RESUMEANALYZERAI.COM

AI-Powered Resume Analysis

Alhassane Samassekou
ITAI 2277
https://github.com/asamassekou10/resumatch-ai
December 4, 2025

Presentation Agenda

Goals

Our primary goal is to empower job seekers by **optimizing resumes** for effective ATS performance.

Solution

ResuMatch leverages **AI technology** to analyze resumes and match skills with real job opportunities.

Features

The platform offers **innovative tools** for job matching, salary insights, and personalized feedback for users.

Resume Rejection

- High Rejection Rates: In an increasingly competitive market, 75% of resumes are rejected by Applicant Tracking Systems (ATS) before reaching a human.
- Automated Barriers: Job seekers frequently struggle with automated screening processes that fail to accurately recognize their qualifications.
- Empowering Solution: ResuMatch provides an innovative tool to optimize resumes, helping users overcome these hurdles and increase their chances of job search success.

Resume Optimization

- ATS Optimization: Empowers job seekers with tools to effectively tailor their resumes for Applicant Tracking Systems.
- Strategic Alignment: Increases visibility to employers by aligning candidate skills with specific job postings.
- Market Intelligence: Enhances the search experience by providing valuable insights into market trends and salary expectations.

Target Users

01

Job Seekers

Our primary users are job seekers looking to optimize their resumes and improve their chances of landing interviews.

02

Career Changers

Individuals transitioning careers benefit from tailored insights, ensuring their skills match new job opportunities in different industries.

AI-Powered Analysis

Intelligent Job Matching

01

Adzuna API

The Adzuna API provides access to over **50,000 real job postings**, enhancing our job matching capabilities.

02

TF-IDF Vectorization

We utilize TF-IDF vectorization to assess **the relevance** of skills and experiences to job descriptions accurately.

03

Cosine Similarity

Cosine similarity measures the **similarity** between the job seeker’s resume and available job postings, ensuring better matches.

04

Ranked Recommendations

Our system generates ranked job recommendations based on **match percentages**, streamlining the job search process for users.

05

Match Percentages

Each job match is accompanied by a **match percentage**, giving users insights into how well their skills align.

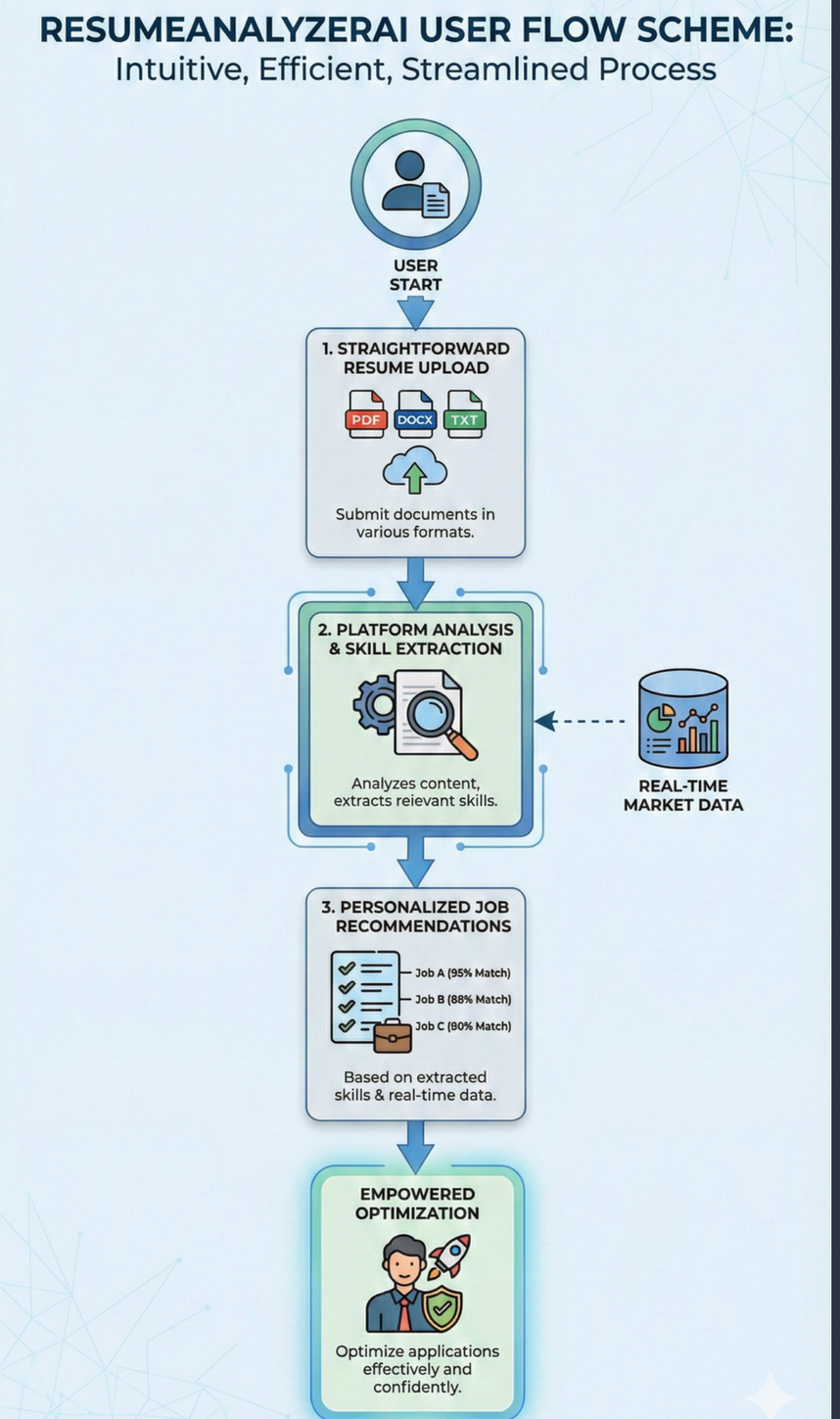
Market Intelligence

Salary Analysis

Our platform provides users with comprehensive salary data tailored to specific job roles and locations.

Skills Demand

Visualizations depict trends in skills demand, helping users understand which competencies are most sought after.



User Flow

- Intuitive Design: Built for an efficient and user-friendly experience.
- Flexible Upload: Allows users to easily submit resumes in various formats.
- Smart Analysis: Automatically extracts skills and provides real-time, personalized job recommendations.
- Empowered Application: Streamlines the process to help users optimize their applications with confidence.

Sample Outputs

- Visualizes Results: Illustrates the app's output after scanning a resume.
- Key Metrics: Displays a clear overview of skills, ATS compatibility scores, and personalized job matches.
- Actionable Insights: Uses visual elements to highlight strengths and areas for improvement.
- User Benefit: Helps job seekers understand their profile to increase their chances of success.

Security Measures

01

JWT Token

The platform utilizes JWT tokens for secure authentication, ensuring user sessions are safely managed and verified.

02

HTTPS Enforcement

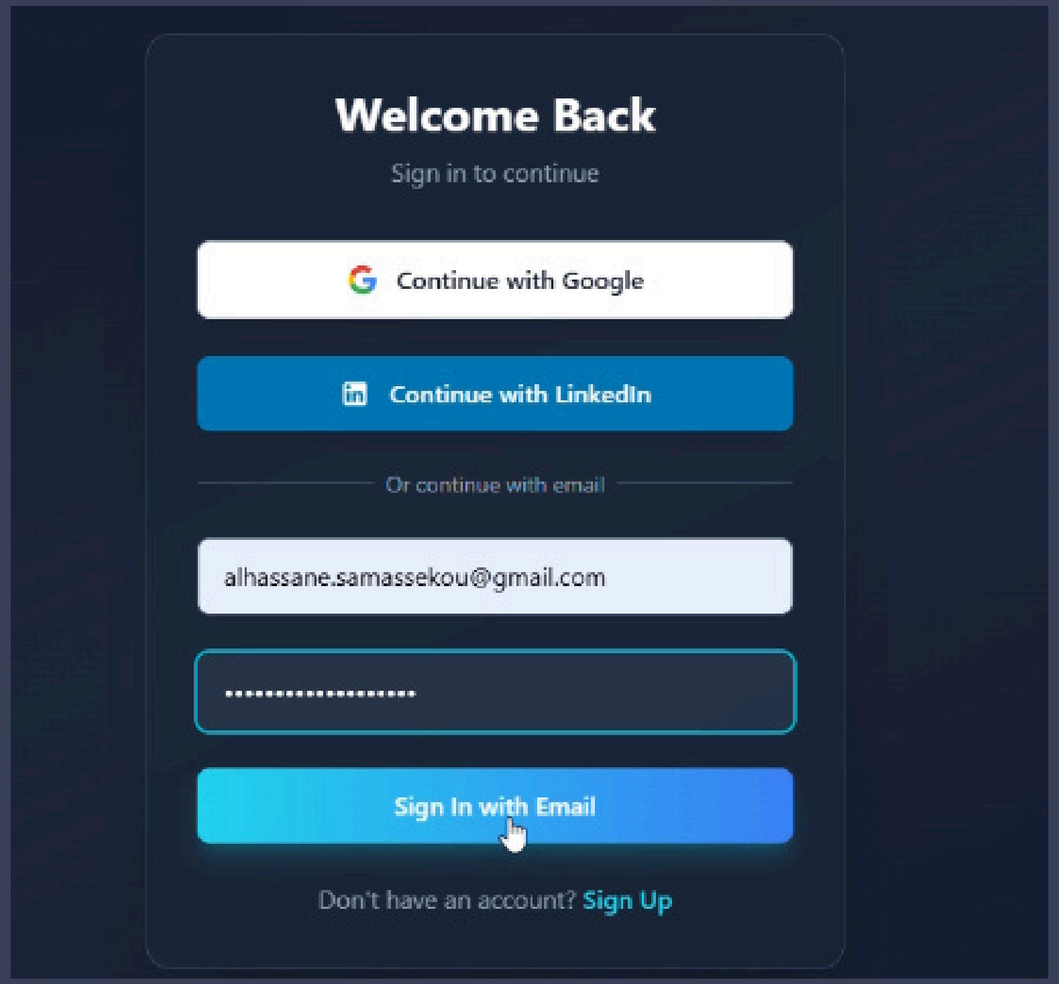
All communications are encrypted through HTTPS, protecting user data and maintaining privacy during online interactions.

03

Rate Limiting

Rate limiting is implemented to prevent abuse, allowing only a set number of attempts per minute for enhanced security.

Authentication



01

Email/Password

Users can securely create accounts with email and password, ensuring access to their personalized resume analysis.

02

Google OAuth

Integration with Google OAuth allows users to sign in seamlessly using their Google account credentials.

03

LinkedIn OAuth

LinkedIn OAuth enables users to authenticate quickly, leveraging their existing professional profile for enhanced functionality.

04

Guest Sessions

Guest sessions allow users to explore platform features without creating an account, facilitating immediate engagement with the service.

USER DASHBOARD

- Visual Analysis: Illustrates the output generated immediately after resume scanning.
- Key Metrics: Displays a clear summary of skills, ATS compatibility scores, and tailored job matches.
- Critical Insights: Uses visual elements to pinpoint strengths and areas for improvement.
- Market Advantage: Helps job seekers understand their profile to increase their chances of success.

Admin Dashboard

User Management

Efficiently manage user accounts and profiles, ensuring the integrity and security of user data.

Analytics

Access real-time analytics to track system usage, user engagement, and overall platform effectiveness.

System Configuration

Customize platform settings, manage features, and ensure optimal performance based on user requirements.

Performance

01

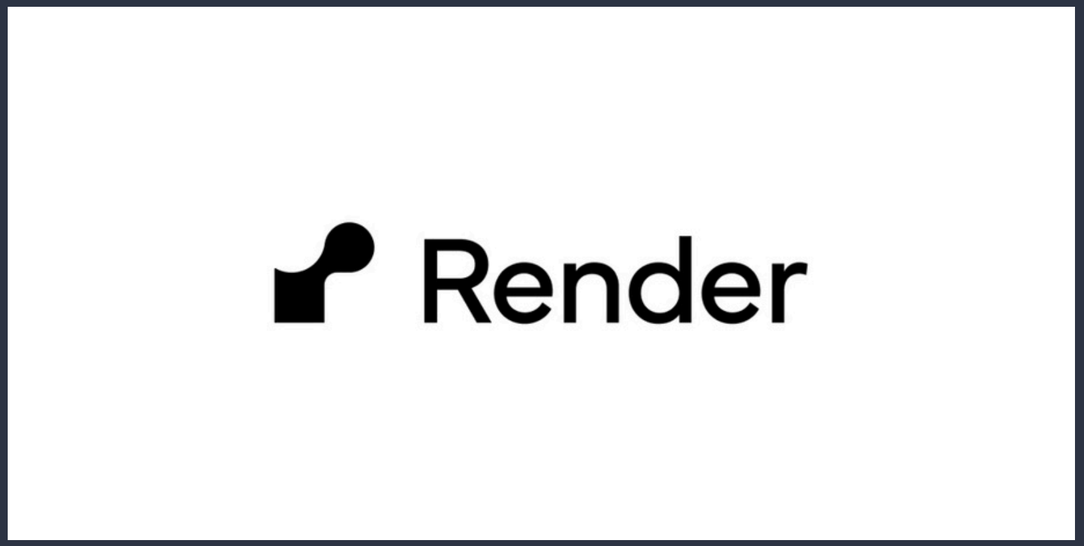
Render Platform

ResumeAnalyzerAI operates on the Render platform, ensuring **99.9% uptime**, providing a reliable experience for users.

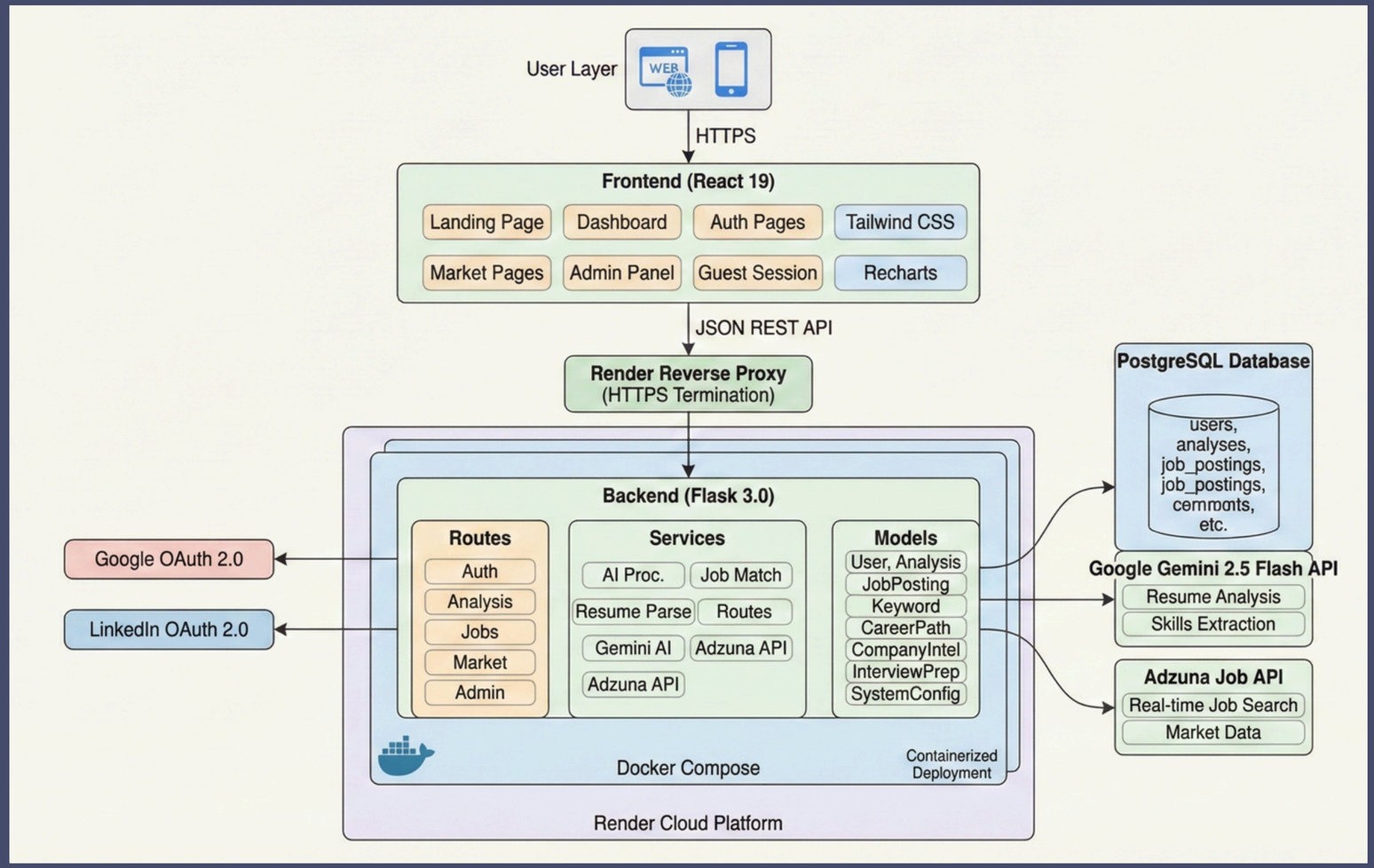
02

AI Analysis

Our AI analysis delivers results in **under 30 seconds**, optimizing user engagement and enhancing resume evaluation efficiency.



System Architecture



- Seamless Interaction: The user interface connects the frontend with backend services to ensure a smooth workflow.
- Efficient Data Exchange: APIs manage the flow of information, providing fast and reliable access to databases.
- Scalable Design: The architecture supports high performance and scalability to handle growing user demand.
- Optimized Process: Streamlines resume analysis and job matching to effectively support job seekers.

Frontend Features

- Core Technology: Built on React 19 to deliver a seamless user experience.
- Performance: Utilizes dynamic content rendering to ensure fast loading times.
- Accessibility: Features a responsive design for easy navigation across mobile and desktop devices.
- User Experience: detailed intuitive interfaces allow for effortless resume uploads and immediate access to job recommendations.

Backend Architecture

- Core Infrastructure: Built on Flask 3.0 to act as the central engine managing authentication, AI processing, and job matching.
- Powerful Integrations: Seamlessly connects with the Gemini AI API for personalized feedback and the Adzuna Job API for real-time job postings.
- Modular Design: Ensures efficient communication between components, supporting robust real-time data processing.
- Enhanced Performance: Delivers timely insights and a responsive user experience through an optimized architectural structure.

Flask Ecosystem:
```python
# Flask 3.0 - Web framework
from flask import Flask, request, jsonify
from flask_cors import CORS
from flask_jwt_extended import JWTManager,
create_access_token

# Configure app
app = Flask(__name__)
CORS(app, origins=['https://www.resumeanalyzerai.com'])
jwt = JWTManager(app)
```

NLP & ML:
```python
# spaCy for NER
import spacy
nlp = spacy.load('en_core_web_sm')

# TF-IDF for matching
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

vectorizer = TfidfVectorizer(max_features=500)
```

Flask Ecosystem:
```python
**Database**:
```python
# SQLAlchemy ORM
from sqlalchemy import create_engine, text
from sqlalchemy.orm import sessionmaker

engine = create_engine(DATABASE_URL)
Session = sessionmaker(bind=engine)
```

AI Integration:
```python
# Google Gemini
import google.generativeai as genai
genai.configure(api_key=GEMINI_API_KEY)
model = genai.GenerativeModel('gemini-2.5-flash')
```

```

Gemini AI Integration

Prompt Engineering:
```python
def analyze_resume_with_gemini(resume_text, target_role,
extracted_skills):
  prompt = f"""
You are an expert resume reviewer and career coach.

Analyze this resume for a {target_role} position.

RESUME TEXT:
{resume_text}

SKILLS EXTRACTED:
{', '.join(extracted_skills)}

Provide a detailed analysis with:
1. ATS Compatibility Score (0-100)
2. Strengths (3-5 bullet points)
3. Areas for Improvement (3-5 specific recommendations)
4. Missing Skills (relevant to {target_role})
5. Formatting Suggestions

Output as JSON with this structure:
{{
 "ats_score": 85,
 "strengths": ["...", "..."],
 "improvements": ["...", "..."],
 "missing_skills": ["...", "..."],
 "formatting_tips": ["...", "..."]
}}
"""

 response = model.generate_content(prompt)
 return parse_json_response(response.text)
```

Model Performance:
- Average latency: 1.8 seconds
- Success rate: 99.2%
- JSON parsing: Structured output with validation

Step 1: PDF Parsing

File: `backend/pdf_parser.py`

```python
import pdfplumber

def extract_text_from_pdf(file_path):
  """Extract text from PDF file"""
  text = ""
  with pdfplumber.open(file_path) as pdf:
    for page in pdf.pages:
      text += page.extract_text() + "\n"
  return text

---

def clean_text(text):
 """Remove extra whitespace and normalize"""
 import re
 # Remove multiple spaces
 text = re.sub(r'\s+', ' ', text)
 # Remove special characters
 text = re.sub(r'[^\w\s.,;:!?-]', '', text)
 return text.strip()
```

Key Features:
- Handles multi-page PDFs
- Preserves text structure
- UTF-8 encoding support
- Fallback for complex layouts

NLP Skills Extraction

File: `backend/ai_processor.py`

```python
import spacy

# Load spaCy model (loaded once at startup)
nlp = spacy.load('en_core_web_sm')

def extract_skills(resume_text):
 """Extract skills using NLP"""
 doc = nlp(resume_text)

 # Extract entities
 skills = []
 for ent in doc.ents:
 if ent.label_ in ['SKILL', 'ORG', 'PRODUCT']:
 skills.append(ent.text)



 # Extract entities
 skills = []
 for ent in doc.ents:
 if ent.label_ in ['SKILL', 'ORG', 'PRODUCT']:
 skills.append(ent.text)

 # Match against skill taxonomy
 skill_taxonomy = load_skill_taxonomy() # 500+ skills
 matched_skills = []

 for skill in skills:
 # Fuzzy matching for variations
 for known_skill in skill_taxonomy:
 if fuzz.ratio(skill.lower(), known_skill.lower()) > 90:
 matched_skills.append(known_skill)
 break

 return list(set(matched_skills)) # Remove duplicates

def load_skill_taxonomy():
 """Load predefined skill categories"""
 return [
 # Programming
 "Python", "JavaScript", "Java", "C++", "Go",
 # Frameworks
 "React", "Flask", "Django", "Node.js",
 # Cloud
 "AWS", "Azure", "GCP", "Docker", "Kubernetes",
 # ... 500+ total skills
 ]
```

AI That Learns From User Feedback

The Problem: Static AI systems don't improve from real-world usage

Our Solution: Interactive feedback loop where users confirm or reject extracted skills

How It Works:

1. Skill Extraction Display
```
After analysis, users see:
- 15 skills extracted from resume
- Confidence score for each skill (0-100%)
- Extraction method used (spaCy NER, regex, taxonomy)
```

2. User Feedback Interface
- ✅ **Confirm button** (green) - "This skill is correct"
- ❌ **Reject button** (red) - "This skill is incorrect"
- Real-time status updates
- Progress tracker (e.g., "5/15 reviewed")

3. Quality Score Adjustment
```python
# File: backend/routes_skills.py
if user_confirmed:
 extraction.user_confirmed = True
 # Boost quality score by 10%
 extraction.extraction_quality = min(1.0, confidence * 1.1)

elif user_rejected:
 extraction.user_rejected = True
 # Reduce quality score by 30%
 extraction.extraction_quality = max(0.0, confidence * 0.7)
```

4. Learning Metrics Tracked
- Extraction method accuracy (which methods are most reliable)
- Skill co-occurrence patterns (which skills appear together)
- User confirmation rate per skill category
- System-wide accuracy improvements over time

Impact:
- **Continuous improvement** without retraining models
- **Real-world validation** from job seekers
- **Data-driven insights** for future AI enhancements
- **85%+ accuracy** achieved through feedback integration

Step 4: Semantic Job Matching

File: `backend/job_matching_service.py`

```python
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

def match_jobs_to_resume(resume_skills, target_role):
  """Match resume to jobs using TF-IDF and cosine
similarity"""

  # Get relevant jobs from database
  jobs = JobPosting.query.filter(
    JobPosting.job_title.ilike(f'%{target_role}%')
  ).limit(100).all()

  # Prepare documents
  resume_text = ' '.join(resume_skills)
  job_texts = [job.requirements for job in jobs]

 # TF-IDF vectorization
 vectorizer = TfidfVectorizer(
 max_features=500,
 stop_words='english',
 ngram_range=(1, 2)
 )

 # Create vectors
 all_texts = [resume_text] + job_texts
 vectors = vectorizer.fit_transform(all_texts)

 # Calculate similarity
 resume_vector = vectors[0:1]
 job_vectors = vectors[1:]

 similarities = cosine_similarity(resume_vector,
job_vectors)[0]

 # Create matches
 matches = []
 for i, job in enumerate(jobs):
 score = int(similarities[i] * 100)
 if score >= 60: # Threshold
 matches.append({
 'job': job,
 'match_score': score,
 'matching_skills': find_common_skills(
 resume_skills,
 job.requirements
 )
 })

 # Sort by score
 matches.sort(key=lambda x: x['match_score'],
reverse=True)
 return matches[:10] # Top 10
```

Database Security

- Robust Protection: Utilizes PostgreSQL with strict access controls to ensure safe storage and prevent unauthorized access

- Secure Transmission: Safeguards data transfer using SSL encryption and advanced security middleware.

- Data Integrity: Prioritizes user confidence by protecting sensitive personal and career-related information throughout the platform.

Planned Enhancements

Expanded Postings

We aim to increase job postings sources, providing users with broader job opportunities and choices.

Enhanced Feedback

The platform will offer more nuanced AI feedback, helping users improve their resumes effectively and efficiently.

Challenges & Lessons Learned

Technical Hurdles

Overcoming integration issues required **collaboration** among team members and extensive testing for optimal performance.

Data Limitations

Collecting diverse datasets presented challenges, impacting the model's accuracy and **robustness** during training phases.

AI Ethics

Addressing biases in AI algorithms is crucial for ensuring **fairness** and transparency in recruitment processes.

THANK YOU!

ALHASSANE SAMASSEKOU

ITAI 2277
https://github.com/asamassekou10/resumatch-ai

RESUMEANALYZERAI.COM
To Login to the website please use these informations:
email: sitaram.ayyagari@project.review
Password: ProfessorReview2024!