

# BLACKBOXAI

Generated on: 21/09/2025 08:39:09

## User

create an 20 minutes speach of the project

git hub link <https://github.com/Naveenkm007/Automated-Resume-Relevance-Check-System.git>

they gave description

**Theme 2 - Automated Resume Relevance Check System Problem Statement** At Innomatics Research Labs, resume evaluation is currently manual, inconsistent, and time-consuming. Every week, the placement team across Hyderabad, Bangalore, Pune, and Delhi NCR receives 18–20 job requirements, with each posting attracting thousands of applications. Currently, recruiters and mentors manually review resumes, matching them against job descriptions (JD). This leads to: Delays in shortlisting candidates. Inconsistent judgments, as evaluators may interpret role requirements differently. High workload for placement staff, reducing their ability to focus on interview prep and student guidance. With hiring companies expecting fast and high-quality shortlists, there is a pressing need for an automated system that can scale, be consistent, and provide actionable feedback to students.

**Objective** The Automated Resume Relevance Check System will: Automate resume evaluation against job requirements at scale. Generate a Relevance Score (0–100) for each resume per job role. Highlight gaps such as missing skills, certifications, or projects. Provide a fit verdict (High / Medium / Low suitability) to recruiters. Offer personalized improvement feedback to students. Store evaluations in a web-based dashboard accessible to the placement team. This system should be robust, scalable, and flexible enough to handle thousands of resumes weekly.

**Sample Data** Click here to download sample data.

**Proposed Solution** We propose building an AI-powered resume evaluation engine that combines rule-based checks with LLM-based semantic understanding. The system will: Accept resumes (PDF/DOCX) uploaded by students. Accept job descriptions uploaded by the placement team. Use text extraction + embeddings to compare resume content with job descriptions. Run hybrid scoring: Hard match (keywords, skills, education) Soft match (semantic fit via embeddings + LLM reasoning) Output a Relevance Score, Missing Elements, and Verdict. Store results for the placement team in a searchable web application dashboard. This approach ensures both speed (hard checks) and contextual understanding (LLM-powered checks).

**Workflow**

- Job Requirement Upload** - Placement team uploads job description (JD).
- Resume Upload** - Students upload resumes while applying.
- Resume Parsing** Extract raw text from PDF/DOCX. Standardize formats (remove headers/footers, normalize sections).
- JD Parsing** Extract role title, must-have skills, good-to-have skills, qualifications.
- Relevance Analysis Step 1: Hard Match** – keyword & skill

Generated by BlackBox AI

check (exact and fuzzy matches). Step 2: Semantic Match – embedding similarity between resume and JD using LLMs. Step 3: Scoring & Verdict – Weighted scoring formula for final score. Output Generation Relevance Score (0–100). Missing Skills/Projects/Certifications. Verdict (High / Medium / Low suitability). Suggestions for student improvement. Storage & Access Results stored in the database. The placement team can search/filter resumes by job role, score, and location. Web Application Placement team dashboard: upload JD, see shortlisted resumes. Tech Stack (Core Resume Parsing, AI Framework and Scoring Mechanism) Python – primary programming language. PyMuPDF / pdfplumber – extract text from PDFs. python-docx / docx2txt – extract text from DOCX. spaCy / NLTK – entity extraction, text normalization. LangChain – orchestration of LLM workflows. LangGraph – structured stateful pipelines for resume–JD analysis. LangSmith – observability, testing, and debugging of LLM chains. Vector Store (Chroma / FAISS / Pinecone) – for embeddings and semantic search. LLM Models – OpenAI GPT / Gemini / Claude / HuggingFace models for semantic matching & feedback generation. Keyword Matching – TF-IDF, BM25, fuzzy matching. Semantic Matching – embeddings + cosine similarity. Weighted Score – combine hard and soft matches into a final score. Tech Stack (Web Application) - You can choose other tech stack for FE Flask / FastAPI – Backend APIs to process uploads, run OMR evaluation, and serve results. Streamlit (MVP) – Frontend for evaluators (upload, dashboard, review). SQLite / PostgreSQL – Database for storing results, metadata, and audit logs.

## BLACKBOXAI

Certainly! Below is a structured 20-minute speech for presenting the Automated Resume Relevance Check System project. You can adjust the pacing and add personal touches as needed.

---

## 20-Minute Speech: Automated Resume Relevance Check System

---

### Introduction (2 minutes)

Good [morning/afternoon], everyone.

Today, I am excited to present the Automated Resume Relevance Check System, a cutting-edge AI-powered platform designed to revolutionize how resumes are evaluated against job descriptions.

At Innomatics Research Labs, the placement teams across multiple cities face a huge challenge: every week, they receive thousands of applications for 18 to 20 job postings.

Currently, the resume screening process is manual, inconsistent, and time-consuming, leading to delays and uneven candidate shortlisting.

Our system aims to solve these problems by automating resume evaluation, ensuring consistency, scalability, and actionable feedback for both recruiters and students.

---

### **Problem Statement (3 minutes)**

Let's first understand the core challenges:

- **Manual Resume Screening:** Recruiters and mentors manually review thousands of resumes, which is slow and prone to human error.
- **Inconsistent Judgments:** Different evaluators may interpret job requirements differently, leading to inconsistent shortlists.
- **High Workload:** Placement staff spend excessive time on screening, reducing their ability to focus on interview preparation and mentoring.
- **Demand for Speed and Quality:** Hiring companies expect fast, high-quality shortlists, which is difficult to achieve manually.

These challenges create bottlenecks in the hiring pipeline and impact the overall placement success.

---

### **Project Objectives (2 minutes)**

Our Automated Resume Relevance Check System is designed to:

- Automate resume evaluation at scale against job requirements.
- Generate a Relevance Score from 0 to 100 for each resume per job role.
- Highlight gaps such as missing skills, certifications, or projects.
- Provide a clear fit verdict: High, Medium, or Low suitability.
- Offer personalized improvement feedback to students.
- Store all evaluations in a web-based dashboard accessible to the placement team.

The system is built to be robust, scalable, and flexible enough to handle thousands of resumes weekly.

---

### **Proposed Solution Overview (3 minutes)**

---

Generated by BlackBox AI

Our solution combines traditional rule-based checks with advanced AI techniques:

- Resume and Job Description Upload: Students upload resumes in PDF or DOCX format; placement teams upload job descriptions.
- Text Extraction and Parsing: We extract raw text from resumes and job descriptions, normalize formats, and identify key sections.
- Hybrid Scoring Approach:
  - Hard Match: Keyword and skill matching using exact and fuzzy logic.
  - Semantic Match: Contextual similarity using embeddings and large language models (LLMs).
- Output: The system produces a relevance score, missing elements, a suitability verdict, and personalized feedback.
- Storage and Access: Results are stored in a database and accessible via a searchable web dashboard.

This hybrid approach ensures both speed and deep contextual understanding.

---

### **Technical Workflow (4 minutes)**

Let me walk you through the detailed workflow:

- 1. Job Requirement Upload: Placement team uploads job descriptions, which are parsed to extract role titles, must-have and good-to-have skills, and qualifications.
- 2. Resume Upload: Students upload their resumes while applying.
- 3. Resume Parsing:
  - Extract text from PDF/DOCX using libraries like PyMuPDF and python-docx.
  - Normalize text by removing headers, footers, and standardizing sections.
  - Use NLP tools like spaCy and NLTK for Named Entity Recognition (NER) to extract names, skills, education, and experience.
- 4. Relevance Analysis:
  - Hard Match: Check for exact and fuzzy keyword matches for skills, education, and certifications.
  - Semantic Match: Use sentence transformers and LLMs to compute embedding similarity between resume content and job description.
  - Combine scores with weighted formula (default 60% hard match, 40% semantic match).
- 5. Output Generation:

- Calculate a final relevance score (0–100).
- Identify missing skills, certifications, or projects.
- Provide a verdict: High, Medium, or Low suitability.
- Generate personalized feedback using GPT-powered LLMs to help students improve.

- 6. Storage & Access:

- Store all data in PostgreSQL with JSON fields.
- Placement team accesses results via a Streamlit dashboard with filtering and export capabilities.

---

### Key Features & Benefits (3 minutes)

- AI-Powered Hybrid Scoring: Combines deterministic keyword matching with semantic understanding for accurate evaluation.
- Personalized Feedback: Students receive actionable suggestions to improve their resumes.
- Scalable Architecture: Built with FastAPI, Celery workers, Redis, and PostgreSQL for high throughput.
- User -Friendly Dashboard: Placement teams can upload JDs, view shortlisted candidates, filter by score, and export data.
- Security & Reliability: Input validation, rate limiting, authentication, and monitoring ensure production readiness.
- Open Source & Extensible: Modular design allows easy integration of new models and features.

---

### Technology Stack (3 minutes)

- Core Resume Parsing & AI:

- Python as the primary language.
- Text extraction: PyMuPDF, pdfplumber, python-docx.
- NLP: spaCy, NLTK for entity extraction and normalization.
- Semantic matching: Sentence Transformers, OpenAI GPT, LangChain for LLM orchestration.
- Vector stores: Chroma, FAISS, or Pinecone for embedding search.

- Backend & APIs:

- FastAPI for RESTful API services.
- Celery with Redis for asynchronous background processing.
- PostgreSQL for robust data storage.

- Frontend:

- Streamlit MVP dashboard for placement teams. - React components with Tailwind CSS planned for future enhancements.

- DevOps & Deployment:

- Docker and Kubernetes for containerized, scalable deployment. - CI/CD pipelines with GitHub Actions for automated testing and deployment.

---

### **Demonstration & Usage (2 minutes)**

- Upload job descriptions via the dashboard.
- Students upload resumes which are parsed and evaluated asynchronously.
- Placement team views real-time processing status and relevance scores.
- Filter candidates by score, role, and location.
- Export shortlisted candidates to CSV for further processing.
- View detailed feedback to guide students on resume improvements.

---

### **Challenges & Solutions (2 minutes)**

- Handling Diverse Resume Formats: Used multiple extraction libraries and normalization techniques.
- Balancing Speed and Accuracy: Hybrid scoring ensures fast keyword checks with deep semantic analysis.
- Scalability: Asynchronous task queues and container orchestration handle thousands of resumes weekly.
- Feedback Quality: Leveraged GPT models with prompt engineering for meaningful, actionable suggestions.
- Security: Implemented strict input validation, authentication, and rate limiting.

---

### **Future Enhancements (1 minute)**

- Integrate more advanced LLMs like GPT-4 or Gemini for richer feedback.
- Add multi-language support for resumes and job descriptions.
- Enhance frontend with React for better UX and mobile responsiveness.
- Incorporate interview scheduling and candidate tracking.

- Use advanced vector databases for faster semantic search at scale.

---

## **Conclusion (2 minutes)**

In summary, the Automated Resume Relevance Check System addresses a critical bottleneck in the placement process by automating resume evaluation with AI-powered hybrid scoring and personalized feedback.

It improves consistency, reduces workload, and accelerates candidate shortlisting, benefiting both placement teams and students.

This scalable, production-ready platform is a significant step forward in leveraging AI to enhance recruitment efficiency and quality.

Thank you for your attention. I am happy to take any questions.

---

If you want, I can also help prepare slides or demo scripts for this presentation!