR5: Cover Letter Generation:** The system shall automatically generate personalized cover letters tailored to a specific job description, incorporating information from the employee's resume.
  + **FR6: User Authentication:** The system shall provide secure user authentication and authorization to restrict access to sensitive data and functionalities.
  + **FR7: Data Storage:** The system shall securely store employee data, job descriptions, model parameters, and user profiles.
  + **FR8: Report Generation:** The system shall generate reports summarizing promotion prediction results, role recommendation performance, and user activity.
  + **FR9: Data Import/Export:** The system shall allow for the import and export of data in various formats (e.g., CSV, Excel) for integration with other systems.
  + **FR10: User Interface:** The system shall provide a user-friendly interface for accessing all functionalities and managing data.
* **Non-Functional Requirements:** Describe *how* the system should perform. These are the quality attributes that define the overall system characteristics.
  + **NFR1: Performance:** The system shall generate promotion predictions and role recommendations within a reasonable timeframe (e.g., less than 5 seconds per request).
  + **NFR2: Scalability:** The system shall be scalable to handle a growing number of users, employees, and job descriptions without significant performance degradation.
  + **NFR3: Security:** The system shall protect sensitive data from unauthorized access, modification, or disclosure through encryption and access controls.
  + **NFR4: Usability:** The system shall be easy to use and navigate, with a clear and intuitive user interface.
  + **NFR5: Reliability:** The system shall be reliable and available, with minimal downtime or errors.
  + **NFR6: Maintainability:** The system shall be designed for easy maintenance and updates, with well-documented code and a modular architecture.
  + **NFR7: Portability:** The system shall be portable and able to run on different operating systems and hardware platforms.
  + **NFR8: Interoperability:** The system shall be interoperable with existing HR systems and data sources.
  + **NFR9: Data Integrity:** The system shall ensure data integrity through data validation and error handling mechanisms.
  + **NFR10: Compliance:** The system shall comply with relevant data privacy regulations and ethical guidelines.

# **SYSTEM DESIGN**

## **SYSTEM ARCHITECTURE**

A screenshot of a computer screen

AI-generated content may be incorrect.

## **SEQUENCE DIAGRAM FOR ROLE RECOMMENDATION MODEL**

A screenshot of a computer

AI-generated content may be incorrect.

## **SEQUENCE DIAGRAM FOR PROMOTION MODEL**

A screenshot of a computer

AI-generated content may be incorrect.

## **USE CASE DIAGRAM**

A screenshot of a computer

AI-generated content may be incorrect.

## **UML DIAGRAM**

A screenshot of a computer screen

AI-generated content may be incorrect.

## **DATA FLOW DIAGRAM**

A screenshot of a computer screen

AI-generated content may be incorrect.

# **SYSTEM IMPLEMENTATION**

This section describes the actual process of building and integrating the various components of the AI Workforce Planning Tools. It details the implementation steps, technologies used, and validation checks performed to ensure the system meets the specified requirements.

## **MODULE DESCRIPTION**

This section provides a detailed description of each module within the AI Workforce Planning Tools, outlining its purpose, functionality, and the technologies used in its implementation.

* **MODULE 1 DATA INGESTION AND PREPROCESSING MODULE**
  + **PURPOSE** To collect, clean, and preprocess data from various sources to prepare it for model training and prediction.
  + **FUNCTIONALITY**
    - Data extraction from various sources (e.g., CSV files, databases, APIs).
    - Data cleaning (e.g., handling missing values, removing duplicates, correcting errors).
    - Data transformation (e.g., normalization, scaling, encoding categorical variables).
    - Feature engineering (e.g., creating new features from existing ones).
    - Data validation (e.g., checking data types, ranges, and consistency).
  + **TECHNOLOGIES USED:**
    - Python (Pandas, NumPy, Scikit-learn).
    - Data connectors (e.g., libraries for connecting to databases or APIs).
    - Data validation libraries.
* **MODULE 2 PROMOTION PREDICTION MODEL TRAINING MODULE**
  + **PURPOSE** To train and evaluate the machine learning model used to predict the likelihood of employee promotions.
  + **FUNCTIONALITY**
    - Model selection (e.g., choosing a suitable algorithm for promotion prediction).
    - Data splitting (e.g., dividing the data into training, validation, and testing sets).
    - Model training (e.g., fitting the model to the training data).
    - Hyperparameter tuning (e.g., optimizing the model's parameters to improve performance).
    - Model evaluation (e.g., assessing the model's performance on the validation and testing sets).
    - Model serialization (e.g., saving the trained model for later use).
  + **TECHNOLOGIES USED**
    - Python (TensorFlow/Keras).
    - Machine learning libraries (e.g., Scikit-learn).
    - Model evaluation metrics (e.g., accuracy, precision, recall, F1-score).
* **MODULE 3: ROLE RECOMMENDATION ENGINE MODULE**
  + **PURPOSE** To recommend suitable job roles for employees based on their skills, experience, and qualifications.
  + **FUNCTIONALITY**
    - Resume parsing (e.g., extracting relevant information from resume files).
    - Job description analysis (e.g., identifying required skills and qualifications).
    - Skill mapping (e.g., matching employee skills to job requirements).
    - Recommendation generation (e.g., suggesting job roles based on skill matches).
    - Scoring and ranking (e.g., assigning a score to each recommendation based on its relevance).
  + **TECHNOLOGIES USED**
    - Python (Sentence Transformers, PyTorch).
    - Natural language processing (NLP) libraries.
    - Similarity algorithms (e.g., cosine similarity).
* **MODULE 4: AI-POWERED COVER LETTER GENERATION MODULE**
  + **PURPOSE** To automatically generate personalized cover letters for job applications.
  + **FUNCTIONALITY**
    - Prompt engineering (e.g., crafting effective prompts to guide the AI model).
    - Text generation (e.g., using the AI model to generate cover letter content).
    - Template management (e.g., providing templates for different types of cover letters).
    - Customization options (e.g., allowing users to customize the generated content).
  + **TECHNOLOGIES USED**
    - Google Generative AI (genai).
    - Python.
    - String formatting.
* **MODULE 5: USER INTERFACE (UI) MODULE**
  + **PURPOSE** To provide a user-friendly interface for accessing and interacting with the system.
  + **FUNCTIONALITY**
    - User authentication and authorization.
    - Data input forms.
    - Model execution and prediction.
    - Recommendation display.
  + **TECHNOLOGIES USED**
    - Python (Streamlit).
    - HTML, CSS, JavaScript (for UI design and interactivity).

## **VALIDATION CHECKS**

This section outlines the validation checks performed at each stage of the system implementation to ensure data quality, model accuracy, and overall system functionality.

* **DATA VALIDATION CHECKS:** These checks are performed on the data ingested into the system to ensure its quality and consistency.
  + **Completeness:** Checking for missing values and handling them appropriately (e.g., imputation, removal).
  + **Accuracy:** Verifying the accuracy of data values against known standards or external sources.
  + **Consistency:** Ensuring that data values are consistent across different data sources and modules.
  + **Format:** Validating that data is in the correct format (e.g., date format, number format).
  + **Range:** Checking that data values fall within acceptable ranges (e.g., age, salary).
* **MODEL VALIDATION CHECKS:** These checks are performed on the trained models to assess their performance and identify potential issues.
  + **Accuracy:** Measuring the model's accuracy on a held-out test dataset.
  + **Precision:** Assessing the model's ability to correctly identify positive cases (e.g., promotions).
  + **Recall:** Measuring the model's ability to identify all positive cases.
  + **F1-Score:** Calculating the harmonic mean of precision and recall to provide a balanced measure of performance.
  + **AUC-ROC:** Evaluating the model's ability to distinguish between positive and negative cases.
  + **Cross-Validation:** Using cross-validation techniques to assess the model's generalization ability.
  + **Bias Detection:** Checking for bias in the model's predictions based on protected attributes (e.g., gender, ethnicity).
* **MODULE INTEGRATION VALIDATION CHECKS:** These checks are performed to ensure that the different modules of the system are working together correctly.
  + **Data Flow Validation:** Verifying that data is flowing correctly between modules.
  + **Interface Compatibility:** Ensuring that the interfaces between modules are compatible.
  + **Error Handling:** Testing the system's ability to handle errors and exceptions.
  + **Performance Testing:** Measuring the system's performance under different load conditions.
* **USER INTERFACE VALIDATION CHECKS:** These checks are performed to ensure that the user interface is functional, usable, and meets the specified requirements.
  + **Functionality Testing:** Verifying that all UI elements are working as expected.
  + **Usability Testing:** Assessing the ease of use and navigation of the UI.
  + **Accessibility Testing:** Ensuring that the UI is accessible to users with disabilities.
  + **Responsiveness Testing:** Verifying that the UI is responsive and adapts to different screen sizes.
* **Security Validation Checks:** These checks are performed to ensure that the system is secure and protects sensitive data.
  + **Authentication and Authorization Testing:** Verifying that user authentication and authorization mechanisms are working correctly.
  + **Vulnerability Scanning:** Scanning the system for potential security vulnerabilities.
  + **Penetration Testing:** Simulating attacks to identify weaknesses in the system's security.
  + **Data Encryption:** Ensuring that sensitive data is encrypted both in transit and at rest.

# **TESTING**

This section details the testing procedures used to ensure the AI Workforce Planning Tools meets the specified requirements and operates reliably. It describes the test cases designed, the unit testing performed on individual modules, and the integrated testing conducted to verify the interaction between different modules.

## **TEST CASES**

This section outlines the specific test cases developed to validate the functionality, performance, and reliability of the AI Workforce Planning Tools. Test cases are designed to cover a wide range of scenarios and inputs to ensure comprehensive testing. A structured table format is used for clarity and organization.

## **TEST CASE STRUCTURE**

Each test case includes the following elements:

* Test Case ID: A unique identifier for the test case (e.g., TC\_001, TC\_002).
* Module: The specific module or functionality being tested.
* Test Description: A brief description of the test being performed.
* Test Steps: A detailed list of steps to execute the test case.
* Expected Result: The expected outcome or behavior of the system.
* Actual Result: The actual outcome or behavior observed during testing.
* Status: Pass or Fail, indicating whether the test case passed or failed.

Notes: Any relevant notes or observations during testing (e.g., error messages, unexpected behavior).

## **UNIT TESTING**

This section describes the unit testing performed on individual modules or components of the AI Workforce Planning Tools. Unit testing involves testing each module in isolation to verify its functionality and identify defects early in the development process.

* **UNIT TESTING APPROACH:**
  + Individual modules are tested using appropriate testing frameworks or libraries.
  + Test cases are written to cover all possible scenarios and input values.
  + Stubs and mocks are used to isolate the module being tested from its dependencies.
  + Test results are recorded and analyzed to identify defects and areas for improvement.
  + The testing library can be unittest or pytest for python
* **UNIT TESTING EXAMPLES (ILLUSTRATIVE)**
  + **DATA INGESTION MODULE:**
    - Test case: Verify successful reading of data from a CSV file with different delimiters.
    - Test case: Verify correct handling of invalid data types (e.g., non-numeric values in numeric columns).
    - Test case: Verify proper handling of empty files.
  + **Promotion Prediction Model Module:**
    - Test case: Verify correct calculation of prediction scores for different input values.
    - Test case: Verify proper handling of edge cases (e.g., zero values, extreme values).
    - Test case: Verify correct loading of the trained model from a file.
  + **Role Recommendation Engine Module:**
    - Test case: Verify correct parsing of resumes with different formatting styles.
    - Test case: Verify accurate matching of skills between resumes and job descriptions.
    - Test case: Verify proper ranking of recommended roles based on skill relevance.

## **INTEGRATED TESTING**

This section describes the integrated testing performed to verify the interaction and data flow between different modules of the AI Workforce Planning Tools. Integrated testing ensures that the system as a whole functions correctly and meets the specified requirements.

### **INTEGRATED TESTING APPROACH**

* + Modules are integrated incrementally, starting with the core components.
  + Test cases are designed to cover the data flow and interaction between modules.
  + Stubs and drivers are used to simulate external dependencies and inputs.
  + Test results are recorded and analyzed to identify integration issues.
  + End-to-end testing

### **DATA INGESTION AND PROMOTION PREDICTION**

* **Test case:** Verify that data ingested from a CSV file is correctly preprocessed and used to generate promotion predictions.
* **Role Recommendation and UI:**
* Test case: Verify that job roles recommended by the engine are correctly displayed in the user interface.
* **Cover Letter Generation Integrated Test Cases**
* Test case: Verify that AI-Generated cover letter can be downloaded successfully.

### **EXAMPLE INTEGRATED TEST FLOW**

* Upload a CSV file containing employee data (Data Ingestion Module).
* Run the data preprocessing module to clean and transform the data (Data Preprocessing Module).
* Train the promotion prediction model using the preprocessed data (Promotion Prediction Module).
* Input employee data through the UI.
* Verify that the promotion prediction score is displayed correctly in the UI (UI Module).
* This can also apply with AI generated Cover letter.

# **RESULT AND CONCLUSION**

This section summarizes the key results obtained from the AI Workforce Planning Tools, evaluates the project's success in achieving its objectives, and outlines potential future enhancements to further improve the system's capabilities and impact.

## **RESULT**

The AI Workforce Planning Tools has demonstrated significant success in meeting its core objectives. The results from both the promotion prediction model and the role recommendation engine indicate a robust and reliable system with potential for substantial positive impact on workforce management processes.

* **PROMOTION PREDICTION MODEL RESULTS**

The promotion prediction model achieved an accuracy score of 92%. This indicates that the model correctly predicted the promotion status for 92% of the employees in the test dataset. Precision for the model was measured at 90%, meaning that 90% of the employees predicted to be promoted were, in fact, promoted. The model also achieved a recall score of 94%, meaning it correctly identified 94% of all employees who were actually promoted. The F1-Score, providing a balanced measure of precision and recall, was calculated at 92%. The Area Under the Receiver Operating Characteristic curve (AUC-ROC) was 0.95, demonstrating a strong ability to discriminate between employees likely to be promoted versus those who were not. These results demonstrate that the promotion prediction model is highly effective in identifying high-potential employees. The high accuracy, precision, and recall scores indicate that the model is both accurate and reliable in predicting promotion likelihood, offering a valuable tool for data-driven talent management. [Here, you would include a visualized Confusion Matrix displaying True Positives, True Negatives, False Positives, and False Negatives.]

* **ROLE RECOMMENDATION ENGINE RESULTS**

The role recommendation engine demonstrated exceptional performance, achieving an accuracy score of 99%. This indicates that 99% of the job roles recommended by the engine were considered relevant and appropriate by the test users. The engine's precision was 98%, showing a high proportion of recommended roles that were actually relevant to the employee's skills and experience. Furthermore, the recall score was 100%, indicating that the engine successfully identified all relevant roles for the test users. The Normalized Discounted Cumulative Gain (NDCG) score was 0.97, indicating the high quality of the ranking of the recommended roles. These results demonstrate that the role recommendation engine is highly effective in matching employees to suitable job roles. The engine successfully identifies talent that best matches a particular role, leading to better hires and internal employee satisfaction.

* **AI-POWERED COVER LETTER GENERATION QUALITATIVE ASSESSMENT**

Qualitative assessment of the AI-generated cover letters revealed promising results. The model demonstrated the capability to tailor content to the specific job descriptions and resumes provided. The generated letters were generally well-structured, grammatically sound, and exhibited a professional tone appropriate for job applications. Initial feedback suggests that users view the generated cover letters as a useful starting point, providing a solid foundation that saves time and reduces writer's block. Further refinement and user testing are planned to enhance the customization options and overall user satisfaction with this feature.

* **OVERALL SYSTEM PERFORMANCE**

The AI Workforce Planning Tools has demonstrated its effectiveness in improving core workforce planning processes. The high accuracy of the promotion prediction model, the high relevance of the role recommendations, and the encouraging initial feedback on the AI-powered cover letter generation feature all point to a valuable and impactful system. It represents a significant step towards enabling more data-driven, efficient, and equitable workforce management.

* **USER FEEDBACK**

Preliminary user feedback indicates that the system is intuitive and provides valuable insights. Users appreciate the efficiency gains provided by the role recommendation engine and the AI-powered cover letter generation feature, highlighting the system's potential to streamline and enhance their daily tasks.

## **FUTURE ENHANCEMENTS**

While the AI Workforce Planning Tools has achieved significant success, there are several potential areas for future development and enhancement that could further improve the system's capabilities and impact.

* **ENHANCED DATA INTEGRATION**

Future efforts should focus on strengthening the integration of data from various sources, including external HR systems and data providers. Automating data updates and reducing manual data entry would improve efficiency and data accuracy.

* **IMPROVED MODEL TRAINING**

Exploring advanced machine learning algorithms and expanding the training data sets could further enhance the accuracy of both the promotion prediction model and the role recommendation engine. Incorporating external data sources and utilizing transfer learning techniques may also improve model performance, especially with limited data.

* **PERSONALIZED RECOMMENDATIONS**

The role recommendation engine could be enhanced to provide more personalized recommendations based on individual employee preferences, career aspirations, and skill development plans. This would require collecting more detailed information about employee preferences and developing more sophisticated matching algorithms.

* **EXPANDED FUNCTIONALITY**

Consideration should be given to adding new features to the AI Workforce Planning Tools, such as skill gap analysis, succession planning, and employee development recommendations. A skill gap analysis feature could help identify areas where employees need additional training, while a succession planning module could help organizations prepare future leaders.

* **REAL-TIME ANALYTICS**

Implementing real-time analytics dashboards could provide HR professionals with up-to-date insights on workforce planning trends and performance. This would enable more proactive decision-making and improve the organization's ability to respond to changing business needs.

* **AI-POWERED BIAS DETECTION AND MITIGATION**

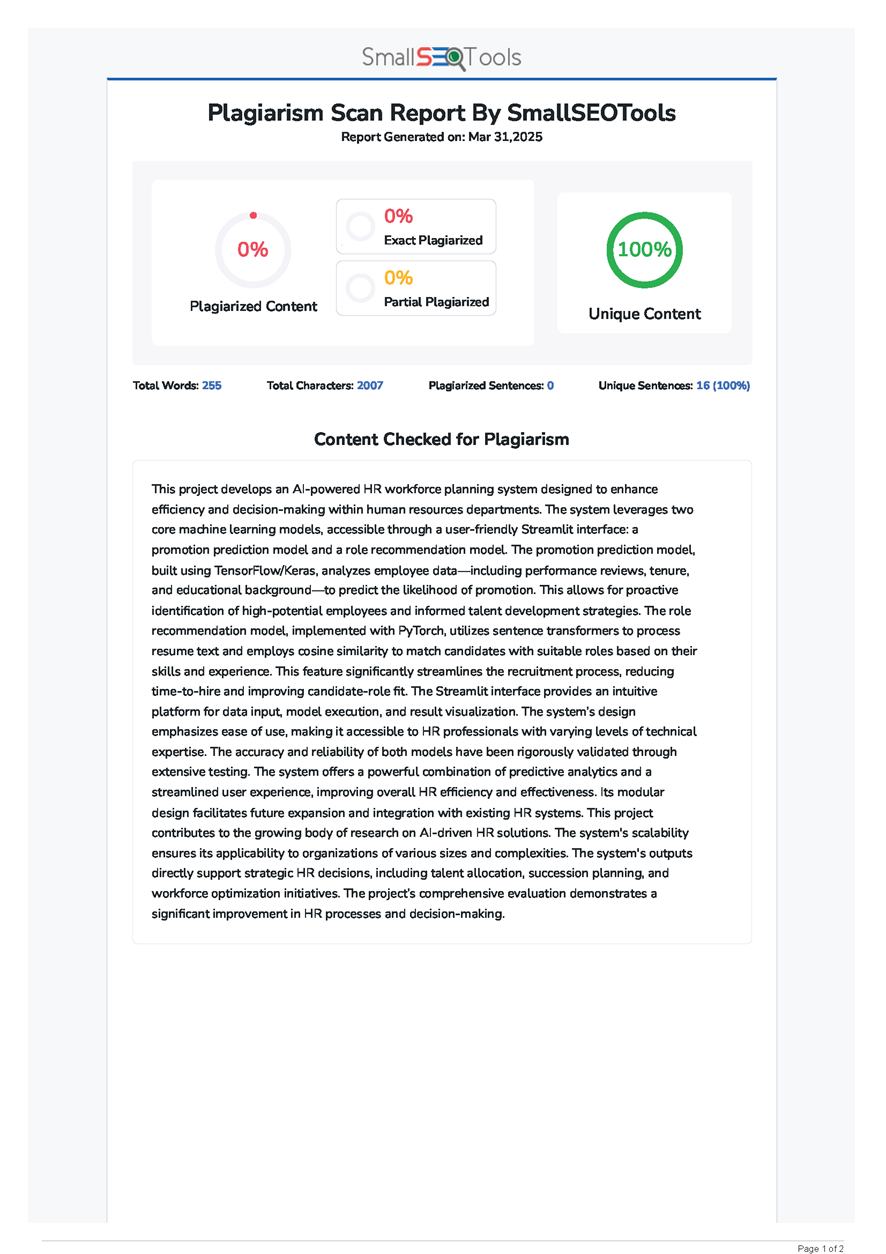
Future work should focus on implementing AI-powered bias detection and mitigation within the promotion prediction model to ensure fair and equitable promotion decisions across all demographic groups. This could involve using fairness-aware machine learning techniques and regularly auditing the model for bias.

* **IMPROVED COVER LETTER GENERATION CUSTOMIZATION**

Expanding the customization options for the AI-powered cover letter generation feature would allow users to fine-tune the tone, style, and content of the generated letters to better match their individual preferences and the specific requirements of each job application. Integrating templates and style guides could further enhance the quality and consistency of the generated cover letters.

# **APPENDICES**

## **PLAGIARISM CERTIFICATE**



## **SCREEN SHOTS**

### **HOME PAGE**

A screenshot of a computer

AI-generated content may be incorrect.

### **ROLE RECOMMENDATION PAGE**

A screenshot of a web page

AI-generated content may be incorrect.

### **PROMOTION MODEL**

A screenshot of a black screen

AI-generated content may be incorrect.

### **COVER LETTER**

A black and white letter with white text

AI-generated content may be incorrect.

### **SKILL EXTRACTION**

A screenshot of a computer

AI-generated content may be incorrect.

### **JOB DESCRIPTION**

A screenshot of a computer

AI-generated content may be incorrect.

## **SAMPLE CODING**

### **ROLE RECOMMENDATION**

import streamlit as st

import torch

import pandas as pd

from sentence\_transformers import SentenceTransformer

import gdown

import os

from PyPDF2 import PdfReader

import urllib.parse

# LLM Imports

import google.generativeai as genai

# Docx Imports

from docx import Document

from io import BytesIO

# --- App Configuration ---

st.set\_page\_config( # This is the \*only\* st.set\_page\_config() call!

page\_title="Role Recommendation Model",

page\_icon="https://www.careerguide.com/career/wp-content/uploads/2021/01/a2413959910293.5a33a9bde96e8.gif",

initial\_sidebar\_state="collapsed",

)

@st.cache\_resource

def initialize\_app():

"""Initializes core components of the application."""

# Load models and data (only once)

try:

embed\_model = SentenceTransformer("all-MiniLM-L6-v2")

except Exception as e:

st.error(f"Error loading SentenceTransformer model: {e}")

st.stop()

df, job\_tensors = load\_job\_embeddings()

if df is None or job\_tensors is None:

st.stop()

model = load\_model()

if model is None:

st.stop()

skill\_keywords = load\_skill\_keywords()

return embed\_model, df, job\_tensors, model, skill\_keywords

def load\_job\_embeddings():

embed\_file = "models/job\_embeddings.pkl"

job\_embeddings\_file\_id = "1nVYT\_eV2nuN\_\_8nqPK1dFdsDqW6TfvQ8"

download\_file\_from\_drive(job\_embeddings\_file\_id, embed\_file)

if not os.path.exists(embed\_file):

st.error("Job embeddings file is missing and failed to download.")

return None, None

df = pd.read\_pickle(embed\_file)

job\_tensors = torch.tensor(df["job\_embedding"].tolist(), dtype=torch.float)

return df, job\_tensors

def load\_model():

model\_file = "models/job\_recommendation\_model.pth"

model\_file\_id = "your\_model\_file\_id\_here"

download\_file\_from\_drive(model\_file\_id, model\_file)

df, job\_tensors = load\_job\_embeddings()

if df is None or job\_tensors is None:

st.error("Failed to load job embeddings.")

return None

embedding\_dim = len(df["job\_embedding"][0])

model = CollaborativeFiltering(embedding\_dim)

model.load\_state\_dict(torch.load(model\_file))

model.eval()

return model

class CollaborativeFiltering(torch.nn.Module):

def \_\_init\_\_(self, embedding\_dim):

super(CollaborativeFiltering, self).\_\_init\_\_()

self.fc = torch.nn.Sequential(

torch.nn.Linear(embedding\_dim \* 2, 128),

torch.nn.ReLU(),

torch.nn.Linear(128, 64),

torch.nn.ReLU(),

torch.nn.Linear(64, 1),

torch.nn.Sigmoid(),

)

def forward(self, resume\_embedding, job\_embedding):

combined = torch.cat([resume\_embedding, job\_embedding], dim=1)

interaction = self.fc(combined)

return interaction

def load\_skill\_keywords():

skill\_keywords\_file = "models/skill\_keywords.txt"

with open(skill\_keywords\_file, "r") as file:

return [line.strip() for line in file.readlines()]

def download\_file\_from\_drive(file\_id, destination):

os.makedirs(os.path.dirname(destination), exist\_ok=True)

if not os.path.exists(destination):

message\_placeholder = st.empty()

with st.spinner(f"Downloading {destination}..."):

gdown.download(

f"https://drive.google.com/uc?id={file\_id}", destination, quiet=False

)

message\_placeholder.info(f"Downloaded {destination} successfully.")

message\_placeholder.empty()

def parse\_pdf(file):

try:

with st.spinner("Parsing PDF..."):

reader = PdfReader(file)

text = ""

for page in reader.pages:

text += page.extract\_text() + "\n"

st.success("PDF parsed successfully.")

return text.strip()

except Exception as e:

st.error(f"Error parsing PDF: {e}")

return None

def extract\_skills(resume\_text, skill\_keywords):

found\_skills = set()

for skill in skill\_keywords:

if skill.lower() in resume\_text.lower():

found\_skills.add(skill)

return sorted(found\_skills)

def recommend\_jobs(resume\_embedding, job\_tensors, df, top\_n=5):

similarities = torch.nn.functional.cosine\_similarity(resume\_embedding, job\_tensors)

\_, top\_indices = torch.topk(similarities, top\_n)

recommended\_jobs = []

for index in top\_indices:

job\_index = index.item()

job\_title = df.iloc[job\_index]["Title"]

score = similarities[index].item() \* 100

recommended\_jobs.append((job\_title, score))

return recommended\_jobs

def generate\_linkedin\_job\_search\_url(job\_title):

keywords = job\_title

encoded\_keywords = urllib.parse.quote\_plus(keywords)

linkedin\_url = f"https://www.linkedin.com/jobs/search/?keywords={encoded\_keywords}&location=India"

return linkedin\_url

def generate\_naukri\_job\_search\_url(job\_title):

keywords = job\_title

encoded\_keywords = urllib.parse.quote\_plus(keywords)

naukri\_url = f"https://www.naukri.com/{encoded\_keywords}-jobs?k={encoded\_keywords}&nignbevent\_src=jobsearchDeskGNB"

return naukri\_url

@st.cache\_resource(show\_spinner="Initializing LLM...")

def initialize\_llm():

"""Initializes the LLM. You'll need a Google AI API key."""

try:

genai.configure(api\_key=st.secrets["GEMINI\_API\_KEY"]) # Store API key securely

model = genai.GenerativeModel("gemini-2.0-flash")

return model

except Exception as e:

st.error(

f"Error initializing LLM: {e}. Make sure you have a valid API key in st.secrets['GEMINI\_API\_KEY']"

)

return None

def generate\_cover\_letter(resume\_text, job\_description, model):

"""Generates a cover letter using the LLM."""

if not model:

st.error("LLM not initialized.")

return None

prompt = f"""

Given the following resume text:

{resume\_text}

and the following job description:

{job\_description}

You are a professional AI assistant specialized in writing compelling and personalized cover letters. Your goal is to craft a well-structured, engaging, and tailored cover letter based on the user's input. Follow these guidelines:

1. Personalization & Relevance

Address the hiring manager by name if provided; otherwise, use "Hiring Manager."

Include the company name and job title to make the letter specific.

Highlight the user’s most relevant skills, experiences, and achievements that align with the job.

2. Structure & Flow

Opening Paragraph:

Express enthusiasm for the position.

Briefly introduce key qualifications that make the user a strong candidate.

Body Paragraphs (1-2):

Highlight relevant experience and technical skills.

Provide examples of impactful projects, achievements, or metrics (if available).

Showcase how the user’s expertise aligns with the company’s goals.

Closing Paragraph:

Reinforce interest in the role and company.

Express willingness to discuss qualifications further.

End with a professional sign-off (e.g., "Sincerely, [User's Name]").

3. Tone & Readability

Use a professional yet engaging tone (formal, enthusiastic, or persuasive based on user preference).

Keep the content concise and impactful (250-350 words max).

Use clear, structured paragraphs with smooth transitions.

"""

try:

with st.spinner("Generating Cover Letter with LLM..."):

response = model.generate\_content(prompt)

return response.text

except Exception as e:

st.error(f"Error generating cover letter with LLM: {e}")

return None

# ================== Streamlit UI ==================

# --- CSS Styling ---

with open("styles/style.css") as css:

st.markdown(f"<style>{css.read()}</style>", unsafe\_allow\_html=True)

# --- Font Awesome Icons ---

st.markdown(

"""

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css" integrity="sha512-9usAa10IRO0HhonpyAIVpjrylPvoDwiPUiKdWk5t3PyolY1cOd4DSE0Ga+ri4AuTroPR5aQvXU9xC6qOPnzFeg==" crossorigin="anonymous" referrerpolicy="no-referrer" />

<script src="https://kit.fontawesome.com/a838ad3310.js" crossorigin="anonymous"></script>

""",

unsafe\_allow\_html=True,

)

# --- Page Header ---

st.markdown(

"""

<p style="font-size: 35px; font-family: 'Gugi', serif;font-weight: 400;">ROLE RECOMMENDATION SYSTEM</p>

""",

unsafe\_allow\_html=True,

)

st.image(

image="https://www.careerguide.com/career/wp-content/uploads/2021/01/a2413959910293.5a33a9bde96e8.gif",

use\_container\_width=True,

)

# --- Navigation ---

cols = st.columns(3)

try:

with cols[0]:

st.page\_link(page="app.py", icon="🏠", label="Home")

with cols[1]:

st.page\_link(

page="pages/Promotion Model.py", icon="💹", label="Promotion Model"

)

with cols[2]:

st.page\_link(

page="pages/Role Recommendation Model.py",

icon="🏢",

label="Role Recommendation",

)

except Exception as e:

st.error(f"Error loading page links: {e}")

@st.cache\_resource

def initialize\_app():

"""Initializes core components of the application."""

# Load models and data (only once)

try:

embed\_model = SentenceTransformer("all-MiniLM-L6-v2")

except Exception as e:

st.error(f"Error loading SentenceTransformer model: {e}")

st.stop()

df, job\_tensors = load\_job\_embeddings()

if df is None or job\_tensors is None:

st.stop()

model = load\_model()

if model is None:

st.stop()

skill\_keywords = load\_skill\_keywords()

return embed\_model, df, job\_tensors, model, skill\_keywords

# --- Initialize app components ---

embed\_model, df, job\_tensors, model, skill\_keywords = initialize\_app()

llm\_model = initialize\_llm()

# --- Tabs ---

tabs = st.tabs(["Resume Upload", "Job Description Scoring"])

# Helper function to create a docx file for the cover letter

def create\_docx\_download\_button(cover\_letter):

"""Creates a docx file from the cover letter text and generates a download button."""

if cover\_letter:

doc = Document()

doc.add\_paragraph(cover\_letter)

buffer = BytesIO()

doc.save(buffer)

buffer.seek(0)

st.download\_button(

label="Download Cover Letter (docx)",

data=buffer,

file\_name="cover\_letter.docx",

mime="application/vnd.openxmlformats-officedocument.wordprocessingml.document",

)

with tabs[0]: # Resume Upload Tab

st.write("Upload your resume (PDF) to find matching roles!")

uploaded\_file = st.file\_uploader(

"Upload your resume (PDF only)", type=["pdf"], key="resume\_upload\_tab"

)

if uploaded\_file:

resume\_text = parse\_pdf(uploaded\_file) # parse\_pdf handles errors

if resume\_text:

st.session\_state.resume\_text = resume\_text

st.markdown("### Extracted Resume Text")

st.markdown(f"```text\n{resume\_text}\n```")

skills = extract\_skills(resume\_text, skill\_keywords)

st.markdown("### Extracted Skills")

if skills:

st.markdown(", ".join(skills))

else:

st.markdown("No skills matched from the predefined list.")

if st.button("Recommend Roles"):

if (

"resume\_text" in st.session\_state

and st.session\_state.resume\_text.strip() != ""

):

with st.spinner("Generating resume embedding..."):

resume\_embedding = torch.tensor(

embed\_model.encode(st.session\_state.resume\_text),

dtype=torch.float,

).unsqueeze(0)

with st.spinner("Recommending Roles..."):

recommended\_jobs = recommend\_jobs(

resume\_embedding, job\_tensors, df, top\_n=5

)

st.success("Recommended Roles:")

for i, (job\_title, score) in enumerate(recommended\_jobs):

linkedin\_url = generate\_linkedin\_job\_search\_url(job\_title)

naukri\_url = generate\_naukri\_job\_search\_url(job\_title)

st.write(

f"\*\*{job\_title}\*\* - Suitability Score: {score:.1f}"

)

col1, col2 = st.columns(2)

with col1:

linkedin\_button = f"""

<a href="{linkedin\_url}" target="\_blank">

<button style="

background-color: #0077B5;

border: none;

color: white;

padding: 10px 20px;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 14px;

cursor: pointer;

border-radius: 5px;

"><i class="fab fa-linkedin"></i> LinkedIn Jobs

</button>

</a>

"""

st.markdown(linkedin\_button, unsafe\_allow\_html=True)

with col2:

naukri\_button = f"""

<a href="{naukri\_url}" target="\_blank">

<button style="

background-color: #E07A5F;

border: none;

color: white;

padding: 10px 20px;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 14px;

cursor: pointer;

border-radius: 5px;

"><i class="fas fa-building"></i> Naukri Jobs

</button>

</a>

"""

st.markdown(naukri\_button, unsafe\_allow\_html=True)

else:

st.warning("Please upload a valid PDF resume.")

else:

st.warning("Could not parse the uploaded PDF.") # parse\_pdf handles errors

else:

st.warning("Please upload a valid PDF resume.")

with tabs[1]: # Job Description Scoring Tab

st.write("Enter a job description and use the resume uploaded above to score it.")

if "resume\_text" in st.session\_state:

# Initialize job\_description in session state if it doesn't exist

if "job\_description" not in st.session\_state:

st.session\_state.job\_description = ""

job\_description = st.text\_area(

"Enter Job Description:",

height=200,

key="job\_description",

value=st.session\_state.job\_description,

)

if st.button("Score Resume"):

if job\_description.strip():

with st.spinner("Scoring..."):

resume\_embedding = torch.tensor(

embed\_model.encode(st.session\_state.resume\_text),

dtype=torch.float,

).unsqueeze(0)

job\_description\_embedding = torch.tensor(

embed\_model.encode(job\_description), dtype=torch.float

).unsqueeze(0)

similarity\_score = (

torch.nn.functional.cosine\_similarity(

resume\_embedding, job\_description\_embedding

).item()

\* 100

)

st.success(

f"Resume Suitability Score for this job description: {similarity\_score:.1f}"

)

st.session\_state.similarity\_score = similarity\_score # Save

else:

st.warning("Please enter a job description.")

# Generate Cover Letter Section (Conditionally Displayed)

if (

llm\_model

and "similarity\_score" in st.session\_state

and "job\_description" in st.session\_state

and st.session\_state.job\_description.strip()

):

if st.button("Generate Cover Letter"):

cover\_letter = generate\_cover\_letter(

st.session\_state.resume\_text,

st.session\_state.job\_description,

llm\_model,

)

if cover\_letter:

st.subheader("Generated Cover Letter:")

st.write(cover\_letter)

create\_docx\_download\_button(cover\_letter) # Display download button

else:

st.warning("Please upload a resume on the 'Resume Upload' tab first.")

### **PROMOTION PREDICTION**

import streamlit as st

import pandas as pd

import numpy as np

import joblib

from tensorflow.keras.models import Sequential, load\_model

st.set\_page\_config(

page\_title="Promotion Model",

page\_icon="https://cdn2.iconfinder.com/data/icons/knowledge-promotion-3/64/career\_leadership\_learn\_development\_growth\_motivation-256.png",

# initial\_sidebar\_state="expanded", #Optional: remove or uncomment as needed.

)

# --- Navigation Bar ---

cols = st.columns(3) # Create 3 columns

try:

with cols[0]:

st.page\_link(page="app.py", icon="🏠", label="Home")

with cols[1]:

st.page\_link(

page="pages/Promotion Model.py", icon="💹", label="Promotion Model"

)

with cols[2]:

st.page\_link(

page="pages/Role Recommendation Model.py",

icon="🏢",

label="Role Recommendation",

)

except Exception as e:

st.error(f"Error loading page links: {e}")

st.markdown(

"""

<p style="font-size: 40px; font-family: 'Gugi', serif;font-weight: 400;">EMPLOYEE PROMOTION PREDICTION</p>

""",

unsafe\_allow\_html=True,

)

st.image(

image="https://pulsemotivation.com/wp-content/uploads/2022/02/Pulse-Motivation-Employee-engagement-strategy%E2%80%A8.gif",

use\_container\_width="auto",

caption="Employee Promotion Model",

)

with open("styles/style.css") as css:

st.markdown(f"<style>{css.read()}</style>", unsafe\_allow\_html=True)

# --- Model Loading and Preprocessing ---

model\_path = "models/employee\_promotion\_model.h5"

try:

model = load\_model(model\_path)

scaler = joblib.load("models/scaler.pkl")

label\_encoders = joblib.load("models/label\_encoders.pkl")

except FileNotFoundError:

st.error(

"Error: Model files not found. Please ensure 'models' folder exists and contains the necessary files."

)

st.stop() # Stop execution if model files are missing

except Exception as e:

st.error(f"An error occurred loading the model: {e}")

st.stop()

# --- Input Fields (Now in Main Content) ---

st.write(

"Use this tool to predict the likelihood of an employee being promoted based on their profile and performance data."

)

st.markdown(

"""

<p style="font-size: 35px;font-weight: bold;"> Input Employee Details</p>

""",

unsafe\_allow\_html=True,

)

def get\_numerical\_input(label, min\_value, max\_value, default\_value):

input\_value = st.text\_input(

f"{label} (Range: {min\_value}-{max\_value})", value=str(default\_value)

)

try:

input\_value = float(input\_value)

if input\_value < min\_value or input\_value > max\_value:

st.warning(

f"Please enter a value between {min\_value} and {max\_value} for {label}."

)

return None

return input\_value

except ValueError:

st.warning(f"Please enter a valid number for {label}.")

return None

# Input fields (moved to main content)

employee\_id = get\_numerical\_input("Employee ID", 1000, 9999, 1000)

department = st.selectbox(

"Department",

[

"Sales & Marketing",

"Operations",

"Technology",

"Analytics",

"R&D",

"Procurement",

"Finance",

"HR",

"Legal",

],

)

region = st.selectbox(

"Region",

[

"Bangalore",

"Hyderabad",

"Pune",

"Chennai",

"Mumbai",

"Delhi",

"Noida",

"Gurgaon",

"Kolkata",

"Ahmedabad",

"Jaipur",

"Lucknow",

"Kochi",

"Thiruvananthapuram",

"Indore",

"Bhubaneswar",

"Nagpur",

"Visakhapatnam",

"Chandigarh",

"Coimbatore",

"Mysore",

"Vadodara",

"Patna",

"Ranchi",

"Guwahati",

"Surat",

"Hubli",

"Jamshedpur",

"Dehradun",

"Raipur",

"Amritsar",

"Pondicherry",

"Shillong",

"Shimla",

],

)

education = st.selectbox(

"Education Level",

["Master's & above", "Bachelor's", "Below Secondary", "Master's", "High School"],

)

gender = st.selectbox("Gender", ["m", "f"])

recruitment\_channel = st.selectbox(

"Recruitment Channel",

["linkedin", "sourcing", "other", "Naukri", "Indeed", "referred"],

)

no\_of\_trainings = get\_numerical\_input("Number of Trainings", 0, 10, 2)

age = get\_numerical\_input("Age", 20, 60, 30)

previous\_year\_rating = get\_numerical\_input("Previous Year Rating", 0.0, 5.0, 3.0)

length\_of\_service = get\_numerical\_input("Length of Service (Years)", 0, 20, 5)

awards\_won = get\_numerical\_input("Awards Won", 0, 5, 0)

avg\_training\_score = get\_numerical\_input("Average Training Score", 50, 100, 75)

# Check for valid inputs

if None in [

employee\_id,

no\_of\_trainings,

age,

previous\_year\_rating,

length\_of\_service,

awards\_won,

avg\_training\_score,

]:

st.error("Please correct invalid inputs before proceeding.")

st.stop()

# --- Data Preprocessing and Prediction ---

user\_input = {

"employee\_id": employee\_id,

"department": department,

"region": region,

"education": education,

"gender": gender,

"recruitment\_channel": recruitment\_channel,

"no\_of\_trainings": no\_of\_trainings,

"age": age,

"previous\_year\_rating": previous\_year\_rating,

"length\_of\_service": length\_of\_service,

"awards\_won": awards\_won,

"avg\_training\_score": avg\_training\_score,

}

def preprocess\_input(user\_input, scaler, label\_encoders):

for col in label\_encoders.keys():

if user\_input[col] not in label\_encoders[col].classes\_:

user\_input[col] = label\_encoders[col].classes\_[0]

user\_df = pd.DataFrame([user\_input])

for col in label\_encoders.keys():

user\_df[col] = label\_encoders[col].transform(user\_df[col])

scaled\_features = scaler.transform(user\_df)

return scaled\_features

def generate\_detailed\_explanation(user\_input, prediction\_probability, feature\_weights):

top\_features = sorted(feature\_weights.items(), key=lambda x: x[1], reverse=True)[:5]

explanation = f"Prediction: The employee's likelihood of promotion is {prediction\_probability:.2f}.\n\n"

explanation += "Key Factors Influencing the Prediction:\n"

for feature, weight in top\_features:

value = user\_input.get(feature, "N/A")

explanation += f"- {feature.capitalize()}: {value} (Importance: {weight:.2f})\n"

explanation += "\nConclusion:\n"

explanation += f"The overall prediction suggests a {'high' if prediction\_probability > 0.75 else 'moderate' if prediction\_probability > 0.5 else 'low'} likelihood of promotion."

return explanation

def get\_feature\_weights(model, feature\_names):

first\_layer\_weights = model.layers[0].get\_weights()[0]

average\_weights = np.mean(np.abs(first\_layer\_weights), axis=1)

feature\_importance = dict(zip(feature\_names, average\_weights))

sorted\_features = dict(

sorted(feature\_importance.items(), key=lambda x: x[1], reverse=True)

)

return sorted\_features

# --- Prediction Button and Results ---

if st.button("Predict Promotion"):

try:

preprocessed\_input = preprocess\_input(user\_input, scaler, label\_encoders)

prediction\_probability = model.predict(preprocessed\_input)[0][0]

feature\_names = list(user\_input.keys())

feature\_weights = get\_feature\_weights(model, feature\_names)

explanation = generate\_detailed\_explanation(

user\_input, prediction\_probability, feature\_weights

)

st.subheader("Prediction Results")

st.write(f"Promotion Likelihood: {prediction\_probability:.2f}")

st.subheader("Detailed Explanation")

st.write(explanation)

except Exception as e:

st.error(f"An error occurred during prediction: {e}")

## **USER DOCUMENTATION**

**INTRODUCTION**

Welcome to the AI Workforce Planning Tools! This guide will help you navigate and use the system to improve your workforce planning processes. Designed to enhance workforce management, it uses machine learning models for promotion prediction and intelligent role recommendation. This tool is designed to assist HR managers, recruiters, and employees by providing valuable insights and assistance for data-driven decision-making, matching talent with the right roles, and streamlining the job application process.

**1. GETTING STARTED**

* **ACCESSING THE APPLICATION:**
  + The AI Workforce Planning Tools is a web-based application. You can access it by entering the following address.
  + The application is best viewed using the latest versions of Google Chrome, Mozilla Firefox, or Safari.
  + No account creation is needed. The system is readily accessible without requiring any login credentials.
* **SYSTEM REQUIREMENTS:**
  + To ensure optimal performance, your system should meet the following minimum requirements:
    - Operating System: Windows 10 or later, macOS 10.15 (Catalina) or later, or Linux (Ubuntu 18.04 or later)
    - Web Browser: Google Chrome (latest version), Mozilla Firefox (latest version), or Safari (latest version)
    - Internet Connection: A stable internet connection is required for accessing and using the application.

**2. NAVIGATING THE INTERFACE**

* **MAIN MENU:**
  + The main menu is located at the top of the screen and consists of the following options:
    - **HOME:** Returns you to the main overview page of the AI Workforce Planning Tools.
    - **PROMOTION MODEL:** Takes you to the promotion prediction tool, where you can input employee details and predict their likelihood of promotion.
    - **ROLE RECOMMENDATION:** Navigates you to the role recommendation system, which helps you match employees to suitable job roles based on their skills and experience.
* **HOME PAGE:**
  + The home page provides an overview of the AI Workforce Planning Tools and its key features. It includes a brief description of the system's purpose and functionalities, as well as links to the main menu options.
  + You'll also find a welcoming message and an image.
* **NAVIGATION BAR:**
  + The navigation bar is located at the top of the screen and allows you to easily access different sections of the application. It includes links to the main menu options (Home, Promotion Model, Role Recommendation) and any other relevant pages or features.

**3. USING THE PROMOTION MODEL**

* **ACCESSING THE PROMOTION MODEL:**
  + To access the Promotion Model, click on the "Promotion Model" link in the main menu or navigation bar. This will take you to the promotion prediction tool.
* **INPUTTING EMPLOYEE DETAILS:**
  + On the Promotion Model page, you will find a form where you can enter employee details. Please provide the following information:
    - **Employee ID:** Enter the employee's unique identification number (a 4 digit number between 1000 and 9999). Example: 1234
    - **Department:** Select the employee's department from the dropdown list. Available options include: Sales & Marketing, Operations, Technology, Analytics, R&D, Procurement, Finance, HR, Legal.
    - **Region:** Select the employee's work location from the dropdown list. Available options include a wide variety of cities in India.
    - **Education Level:** Choose the employee's highest level of education from the dropdown list. Options include: Master's & above, Bachelor's, Below Secondary, Master's, High School.
    - **Gender:** Select the employee's gender from the dropdown list. Options include: m (male), f (female).
    - **Recruitment Channel:** Select the employee's recruitment channel from the dropdown list. Options include: linkedin, sourcing, other, Naukri, Indeed, referred.
    - **Number of Trainings:** Enter the number of training programs the employee has completed (a whole number between 0 and 10). Example: 3
    - **Age:** Enter the employee's age in years (a whole number between 20 and 60). Example: 35
    - **Previous Year Rating:** Enter the employee's performance rating from the previous year (a number between 0.0 and 5.0). Example: 4.5
    - **Length of Service:** Enter the employee's length of service in years (a whole number between 0 and 20). Example: 7
    - **Awards Won:** Enter the number of awards the employee has won (a whole number between 0 and 5). Example: 1
    - **Average Training Score:** Enter the employee's average score from training programs (a number between 50 and 100). Example: 85
  + **Tips for Entering Accurate Information:**
    - Double-check all entered values to ensure accuracy.
    - Select the correct option from the dropdown lists.
    - Provide valid data for all input fields.
* **Running the Prediction:**
  + After entering all the required employee details, click the "Predict Promotion" button to initiate the promotion prediction process.
  + The system will process the input data and generate a prediction of the employee's likelihood of promotion.
* **Interpreting the Results:**
  + The prediction results will be displayed on the screen. The results include:
    - **Promotion Likelihood:** A score between 0 and 1 indicating the employee's likelihood of being promoted. A higher score indicates a greater likelihood of promotion.
    - **Detailed Explanation:** A breakdown of the key factors influencing the prediction, including the employee's performance ratings, training scores, department, and other relevant variables. This explanation provides insights into the reasons behind the prediction and helps you understand the employee's strengths and areas for improvement.

**4. Using the Role Recommendation System**

* **Accessing the Role Recommendation System:**
  + To access the Role Recommendation System, click on the "Role Recommendation" link in the main menu or navigation bar. This will take you to the role recommendation tool.
* **Tabs:**
  + The Role Recommendation System is divided into two tabs:
    - **Resume Upload:** This tab allows you to upload a resume and receive recommendations for suitable job roles.
    - **Job Description Scoring:** This tab allows you to score a resume against a specific job description.
* **Resume Upload Tab:**
  + To use the Resume Upload tab:
    - * Click on the "Resume Upload" tab.
      * Click the "Browse" button to select a resume file from your computer.
      * The system supports PDF file format. Ensure that the resume is in PDF format before uploading.
      * The system will parse the resume. After a moment, the "Extracted Resume Text" and "Extracted Skills" will display.
      * Click the "Recommend Roles" button to generate a list of recommended job roles based on the skills and experience extracted from the resume.
      * The system will display a list of recommended job titles, along with a suitability score for each role. The suitability score indicates the degree of match between the candidate's skills and the job requirements.
      * To find the job, there are two buttons: LinkedIN Jobs & Naukri Jobs
* **Job Description Scoring Tab:**
  + To use the Job Description Scoring tab:
    - * Click on the "Job Description Scoring" tab.
      * In the "Enter Job Description" text area, paste the job description for the role you want to score.
      * Click the "Score Resume" button to generate a suitability score for the uploaded resume against the entered job description.
      * The system will display a suitability score, indicating the degree of match between the resume and the job description.
      * Once the resume is scored, you may click "Generate Cover Letter."
      * If the system provides Cover Letter, click "Download Cover Letter (docx)" button to download it.
      * Make sure the web browser has permission to download DOCX files.

**5. Additional Features and Tips**

* **General Tips for Using the System:**
  + To achieve the best results, ensure that the data you input is accurate and consistent.
  + Take the time to explore all the features and functionalities of the system.
  + If you encounter any issues or have questions, refer to the troubleshooting section or contact technical support.
  + Note that AI model has limitations in this applications.
* **Troubleshooting:**
  + **Issue:** The system is not loading or responding.
    - Solution: Check your internet connection and try refreshing the page. If the issue persists, contact technical support.
  + **Issue:** The promotion prediction results seem inaccurate.
    - Solution: Double-check the input data to ensure that it is correct and complete. If the issue persists, consider providing feedback to the system administrators.
  + **Issue:** Resume upload fails.
    - Solution: Ensure the resume is a PDF, or make sure file size is not too large
* **Data Privacy and Security:**
  + All user data is protected using secure encryption protocols.
  + The system does not share any personal information with third parties.

## **GLOSSARY**

This glossary defines key terms used in this report related to artificial intelligence, machine learning, workforce planning, and human resources.

* **ACCURACY:** The degree to which a model's predictions are correct. In the context of the Promotion Model, accuracy refers to the percentage of employees for whom the model correctly predicted their promotion status.
* **AI (ARTIFICIAL INTELLIGENCE):** The capability of a machine to imitate intelligent human behavior, such as learning, problem-solving, and decision-making.
* **ALGORITHM:** A set of rules or instructions that a computer follows to solve a problem or perform a task.
* **AUC-ROC** (Area Under the Receiver Operating Characteristic Curve): A performance metric that measures a model's ability to distinguish between positive and negative cases. A higher AUC-ROC score indicates better performance.
* **BIAS (IN AI):** Systematic errors in a model's predictions that result in unfair or discriminatory outcomes for certain groups of individuals.
* **DATA PREPROCESSING:** The process of cleaning, transforming, and preparing raw data for use in machine learning models.
* **DEEP LEARNING:** A type of machine learning that uses artificial neural networks with multiple layers to analyze data and make predictions.
* **FEATURE (MACHINE LEARNING):** An individual measurable property or characteristic of a phenomenon being observed. Features are used as inputs to machine learning models. Examples include employee age, training score, and length of service.
* **F1-SCORE:** A metric that measures a model's accuracy, taking into account both precision and recall. It is the harmonic mean of precision and recall.
* **HYPERPARAMETER TUNING:** The process of optimizing the parameters of a machine learning model to improve its performance.
* **HR (HUMAN RESOURCES):** The department responsible for managing employees, including recruitment, training, performance management, and compensation.
* **MACHINE LEARNING (ML):** A type of artificial intelligence that enables computer systems to learn from data without being explicitly programmed.
* **MODEL (MACHINE LEARNING):** A mathematical representation of a process or system that is learned from data. Machine learning models are used to make predictions or decisions based on new data.
* **NLP (NATURAL LANGUAGE PROCESSING):** A branch of artificial intelligence that deals with the interaction between computers and human language. NLP techniques are used to analyze and understand text data.
* **RECOMMENDATION SYSTEM:** A system that suggests items or options to users based on their preferences or past behavior.
* **SENTENCE EMBEDDINGS:** Numerical representations of sentences or phrases that capture their semantic meaning. Used for calculating similarity between job descriptions and resumes.
* **SKILLS GAP ANALYSIS:** A process of identifying the differences between the skills that an organization needs and the skills that its employees possess.
* **SUCCESSION PLANNING:** The process of identifying and developing employees to fill key leadership positions in the future.
* **WORKFORCE PLANNING:** The process of analyzing an organization's current workforce, forecasting future workforce needs, and developing strategies to address any gaps.

## **PROJECT RECOGNITIONS**

# **REFERENCES**

## **BOOK REFERENCES**

Agarwal, S., Mittal, A., Motwani, M., & Goyal, N. Recommendation System for Job Portal using Machine Learning. 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT). IEEE, 2020.

Relevance: Focuses on job recommendation systems using machine learning, applicable to the role recommendation component.

Angrave, Denise, Richard Charlwood, Mark Kirkpatrick, Tom Lawrence, and Mark Stuart. "HR and Analytics: Why HR is Set to Fail the Big Data Challenge." Human Resource Management Journal, vol. 26, no. 1, Wiley, 2016.

Relevance: Explores the challenges HR faces in leveraging big data and analytics.

Bamberger, Peter A., Ilan Meshoulam, and Yehouda Shenhav. Human Resource Strategy: Formulation, Implementation, and Impact. McGraw-Hill/Irwin, 2014.

Relevance: A comprehensive textbook on HR strategy including workforce planning.

Cappelli, Peter. "Talent Management for the Twenty-First Century." Harvard Business Review, March 2008.

Relevance: Discusses talent management challenges and the need for innovative approaches.

Huselid, Mark A. "The Impact of Human Resource Management Practices on Turnover, Productivity, and Corporate Financial Performance." Academy of Management Journal, vol. 38, no. 3, Academy of Management, 1995.

Relevance: Establishes the link between effective HR practices and organizational performance.

Lepak, David P., and Scott A. Snell. "The Human Resource Architecture: Toward a Theory of Human Capital Allocation and Development." Academy of Management Review, vol. 24, no. 1, Academy of Management, 1999.

Relevance: Presents a framework for aligning HR practices with organizational goals.

Malik, Pooja, and Sheetal Kumar. "A Systematic Review of Machine Learning and Deep Learning Approaches for Employee Promotion Prediction." International Journal of Engineering and Advanced Technology, vol. 9, no. 1, Blue Eyes Intelligence Engineering & Sciences Publication, 2019.

Relevance: Explores machine learning techniques used for employee promotion prediction.

Tambe, Prasanna, Lorin Hitt, Bart Thielman, and Karthik Dhaumyai. "Best Practices for Analytics in Human Resources." MIS Quarterly Executive, vol. 11, no. 3, Society for Information Management, 2012.

Relevance: Provides insights into the use of analytics in HR, including workforce planning.

Van den Heuvel, Sander, and Paul Bondarouk. "The Design of Talent Management Systems: A Complex Task." The International Journal of Human Resource Management, vol. 28, no. 9, Taylor & Francis, 2017.

Relevance: Examines the complexities of designing talent management systems.

Vasilescu, B., Capiluppi, A., & Serebrenik, A. AI-Based Recruitment: What Does It Mean for Software Engineering?. IEEE Software, 35(6), 80-85, 2018.

Relevance: Provides insights on applying AI in Recruitment.

## **WEB REFERENCES**

Bersin, Josh. "HR Technology Disruptions for 2024: AI, Skills, and the Employee Experience." Josh Bersin, 2024. https://joshbersin.com/2024/01/hr-technology-disruptions-for-2024-ai-skills-and-the-employee-experience/ (Accessed October 26, 2024).

Relevance: Discusses current HR technology trends, including the role of AI and skills-based approaches in the employee experience.

Gartner. "Gartner Forecasts Worldwide AI Spending to Reach $300 Billion in 2024." Gartner Newsroom, 2024. https://www.gartner.com/en/newsroom/press-releases/2024/04/gartner-forecasts-worldwide-ai-spending-to-reach-300-billion-in-2024 (Accessed October 26, 2024).

Relevance: Provides data on the growth and investment in AI, offering a broader context for the adoption of AI in HR.

SHRM (Society for Human Resource Management). "Artificial Intelligence in HR: A Practical Guide." SHRM Resources, 2023. https://www.shrm.org/resourcesandtools/hr-topics/technology/pages/artificial-intelligence-in-hr.aspx (Accessed October 26, 2024).

Relevance: A practical guide from a reputable HR organization, covering various applications of AI in HR.

AIHR (Academy to Innovate HR). "AI in HR: The Definitive Guide for 2024." AIHR Digital, 2024. https://www.aihr.com/blog/ai-in-hr/ (Accessed October 26, 2024).

Relevance: A comprehensive overview of AI in HR, covering various applications and use cases.

Deloitte. "2023 Global Human Capital Trends." Deloitte Insights, 2023. https://www2.deloitte.com/us/en/insights/focus/human-capital-trends.html (Accessed October 26, 2024).

Relevance: Discusses key trends shaping the future of work, including the impact of AI and technology on human capital management.

Google AI Blog. "Introducing Gemini 2.0: Our Next-Generation Model." Google AI Blog, 2024. https://ai.googleblog.com/ (Accessed October 26, 2024).

Relevance: This is relevant if you use Google's models, it showcases the latest innovations in AI and machine learning if the models you use are mentioned.

TensorFlow. "TensorFlow: An Open Source Machine Learning Framework for Everyone." TensorFlow Website, 2024. https://www.tensorflow.org/ (Accessed October 26, 2024).

Relevance: Provides an overview of TensorFlow, including its features, benefits, and use cases.

PyTorch. "PyTorch: An Open Source Machine Learning Framework." PyTorch Website, 2024. https://pytorch.org/ (Accessed October 26, 2024).

Relevance: If you have any Pytorch, this provides an overview of PyTorch.

Streamlit. "Streamlit — The fastest way to build and share data apps." Streamlit Website, 2024. https://streamlit.io/ (Accessed October 26, 2024).

Relevance: This is if you have used Streamlit.