AI Enabled Resume Shortlisting Application for HR Teams

---

1. Project Overview

Recruitment teams often spend significant time manually reviewing resumes to find suitable candidates for a job role. This project aims to simplify and automate the **resume shortlisting process** using **Generative AI and Retrieval Augmented Generation (RAG)**.

The application allows HR users to upload, manage, and analyze resumes and automatically shortlist the most relevant candidates based on a given **job description**.

---

2. Goal of the Application

To build a **Generative AI–powered application** that helps the HR team:

- Manage resumes efficiently
- Search and understand resume content semantically
- Automatically shortlist the **best-matched resumes** for a job description

---

3. Key Features and Functionalities

3.1 Streamlit UI Application

- User-friendly web interface built using **Streamlit**
- No technical knowledge required for HR users
- Supports interactive actions like upload, delete, and shortlist

---

3.2 Upload New Resumes (PDF)

- Upload resumes in **PDF format**
- Extract text content from PDFs

- Store resumes as vector embeddings in **Chroma Vector Store**

3.3 Update Existing Resume

- Replace an existing resume with an updated PDF
- Automatically:
    - Delete old embeddings
    - Generate and store new embeddings
- Ensures the system always uses the latest resume version

3.4 List All Resumes

- Display all stored resumes with:
    - Candidate name / file name
    - Upload date (optional)
- Helps HR track available resumes

3.5 Delete Selected Resume

- Remove a resume from the system
- Deletes:
    - Resume metadata
    - Corresponding vector embeddings from Chroma
- Keeps the database clean and relevant

3.6 Shortlist Resumes for a Job Description

- HR provides a **job description**
- User specifies **number of resumes to shortlist**
- Application:
    - Converts job description into embeddings
    - Performs similarity search in Chroma
    - Returns the most relevant resumes ranked by relevance

---

4. System Architecture (High-Level)

Resume Management

```
Streamlit UI
     |
     v
Resume PDF Upload (New/Updated)
     |
     v
Text Extraction (PDF)
     |
     v
Embedding Generation
     |
     v
Chroma Vector Store
(Store/Replace in db)
```

Resume Shortlisting

```
Streamlit UI
     |
     v
Job Description Input
     |
     v
Embedding Generation
     |
     v
Chroma Vector Store
Similarity Search (RAG)
     |
     v
Shortlisted Resumes
```

5. RAG (Retrieval Augmented Generation) Approach

Why RAG?

- Resumes are **private, domain-specific documents**
- LLMs alone do not know resume content
- RAG ensures responses are grounded in **actual resume data**

How RAG Works in This Project

1. Resume text is embedded and stored in **Chroma**
2. Job description is embedded at runtime
3. Relevant resume chunks are retrieved
4. LLM uses retrieved context to:
    - Rank resumes

    - Justify shortlisting decisions

---

6. Vector Store – Chroma

Why Chroma?

- Lightweight and easy to integrate
- Persistent local storage
- Fast similarity search
- Ideal for mini-projects and prototypes

What is Stored in Chroma?

- Resume text embeddings
- Metadata such as:
    - Resume ID
    - File name
- Resume document (?)

---

7. Technology Stack

Component	**Technology**
UI	Streamlit
Programming Language	Python
LLM	Any compatible LLM
Embeddings	Any Embedding Model
Vector Store	Chroma
PDF Processing	PyPDFLoader

Component	Technology
RAG Framework	LangChain

---

8. Data Flow Explanation

1. HR uploads resume (PDF)
2. PDF text is extracted
3. Text is split into chunks
4. Embeddings are generated
5. Embeddings stored in Chroma
6. Job description is entered
7. Similarity search is performed
8. Top-N resumes are shortlisted

---

9. Assumptions and Constraints

- Resumes are uploaded only in PDF format
- Shortlisting is based on **semantic similarity**, not exact keyword match
- Accuracy depends on resume quality and job description clarity

---

10. Future Enhancements (Optional)

- Resume scoring with explanation
- Skill extraction and tagging
- Candidate ranking dashboard
- Support for DOCX resumes
- Multi-job profile support
- Authentication for HR users

---

11. Learning Outcomes

Through this mini-project, you will understand:

- Practical use of **Generative AI**
- Concept of **RAG architecture**
- Vector databases and embeddings
- Real-world AI application design
- Integration of UI with AI backend

---

12. Conclusion

This project demonstrates how **Generative AI and RAG** can be applied to solve a real-world HR problem. By combining **Streamlit, Chroma, and LLMs**, the application provides an efficient and scalable way to shortlist resumes while maintaining transparency and relevance.