




# Nimra Bilal

## HireSense - FYP SRS Report

-  plag checker
-  FYP (F25 batch)
-  lahore garrison LGU

---

### Document Details

**Submission ID**

trn:oid::1:3278303468

**Submission Date**

Jun 17, 2025, 1:05 AM GMT+5

**Download Date**

Jun 17, 2025, 1:07 AM GMT+5

**File Name**

HireSense\_SRS.docx

**File Size**

255.0 KB

**14 Pages****2,581 Words****15,463 Characters**

## \*% detected as AI

AI detection includes the possibility of false positives. Although some text in this submission is likely AI generated, scores below the 20% threshold are not surfaced because they have a higher likelihood of false positives.

**Caution: Review required.**

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

### Disclaimer

Our AI writing assessment is designed to help educators identify text that might be prepared by a generative AI tool. Our AI writing assessment may not always be accurate (it may misidentify writing that is likely AI generated as AI generated and AI paraphrased or likely AI generated and AI paraphrased writing as only AI generated) so it should not be used as the sole basis for adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.

## Frequently Asked Questions

### How should I interpret Turnitin's AI writing percentage and false positives?

The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (\*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

### What does 'qualifying text' mean?

Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.



# HireSense – AI-Based Resume Matching System

## Software Requirement Specifications

### Bachelor of Science in Software Engineering

By

	Name	Registration #	Mobile #	E-Mail
1	Muhammad Farrukh Ejaz	BSSE-SP21-027	0332-4487187	farrukhejaz125@gmail.com
2	Muhammad Zaid Ali	BSSE-SP22-070	0303-5881547	me.mzaidali@gmail.com
3	Hamza Abbas Bajwa	BSSE-FA21-210	0310-4822584	hb060820@gmail.com

**Supervised by**

**Co-Supervised by**

---

**Nimra Bilal**

Teacher

---

**Adnan Tahir**

Industrial



Department of Software Engineering

Lahore Garrison University

Lahore

# Table of Contents

<b>1. Introduction</b>	3
1.1 Purpose	3
1.2 Document Conventions	3
1.3 Intended Audience and Reading Suggestions	3
1.4 Product Scope	3
<b>2. Overall Description</b>	4
2.1 Product Perspective	4
2.2 Product Functions	4
2.3 User Classes and Characteristics	5
2.4 Operating Environment	5
2.5 Design and Implementation Constraints	5
2.6 User Documentation	6
2.7 Assumptions and Dependencies	6
<b>3. External Interface Requirements</b>	6
3.1 User Interfaces	6
3.2 Hardware Interfaces	7
3.3 Software Interfaces	7
3.4 Communications Interfaces	8
<b>4. System Features</b>	8
<b>5. Other Nonfunctional Requirements</b>	11
5.1 Performance Requirements	11
5.2 Safety Requirements	11
5.3 Security Requirements	11
5.4 Software Quality Attributes	12
5.5 Business Rules	12
<b>6. System Modeling</b>	12
6.1 Data Flow Diagram	12
6.2 Sequence Diagram	13
6.3 Architectural Diagram	13
<b>References</b>	14

# 1. Introduction

## 1.1 Purpose

The purpose of this Software Requirements Specification (SRS) document is to define the requirements for the HireSense web application, an AI-powered recruitment platform developed as part of the final year project for the Bachelor of Science in Software Engineering. HireSense aims to automate the resume shortlisting process and improve job matching through artificial intelligence, specifically using the BERT model with transformers for semantic understanding. This document outlines the full system requirements including functional, non-functional, and interface-related needs. It will hopefully serve as a detailed guide for developers, testers, stakeholders, and academic evaluators to understand and evaluate the system.

## 1.2 Document Conventions

This document is written in formal academic English using standard paragraph formatting. Headings and subheadings follow a logical numbering pattern (e.g., 1.1, 1.2). Each requirement is described in paragraph form and where necessary. Priority levels for features will be mentioned using common terms such as High, Medium, or Low. No special color coding or technical symbols are used to maintain simplicity.

## 1.3 Intended Audience and Reading Suggestions

This SRS is primarily intended for project supervisors, academic reviewers, team members, and future developers who may work on extending the system. Testers and documentation writers may also refer to this document. Readers unfamiliar with the technical terms should begin with Section 1 and 2 for an overview. Developers should focus on Sections 3 and 4 for specific functional and interface requirements. Non-functional aspects are detailed in Section 5.

## 1.4 Product Scope

HireSense is a web-based AI recruitment system designed to automate the screening and shortlisting of candidates by matching resumes to job descriptions using semantic analysis. It supports job seekers through CV recommendations and recruiters through smart candidate filtering. The project contributes to modernizing recruitment processes and reducing hiring delays. It supports scalable, accurate, and intelligent hiring workflows through its user-friendly portals for job seekers, recruiters, and admins. HireSense fits into the broader goal of integrating machine learning into everyday business functions like talent acquisition.

## 2. Overall Description

### 2.1 Product Perspective

HireSense is a standalone software solution designed from scratch to serve as an intelligent recruitment management system. It does not build on an existing product but introduces a fresh approach by integrating AI capabilities such as resume parsing, BERT-based matching, and job recommendations into one cohesive platform. The product includes three distinct portals: one for job seekers, one for recruiters, and one for administrators. It interacts with a MySQL database for storing user and job data and leverages the BERT model for semantic similarity analysis between resumes and job descriptions. The platform is developed using Django for the backend and Bootstrap with JavaScript for the frontend.

### 2.2 Product Functions

The main functions of HireSense include:

- **Secure User Authentication** for job seekers, recruiters, and administrators.
- **Resume Upload and Parsing** to extract key information like skills, education, and experience.
- **Job Posting and Management** by recruiters.
- **AI-Based Resume-to-Job Matching** using BERT and cosine similarity.
- **AI-Driven Resume Scoring** and suggestions for improvement.
- **Job Recommendations** for job seekers based on their resume content.
- **Resume Downloading and Management** by recruiters.
- **Admin Functionalities** including user, job, and resume management, as well as analytics and logs.

## 2.3 User Classes and Characteristics

- **Job Seekers**

Individuals looking for employment. They may not have technical expertise, so the interface should be intuitive. They will upload resumes, receive job recommendations, and get AI-powered suggestions for improvement.

- **Recruiters**

Professionals or organizations posting job listings and reviewing candidate profiles. They expect fast, intelligent filtering of resumes and a simplified process for contacting suitable candidates.

- **Administrators**

Responsible for overseeing the platform's activities. They manage user accounts, review job postings, ensure system security, and monitor logs and analytics. They require access to the entire system.

## 2.4 Operating Environment

The system will run as a web application accessible through modern web browsers (e.g., Chrome, Firefox, Edge). The backend is built with Django (Python), using a MySQL database hosted on a server. The frontend uses HTML, CSS, Bootstrap, and JavaScript for responsiveness. The AI model (BERT) will be deployed using Python libraries and served on the same backend environment. The platform will operate on a Linux-based web server with secure HTTPS communication.

## 2.5 Design and Implementation Constraints

- The application must use **Django** for backend development.
- The database system must be **MySQL**.
- Resume parsing must be implemented using **pdfminer.six**.
- Resume-job matching must use **BERT** with cosine similarity.
- Only **PDF resumes** are supported.
- User roles and permissions must be strictly managed to ensure data privacy and platform integrity.

## 2.6 User Documentation

The following documentation will be delivered with the software:

- User manual for job seekers and recruiters
- Administrator guide
- Online help section integrated into the portal
- PDF tutorials for common actions (uploading a resume, posting a job, etc.)

## 2.7 Assumptions and Dependencies

- It is assumed that users will have a stable internet connection.
- The system depends on third-party libraries like pdfminer.six, TensorFlow(for BERT), and Django packages.
- The BERT model will be locally hosted; cloud APIs like OpenAI are not used.
- If any major libraries or frameworks change, the system may require maintenance updates.
- Deployment assumes server-side file system support for storing uploaded CVs.

## 3. External Interface Requirements

### 3.1 User Interfaces

The system will offer a clean, user-friendly graphical user interface (GUI) for three types of users: job seekers, recruiters, and administrators. Each will access a dedicated portal with features relevant to their role.

- **Job Seeker Interface:**
  - Sign-up/Login page with email and password validation.
  - Dashboard showing job recommendations and resume score.
  - Upload resume button with progress indicator.
  - Resume suggestions section powered by AI.
  - Application history and tracking page.
- **Recruiter Interface:**
  - Dashboard for posting jobs and viewing applicants.



- Resume search, filtering, and weighted scoring options.
- Candidate shortlist with scheduling options for interviews.
- Download resumes in bulk.
- **Admin Interface:**
  - Full user management dashboard.
  - View and moderate job postings.
  - Resume monitoring tools.
  - System logs, analytics reports, and platform health indicators.

All interfaces will follow responsive design principles to ensure usability on desktop, tablet, and mobile devices. Common elements include navigation bars, tooltips, confirmation popups, and form validation alerts.

### 3.2 Hardware Interfaces

Since HireSense is a web-based platform, there are no specific hardware interfaces beyond standard client machines (laptops, desktops, tablets, smartphones).

The server-side will require:

- Sufficient CPU/GPU (optional for local AI model processing).
- Adequate RAM and storage for handling uploaded resumes and job data.

### 3.3 Software Interfaces

The system will interact with multiple software components:

- **Database:**
  - MySQL to store user data, job listings, parsed resume content, and analytics logs.
- **Backend:**
  - Django framework written in Python for routing, authentication, and logic.
  - pdfminer.six for extracting text from PDF resumes.
  - Pre-trained BERT model for semantic job-resume matching.
  - Cosine similarity libraries for vector comparison.

- **Frontend:**
  - HTML, CSS, Bootstrap, and JavaScript for UI development.

Each interface will be defined through REST APIs and Django views, ensuring loose coupling and ease of integration.

### 3.4 Communications Interfaces

The platform will rely on standard communication protocols for web applications:

- **HTTP/HTTPS:** For secure data transfer between client and server.
- **Web Browser Compatibility:** The platform will be accessible on all major modern browsers such as Chrome, Firefox, Safari, and Edge.

All sensitive data transfers (e.g., login credentials, uploaded resumes) will be encrypted using HTTPS. Authentication tokens will be used to maintain secure user sessions.

## 4. System Features

### 4.1 Resume Upload and Parsing

#### 4.1.1 Description and Priority:

This is a **high-priority** feature. Job seekers can upload their resumes in PDF format. The system will extract key information such as name, email, skills, education, and experience using the pdfminer.six library.

#### 4.1.2 Stimulus/Response Sequences:

- User uploads resume → System parses content → Parsed data is saved in the database → Confirmation message is shown.

#### 4.1.3 Functional Requirements:

- REQ-1: The system shall allow resume uploads in PDF format.
- REQ-2: The system shall parse resumes using pdfminer.six.
- REQ-3: Extracted data shall be stored in structured fields in the database.

## 4.2 Job Posting by Recruiters

### 4.2.1 Description and Priority:

A **core high-priority** feature that enables recruiters to add job listings with skill, experience, and location filters.

### 4.2.2 Stimulus/Response Sequences:

- Recruiter clicks “Post Job” → Enters job details → Job is published → Available to job seekers for matching.

### 4.2.3 Functional Requirements:

- REQ-4: The recruiter shall be able to post job descriptions.
- REQ-5: The system shall validate inputs before submission.
- REQ-6: Jobs shall be stored and visible on the recruiter dashboard.

## 4.3 Resume Matching and Scoring

### 4.3.1 Description and Priority:

This is the **core intelligence** of the system. It is **high priority**. Resumes are semantically compared to job descriptions using the BERT model with cosine similarity.

### 4.3.2 Stimulus/Response Sequences:

- New job posted or new resume uploaded → System performs matching → Top-ranked profiles are listed.

### 4.3.3 Functional Requirements:

- REQ-7: The system shall use BERT to convert text into embeddings.
- REQ-8: It shall apply cosine similarity to determine resume-job relevance.
- REQ-9: Top matches shall be ranked and shown to recruiters.

## 4.4 Resume Suggestions and Improvement Tips

### 4.4.1 Description and Priority:

This **medium-priority** feature improves user experience by offering resume enhancement suggestions.

### 4.4.2 Stimulus/Response Sequences:

- Resume parsed → AI reviews structure/content → Suggestions shown on dashboard.

#### 4.4.3 Functional Requirements:

- REQ-10: System shall analyze resume content using rules/ML.
- REQ-11: Suggestions must be shown in a readable, friendly format.

### 4.5 Job Recommendation to Seekers

#### 4.5.1 Description and Priority:

This **medium-to-high-priority** feature recommends jobs based on a user's profile and parsed resume.

#### 4.5.2 Stimulus/Response Sequences:

- Resume parsed → Matching jobs fetched → Displayed on dashboard.

#### 4.5.3 Functional Requirements:

- REQ-12: Jobs must be recommended using skill and experience matching.
- REQ-13: Recommendations must update dynamically when a new job is posted.

### 4.6 Interview Scheduling

#### 4.6.1 Description and Priority:

This **medium-priority** feature allows recruiters to schedule interviews with selected candidates.

#### 4.6.2 Stimulus/Response Sequences:

- Recruiter selects a candidate → Chooses date/time → Notification sent to seeker.

#### 4.6.3 Functional Requirements:

- REQ-14: Recruiters can send interview invites with date and time.
- REQ-15: Job seekers must receive interview notifications via email.

### 4.7 Admin Monitoring and Management

#### 4.7.1 Description and Priority:

This is a **high-priority** feature required to maintain platform integrity.

#### 4.7.2 Stimulus/Response Sequences:

- Admin logs in → Views users, job posts, resumes → Can activate/deactivate access.

#### 4.7.3 Functional Requirements:

- REQ-16: Admin must have access to all user and job data.

- REQ-17: Admin shall manage accounts and moderate content.

## **5. Other Nonfunctional Requirements**

### **5.1 Performance Requirements**

The HireSense platform is designed to provide high performance and responsiveness for its users. It must be capable of handling multiple concurrent users, ideally up to 500 active sessions, without significant delays or system crashes. Resume parsing, job matching, and job recommendations should be completed within 2 to 3 seconds of the user action, ensuring a smooth and efficient user experience. The backend system must be optimized to handle large volumes of resume data and job postings while maintaining consistent performance across the platform. This level of responsiveness is essential to meet the expectations of recruiters and job seekers who rely on the system for real-time interaction and decision-making.

### **5.2 Safety Requirements**

To ensure safety during system usage, the platform implements several safeguards against data loss and malicious content. All resumes uploaded to the system are scanned for viruses or harmful code before processing and storage. The system also logs all user activities, including login attempts and resume uploads, to detect unusual behavior patterns that might indicate security threats. In case of suspicious activity, the system will flag the event for administrative review. Regular backups of the database and file storage are scheduled to prevent data loss in case of unexpected server failures or outages. These safety measures help maintain the integrity and trustworthiness of the platform.

### **5.3 Security Requirements**

Security is a fundamental component of the HireSense architecture. All data exchanges between the client and server are encrypted using HTTPS to prevent interception and tampering. User credentials, including passwords, are stored securely using hashing algorithms such as bcrypt, ensuring that sensitive information remains protected even if the database is compromised. Access to different parts of the system is strictly controlled through role-based authentication, which ensures that users can only access features relevant to their assigned roles (job seeker, recruiter, or admin). Additionally, security logs are maintained to record important actions and support audits in case of a breach or system misuse.

## 5.4 Software Quality Attributes

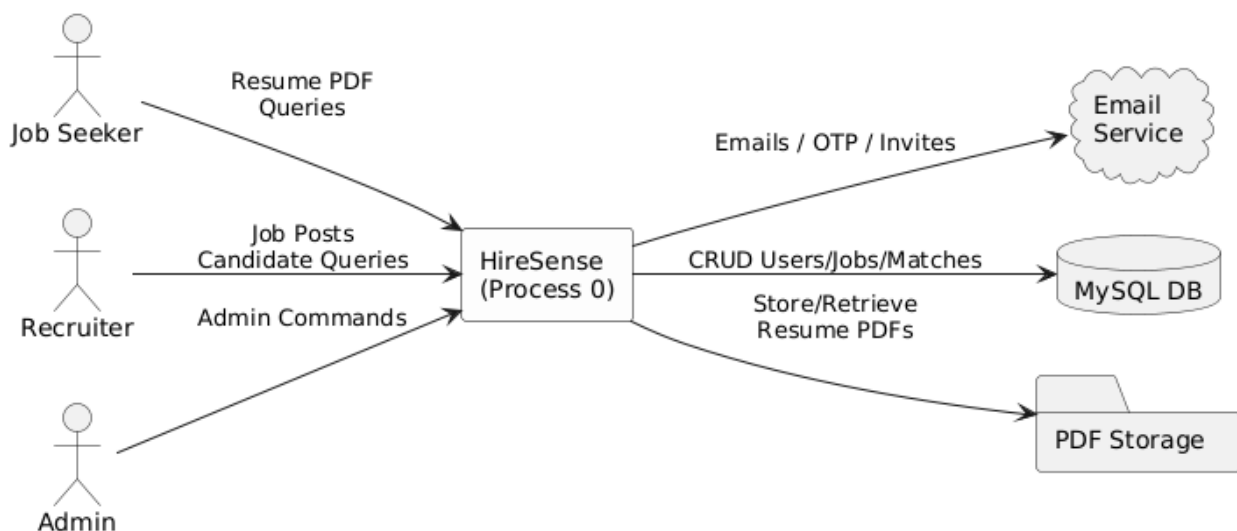
The system is built with a focus on several key software quality attributes, including usability, maintainability, scalability, and reliability. The user interface is designed to be intuitive, minimizing the learning curve for non-technical users such as job seekers and HR personnel. Maintainability is achieved by adhering to modular design principles and Django coding standards, allowing developers to easily update or extend the system in the future. The application is scalable, with infrastructure support to accommodate a growing number of users, resumes, and job listings. Reliability is ensured by implementing fault-tolerant design patterns and regular testing procedures to detect and resolve bugs before deployment.

## 5.5 Business Rules

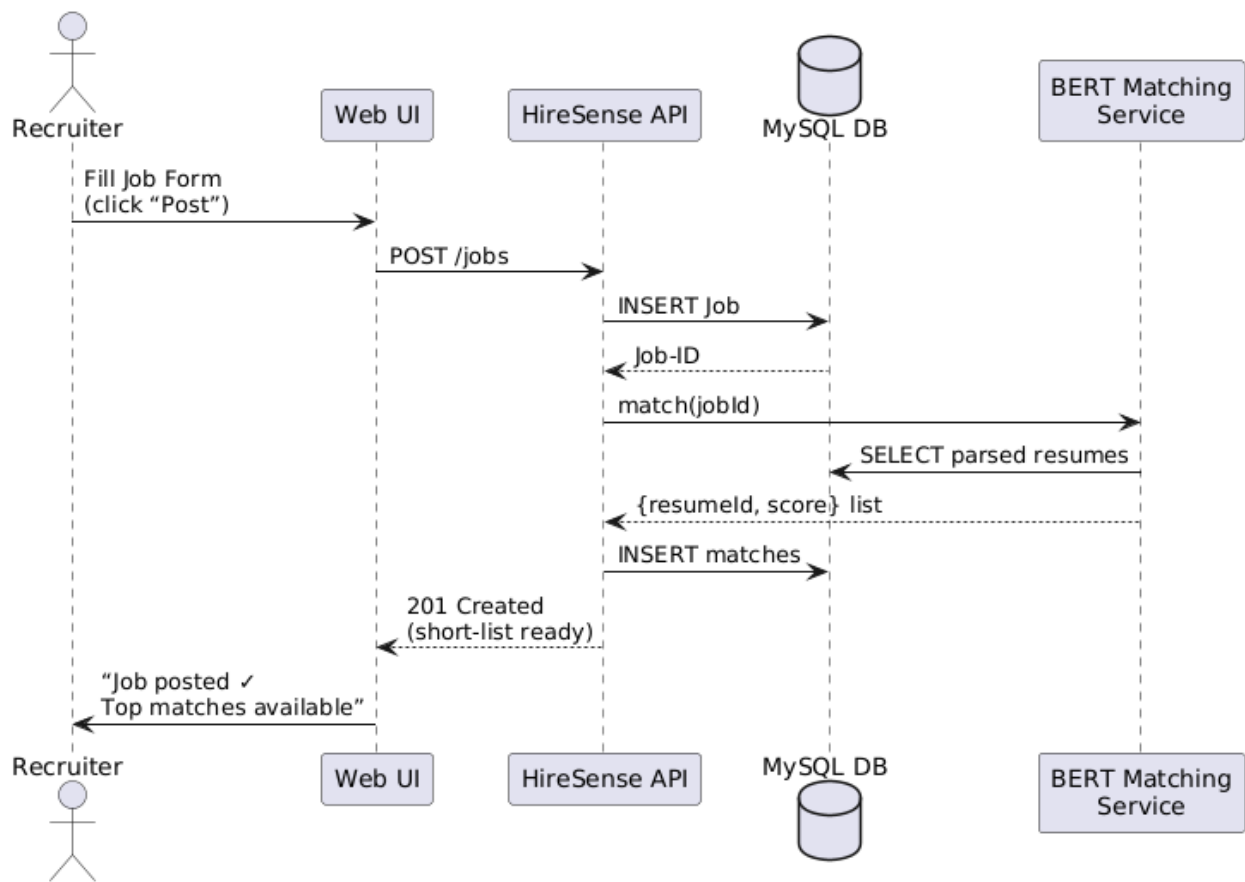
Several business rules are embedded within the functionality of HireSense to ensure fairness, clarity, and efficient use of the platform. Each user account must be unique, and an individual cannot register as both a job seeker and a recruiter using the same email address. Only recruiters who have been verified by the admin are permitted to post job listings, ensuring the legitimacy of job opportunities on the platform. Resume suggestions and matching results are confidential and only accessible to the owner of the resume or the recruiter with matching permissions. Furthermore, all job postings must meet a minimum completeness standard before they are made publicly visible, helping maintain a high-quality job marketplace.

## 6. System Modeling

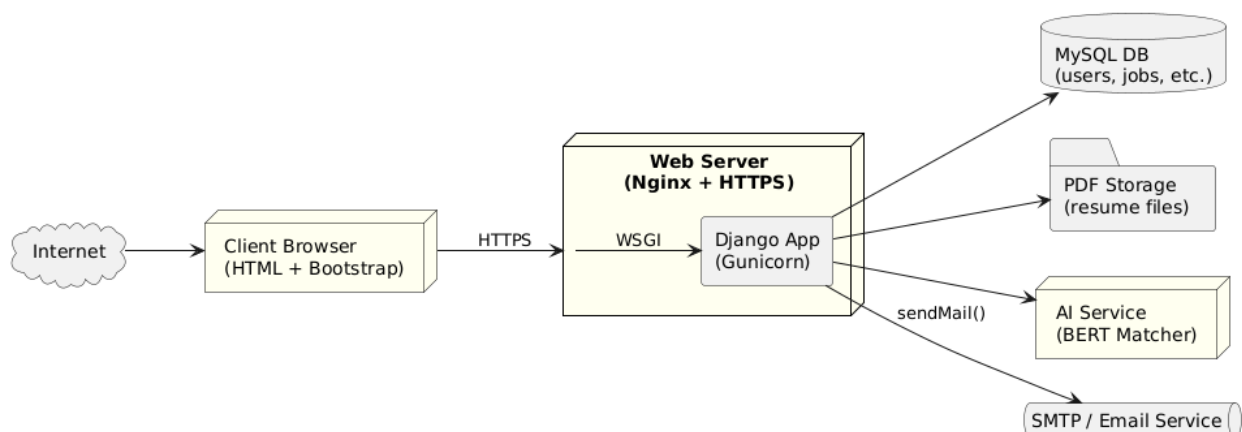
### 6.1 Data Flow Diagram



## 6.2 Sequence Diagram



## 6.3 Architectural Diagram



## References

### LinkedIn Talent Blog (2023)

Blog, L.T. (2023) *LinkedIn*. Available at: <https://www.linkedin.com/business/talent/blog> (Accessed: June 14, 2025).

### pdfminer.six

*Welcome to pdfminer.six's documentation!* (2019).  
Available at: <https://pdfminersix.readthedocs.io/> (Accessed: 14 June 2025).

### BERT (Bidirectional Encoder Representations from Transformers)

Devlin, J. *et al.* (2019) *Bert: Pre-training of deep bidirectional Transformers for language understanding*, *arXiv.org*. Available at: <https://arxiv.org/abs/1810.04805> (Accessed: 14 June 2025).