Technical Documentation

Project: coRecruit Platform

AI-Driven Recruitment Solution

Version: 3.0 (Final)

Date: August 1st, 2025

Prepared by: Team Siberbyte

Table of Contents

1. Introduction

Title: coRecruit Platform

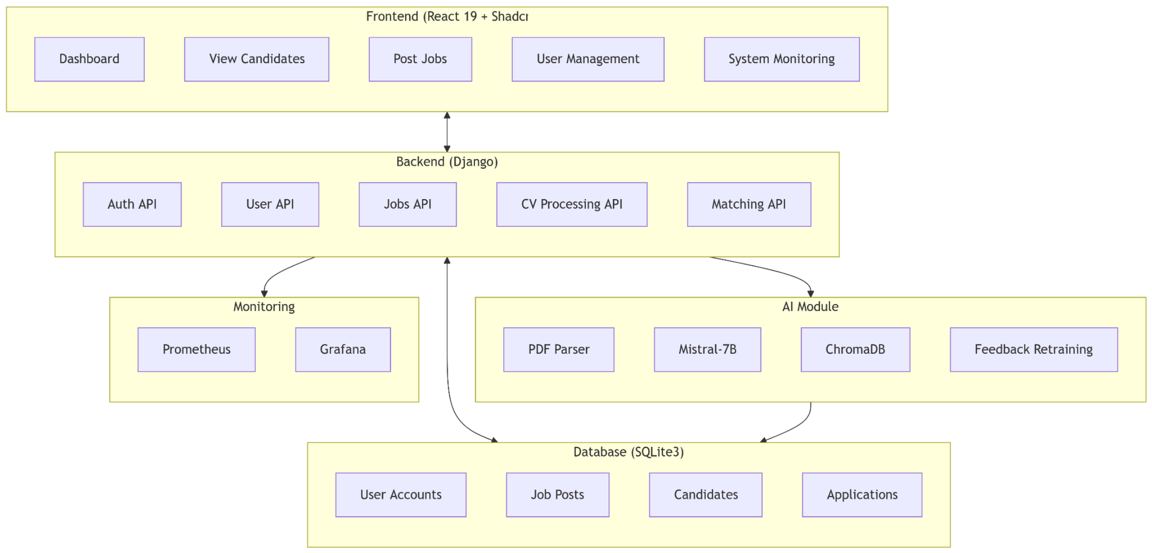
Objective: An offline-capable, AI-powered recruitment platform that automates resume parsing, semantic job matching, and continuously improves through HR feedback-driven model refinement.

Key Features:

- AI Resume Parser (PDF/DOCX/DOC)
- Semantic Job Matching (Skills/Role/Experience)
- HR Feedback System for Model Refinement
- Offline-Capable Processing
- Admin Dashboard with System Monitoring

2. System Overview

2.1 Architecture Diagram



2.2 Core Components

Component	Technology
Frontend	React 19 (JavaScript), Shadcn Radix UI 1.1
Backend	Django, Docker

AI Module	Mistral-7B (GGUF 4-bit), spaCy, ChromaDB
Database	SQLite3
Monitoring	Prometheus + Grafana

3. Functional Specifications

3.1 User Roles & Permissions

Role	Access
HR Manager	Dashboard, View candidates, Post jobs, Give feedback
Recruiter	Dashboard, View Candidates
Basic Admin	User account management, Job Category management
Advanced Admin	User account management, Job Category management, System health monitoring
Full Admin	All access + security config

3.2 Key Workflows

1. Resume Processing:
 - Upload → Field extraction via hybrid method (Python libraries for contact info/name, Mistral-7B for other fields) → Vector embedding

2. Job Matching:
 - Semantic matching (ChromaDB) → Ranked candidate list

3. Feedback Loop:
 - HR reviews matches → Feedback stored in local file system → On next inference, stored feedback converted to embeddings → Embeddings guide future results

4. Technical Specifications

4.1 Frontend

- Framework: React 19 (JavaScript) + Shadcn Radix UI 1.1

Key Pages:

- Dashboard

- View Candidates
- Post Jobs
- User Account Management
- Job Category Management
- System Health Monitoring

4.2 Backend

APIs:

POST /api/login/ - User login

GET /api/useraccounts/ - List all user accounts

GET /api/jobdetails/ - List all job details

POST /api/upload-cv/ - Upload a CV

GET /api/candidates/ - List all candidates

GET /api/jobapplication/ - List all job applications

4.3 AI Module

Pipeline:

1. Parsing: PyMuPDF + spaCy + Mistral-7B
2. Matching: ChromaDB similarity search
3. Feedback Integration: Model guidance

Models:

- Primary: Mistral-7B (4-bit quantized)
- Embedding Model: all-MiniLM-L6-v2

4.4 Scoring Logic

Objective:
To generate a comprehensive candidate match score based on a weighted blend of technical, cultural, experiential, educational, and semantic alignment factors.

Scoring Formula:
The final overall_score for a candidate is computed as:

overall_score = (

```
    technical_score * 0.30 +      # 30% - Technical skills
    cultural_score * 0.20 +       # 20% - Cultural fit/soft skills
    experience_score * 0.25 +     # 25% - Experience match
    education_score * 0.10 +      # 10% - Education match
    ai_enhanced_score * 0.15      # 15% - AI semantic understanding
)
```

- `technical_score`, `cultural_score`, `experience_score`, and `education_score` are derived from resume parsing and profile extraction.
- `ai_enhanced_score` is calculated via semantic similarity between candidate vectors and job description embeddings using ChromaDB and all-MiniLM-L6-v2.

5. Deployment & Infrastructure

Aspect	Solution
Hosting	Docker
CI/CD	GitHub Actions
Monitoring	Prometheus (metrics) + Grafana (dashboard)

6. Data Privacy & Compliance

- **Data Storage:**
  Resume and user data is securely stored using Django's default mechanisms, including hashed credentials, secure session management, and database access controls.
- **Data in Transit:**
  All communication between the client and server is encrypted via **HTTPS**, ensuring secure data transmission.
- **Session & Access Controls:**
  User sessions are securely managed with support for single-login

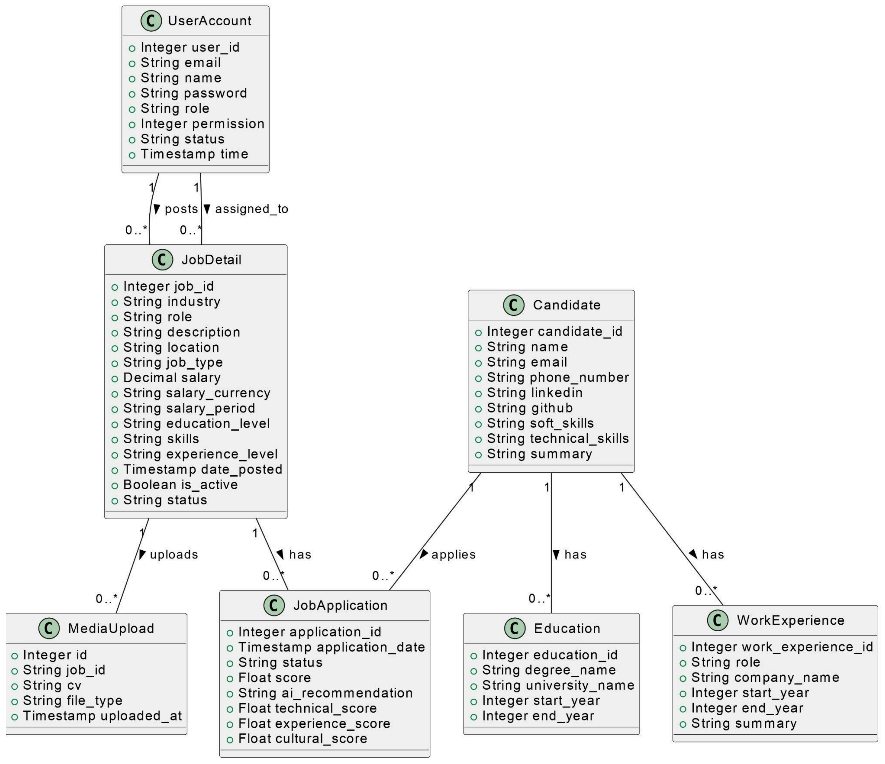
enforcement, session cleanup on logout, and role-based access restrictions across frontend and backend.

7. Appendices

A. Tech Stack Summary

Category	Tools
Frontend	React, Shadcn Radix UI
Backend	Django, SQLite3
AI/ML	Mistral-7B, spaCy, Chroma
DevOps	Docker, Prometheus, Grafana

B. Database Design



C. Database Schema

CREATE TABLE useraccount (

```
  user_id SERIAL PRIMARY KEY,
  email VARCHAR(254) UNIQUE NOT NULL,
  name VARCHAR(128) NOT NULL,
  password VARCHAR(128) NOT NULL,
  role VARCHAR(20) CHECK (role IN ('full_admin', 'basic_admin', 'advanced_admin',
'hr_manager', 'recruiter')),
  permission INTEGER CHECK (permission BETWEEN 1 AND 10),
  status VARCHAR(10) DEFAULT 'Active' CHECK (status IN ('Active', 'Suspended')),
  time TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);
CREATE TABLE jobdetail (
  job_id SERIAL PRIMARY KEY,
  industry VARCHAR(255) NOT NULL,
  role VARCHAR(255) NOT NULL,
  description TEXT NOT NULL,
  location VARCHAR(100) DEFAULT 'Remote',
  job_type VARCHAR(50) CHECK (job_type IN ('Full-time', 'Part-time', 'Contract')),
  salary DECIMAL(10, 2),
  salary_currency VARCHAR(10) DEFAULT 'USD',
  salary_period VARCHAR(20) DEFAULT 'year' CHECK (salary_period IN ('year', 'month')),
  education_level VARCHAR(255) NOT NULL,
  skills TEXT NOT NULL,
  experience_level VARCHAR(255) NOT NULL,
  date_posted TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  is_active BOOLEAN DEFAULT TRUE,
  status VARCHAR(10) DEFAULT 'Active' CHECK (status IN ('Active', 'Suspended')),
  posted_by_id INTEGER REFERENCES useraccount(user_id),
  assigned_to_id INTEGER REFERENCES useraccount(user_id)
);
CREATE TABLE candidate (
  candidate_id SERIAL PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  email VARCHAR(254) UNIQUE NOT NULL,
  phone_number VARCHAR(20),
  linkedin VARCHAR(200),
  github VARCHAR(200),
  soft_skills TEXT NOT NULL,
  technical_skills TEXT NOT NULL,
  summary TEXT NOT NULL
);
CREATE TABLE workexperience (
  work_experience_id SERIAL PRIMARY KEY,
  candidate_id INTEGER REFERENCES candidate(candidate_id) ON DELETE CASCADE,
  role VARCHAR(255) NOT NULL,
  company_name VARCHAR(255) NOT NULL,
  start_year INTEGER NOT NULL,
  end_year INTEGER,
  summary TEXT NOT NULL
);
CREATE TABLE education (
  education_id SERIAL PRIMARY KEY,
  candidate_id INTEGER REFERENCES candidate(candidate_id) ON DELETE CASCADE,
  degree_name VARCHAR(255) NOT NULL,
  university_name VARCHAR(255) NOT NULL,
  start_year INTEGER NOT NULL,
  end_year INTEGER
);
CREATE TABLE jobapplication (
  application_id SERIAL PRIMARY KEY,
  job_id INTEGER REFERENCES jobdetail(job_id) ON DELETE CASCADE,
  candidate_id INTEGER REFERENCES candidate(candidate_id) ON DELETE CASCADE,
  application_date TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  status VARCHAR(20) DEFAULT 'not_selected' CHECK (status IN (
    'rejected', 'not_selected', 'initial_screening', 'final_screening', 'rejected_by_hr'
  )),
  score FLOAT,
  ai_recommendation VARCHAR(255),
  technical_score FLOAT,
  experience_score FLOAT,
  cultural_score FLOAT,
  UNIQUE (job_id, candidate_id)
);
CREATE TABLE mediaupload (
  id SERIAL PRIMARY KEY,
  job_id VARCHAR(100) NOT NULL,
  cv TEXT NOT NULL,
  file_type VARCHAR(10) NOT NULL,
  uploaded_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);
```