CASE STUDY: AI-Powered HR Insights & Assistant for Acme Corp

Project Background

Client: Acme Corp (Fictional enterprise)

Acme Corp is a global mid-sized enterprise with over 12,000 employees across multiple regions. The company is experiencing **above-average voluntary attrition rates**, especially among younger and mid-level employees. Employee exit interviews point to poor work-life balance, unclear growth paths, and dissatisfaction with management practices.

In parallel, Acme's HR department has reported being **overwhelmed by repetitive employee queries** regarding policies, leave rules, reimbursement processes, and more. This has created bottlenecks, delayed service, and negatively affected employee experience.

As the lead consultant in this engagement, you have been brought in to deliver a **data-driven diagnostic of the attrition issue** and to **prototype a Generative AI-powered HR assistant** that can answer employee queries automatically, freeing up HR capacity.

Problem Statement

Your goal is to solve two interconnected problems for the client:

1. **Why are employees leaving?** – Use HR data to identify the drivers of attrition. The client is looking for actionable insights that can be addressed with policy or structural changes.
2. **How can we improve HR support at scale?** – Many HR queries are routine and policy-related. You've been asked to prototype an internal AI assistant capable of answering these common questions using the client's policy handbook as its knowledge base.

You are expected to combine your **technical skills, product thinking, and communication ability** to deliver a compelling end-to-end solution under tight timelines.

Exercise Objective

You are expected to complete the following tasks:

Part 1: Attrition Analysis (Python/EDA)

- Analyze the dataset of 10,000 employees (employee_attrition.csv)
- Explore attrition patterns by:
    - Department, Job Role, Tenure, Performance Rating
    - Job Satisfaction, Overtime, Work-Life Balance
- Present at least **2–3 actionable insights** with visualizations
- **Your analysis code should be scalable** and written in a way that it can efficiently handle larger datasets (e.g., 50,000+ rows) with minimal changes. This includes using optimized data operations, avoiding hard-coded limits, and ensuring performance remains stable as the data grows.

Part 2: HR Assistant with GenAI (LangChain + RAG)

Build a **Retrieval-Augmented Generation (RAG)** pipeline using the provided HR policy document (acme_hr_policy.txt):

Expected Components:

- Chunk the document using sentence/paragraph-based strategies
- Create vector embeddings for those chunks
- Use a retriever-LLM chain to generate grounded answers

You are encouraged to use tools and frameworks you are most comfortable with to complete this case. Some components of the solution (e.g., language models, embedding services) may use **proprietary APIs** like OpenAI. If so, please manage API keys securely and **do not hardcode credentials in your submission**.

Ensure your solution is **scalable and efficient** enough to handle large documents (50+ pages or 100+ chunks) without breaking or slowing down significantly.

Note: You are not required to use paid or proprietary tools. If you prefer, feel free to use open-source alternatives.

Part 3: Web App Prototype

Build a lightweight, user-friendly prototype with two tabs:

1. **HR Dashboard**
    - Visualize insights from your data analysis
    - Highlight major attrition trends with summary commentary
2. **Ask HR Assistant**
    - Interface for employees to input a question
    - Output should be the AI-generated answer (via your RAG pipeline)
    - Include basic UI/UX considerations (input box, loading message, etc.)

You may use:

- Streamlit (preferred and open source)
- Any equivalent low-code Python framework

Part 4: Cloud Deployment – Architectural Overview (Bonus)

You are **not expected to deploy the app**, but must outline **how you would** do it in a real-world project. In your documentation, include:

A step-by-step explanation of:

- What parts of the solution would be containerized (e.g., Streamlit app + LangChain backend)
- How vector DB and embeddings would be hosted (e.g., Chroma on Azure Blob or managed DB)
- How secrets like API keys would be handled (e.g., using Azure Key Vault or environment variables)
- Deployment method (e.g., Docker → Azure App Service / Streamlit Cloud / AWS Fargate)

A basic architecture diagram (hand-drawn or digital) is encouraged.

Submission Guidelines

Please submit the following as a structured folder or GitHub repository:

1. Code Repository

- notebooks/ or eda/: Your EDA and visualizations
- app/: Streamlit or equivalent app (app.py)
- chatbot/: LangChain pipeline or GenAI components
- requirements.txt: Clearly listing all packages used

2. Working Web App

- A working Streamlit app (app.py)
- App should ideally demonstrate both Q&A assistant (mandatorily) and Insights (If possible)
- App should run locally with minimal setup.

3. Final Report / Slides

Either a concise slide deck (7-10 slides) or a short-written report (~3–5 pages) including:

- Executive Summary
- Attrition Findings & Visuals
- Chatbot Architecture & Examples
- Cloud Deployment Plan
- Assumptions & Recommendations

4. Supporting Documents

- employee_attrition.csv (already provided)
- acme_hr_policy.txt (already provided)
- Any additional mock data or documents created

What You Will be Assessed On

Competency	What We're Looking For
Technical Capability	Structured Python code, effective LangChain/RAG design, basic data science & modeling skills
Tool Familiarity	Smart use of open source libraries; secure and responsible use of any paid APIs
Problem Solving	Ability to independently plan and structure the solution under time pressure
User Experience	Clean, intuitive, minimal UI/UX (does not need to be fancy, but should be functional)
Communication	Well-structured documentation or slides that clearly explain your decisions and insights
Consulting Mindset	Recommendations grounded in business impact, not just technical feasibility
Cloud Understanding	Realistic deployment plan showing awareness of scalability, security, and integration factors