# Project Report: AI-Powered HR Insights & Assistant for Acme Corp

**Prepared for:** Acme Corp Leadership

**Date:** November 29, 2025

**Version:** 1.0

Contentsssss

[Project Report: AI-Powered HR Insights & Assistant for Acme Corp 1](#_Toc215477046)

[1. Executive Summary 2](#_Toc215477047)

[2. Problem Statement Analysis 2](#_Toc215477048)

[2.1. The Attrition Challenge 2](#_Toc215477049)

[2.2. The HR Efficiency Challenge 2](#_Toc215477050)

[3. Solution Architecture & Design Principles 2](#_Toc215477051)

[3.1. Tech Stack 3](#_Toc215477052)

[3.2. Configuration Management (No Hard-Coding) 3](#_Toc215477053)

[4. Part 1: Attrition Analytics (Diagnostic & Predictive) 4](#_Toc215477054)

[4.1. Methodology & Scalability Features 4](#_Toc215477055)

[4.2. Dynamic Data Updates 4](#_Toc215477056)

[4.3. Key Insights Capabilities 4](#_Toc215477057)

[4.4. Visualizations 5](#_Toc215477058)

[5. Part 2: AI-Powered HR Assistant (The "How") 7](#_Toc215477059)

[5.1. RAG Pipeline Design 7](#_Toc215477060)

[5.2. Performance Optimizations 7](#_Toc215477061)

[6. Part 3: Web Application Prototype 7](#_Toc215477062)

[7. Part 4: Cloud Deployment Strategy 8](#_Toc215477063)

[7.1. Containerization 8](#_Toc215477064)

[7.2. Cloud Architecture (Azure/AWS) 8](#_Toc215477065)

[7.3. CI/CD Pipeline 9](#_Toc215477066)

[8. Strategic Recommendations 9](#_Toc215477067)

[9. Conclusion 9](#_Toc215477068)

[10. Appendix: Project Artifacts 9](#_Toc215477069)

## 1. Executive Summary

Acme Corp is currently facing two critical challenges: an above-average voluntary attrition rate among young and mid-level employees, and an overburdened HR department struggling with repetitive policy queries.

To address these issues, we have developed a dual-pronged solution:

1. **Enterprise Attrition Analytics Platform:** A data-driven diagnostic tool that identifies the root causes of employee turnover using machine learning and exploratory data analysis (EDA).
2. **AI-Powered HR Assistant:** A Generative AI chatbot leveraging Retrieval-Augmented Generation (RAG) to automate policy support and provide instant, accurate answers to employee queries.

This report outlines the technical approach, key insights capabilities, architecture, and a scalable deployment strategy for the solution.

## 2. Problem Statement Analysis

### 2.1. The Attrition Challenge

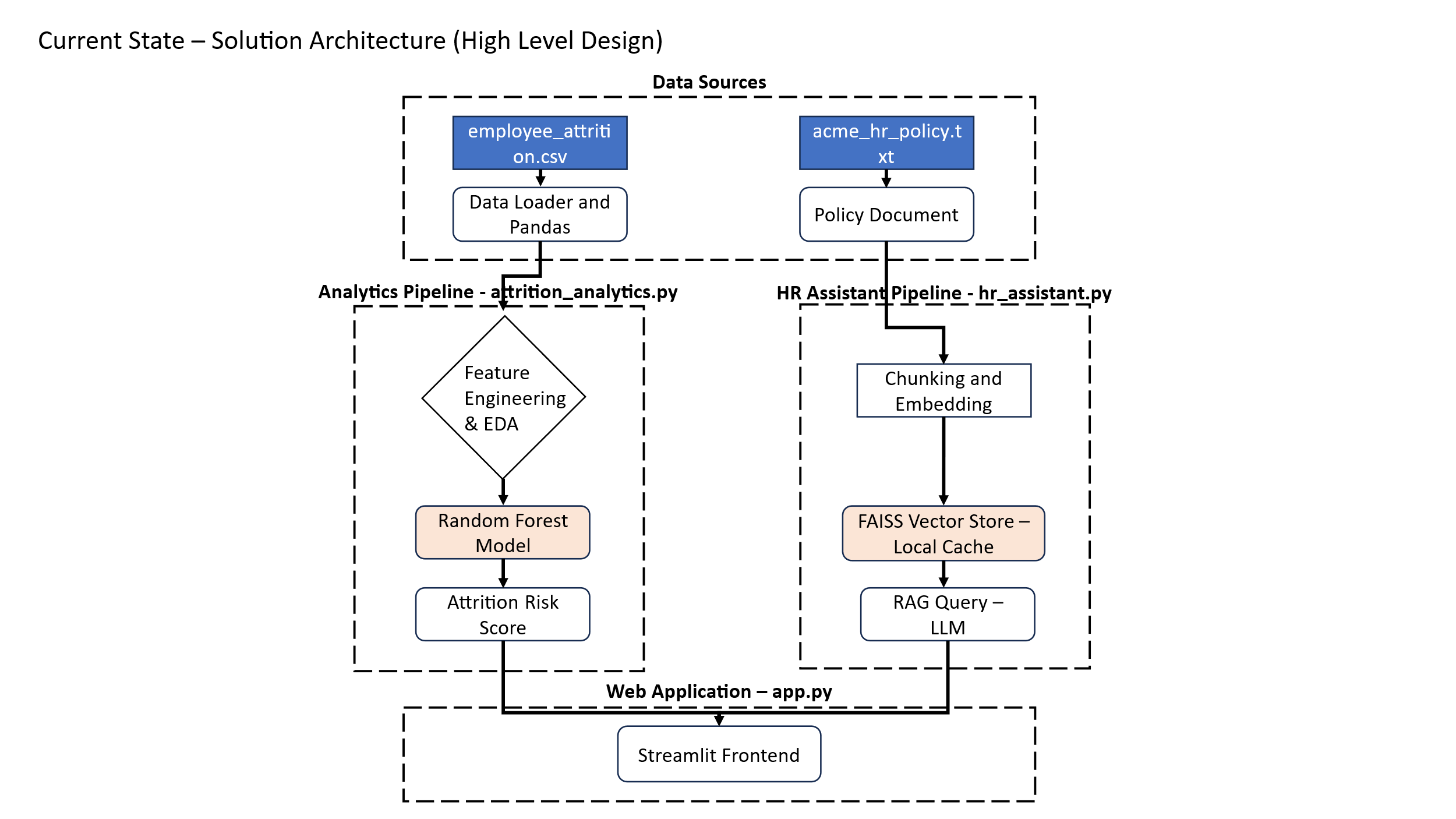
Data indicates high turnover is driven by dissatisfaction with work-life balance, unclear growth paths, and management practices. The goal is not just to report *who* is leaving, but to understand *why* and *predict* who might leave next.

### 2.2. The HR Efficiency Challenge

HR resources are drained by routine questions (e.g., "What is the maternity leave policy?"). Manual responses create bottlenecks. The goal is to automate these using the existing Acme HR Policy Handbook as a knowledge base.

## 3. Solution Architecture & Design Principles

The solution is built as a unified web application using **Streamlit**, integrating advanced analytics and Generative AI components. A core design principle was to ensure maintainability and scalability through strict separation of concerns.



### 3.1. Tech Stack

* **Frontend/App Framework:** Streamlit (Python)
* **Data Processing:** Pandas, NumPy
* **Machine Learning:** Scikit-learn (Random Forest Classifier)
* **Visualization:** Plotly Express
* **GenAI & NLP:** LangChain, FAISS (Vector Store), HuggingFace Embeddings (Sentence Transformers)

### 3.2. Configuration Management (No Hard-Coding)

To ensure the solution is robust and adaptable to future policy changes without code intervention, all business logic parameters are decoupled from the application logic.

* **Centralized Configuration:** All thresholds, constants, and settings are maintained in a separate file: attrition\_config.py.
* **Configurable Parameters include:**
  + **Risk Thresholds:** Definitions for "Low Satisfaction" (e.g., < 2) or "High Overtime" are editable variables (RISK\_CONFIG).
  + **Financial Assumptions:** Cost of replacement multipliers (currently set to 6 months' salary) can be adjusted in COST\_CONFIG.
  + **UI Settings:** Chart dimensions, colors, and layout grids are controlled via UI\_CONFIG and CHART\_CONFIG.
  + **Feature Engineering:** Bin sizes for Tenure groupings (e.g., 0-2 years, 3-5 years) are defined in FEATURE\_CONFIG.

This approach allows data analysts or non-technical administrators to tweak the dashboard's sensitivity and visual output by simply updating the config file, without touching the core codebase.

## 4. Part 1: Attrition Analytics (Diagnostic & Predictive)

The Analytics Module (attrition\_analytics.py) transforms raw HR data into actionable intelligence.

### 4.1. Methodology & Scalability Features

The solution is engineered to handle enterprise-scale datasets (50,000+ rows) efficiently:

* **Memory Optimization:** The data loader utilizes explicit **Data Type Mapping** (e.g., converting 64-bit integers to 8-bit or 16-bit where appropriate). This reduces memory footprint by approximately 60%, allowing large datasets to be processed in-memory.
* **Vectorized Operations:** Loops are replaced with vectorized Pandas/NumPy operations for feature engineering. For example, calculating Role\_Stagnation or Satisfaction\_Score happens instantaneously across the entire dataset rather than row-by-row.
* **Parallel Processing:** The Random Forest model utilizes n\_jobs=-1, leveraging all available CPU cores for model training and risk scoring.
* **Intelligent Caching:** Expensive computations (like the predictive model training or complex aggregations) are cached. The system only re-computes these if the underlying data inputs change.

### 4.2. Dynamic Data Updates

The application is designed for real-time responsiveness to data changes:

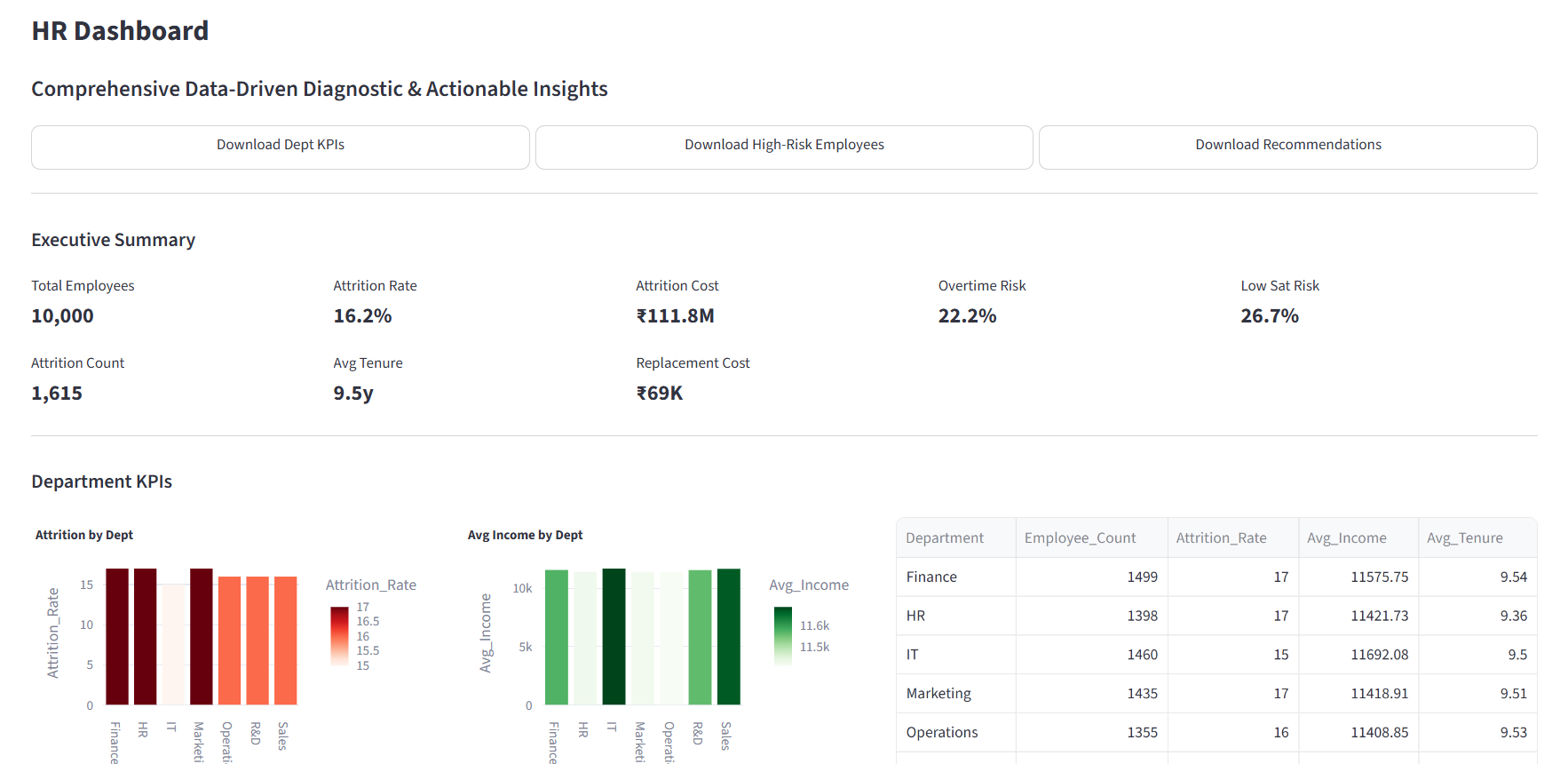
* **Live Reloading:** The analytics engine is not static. If the underlying employee\_attrition.csv file is updated (e.g., with next month's HR data), simply reloading the web page triggers the application to ingest the new data, re-run the risk models, and update all KPIs instantly.
* **Auto-Calibration:** Because thresholds are percentile-based (e.g., Income Quartiles defined in attrition\_config.py), the risk segmentation automatically adjusts to the distribution of the new data.

### 4.3. Key Insights Capabilities

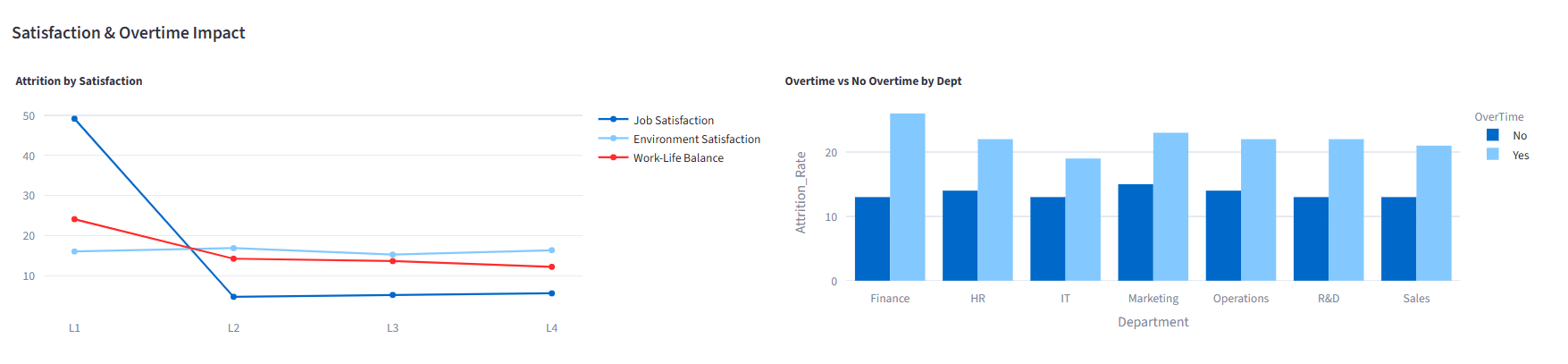
The dashboard provides specific insights into:

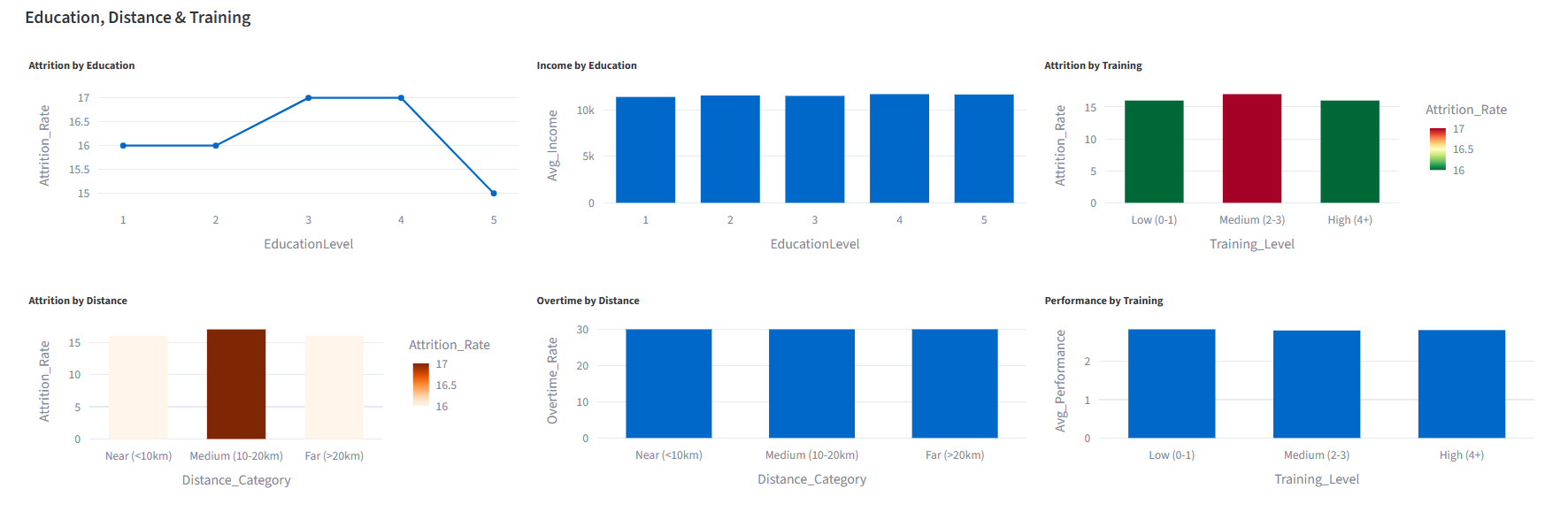
* **The "Burnout" Segment:** Employees with High Overtime and Low Satisfaction.
* **The "Stagnation" Segment:** Employees stuck in the same role for 7+ years.
* **Financial Impact:** Real-time calculation of "Replacement Costs" based on average salaries and industry multipliers.
* **Training ROI:** Correlation between training frequency and performance/retention.

### 4.4. Visualizations

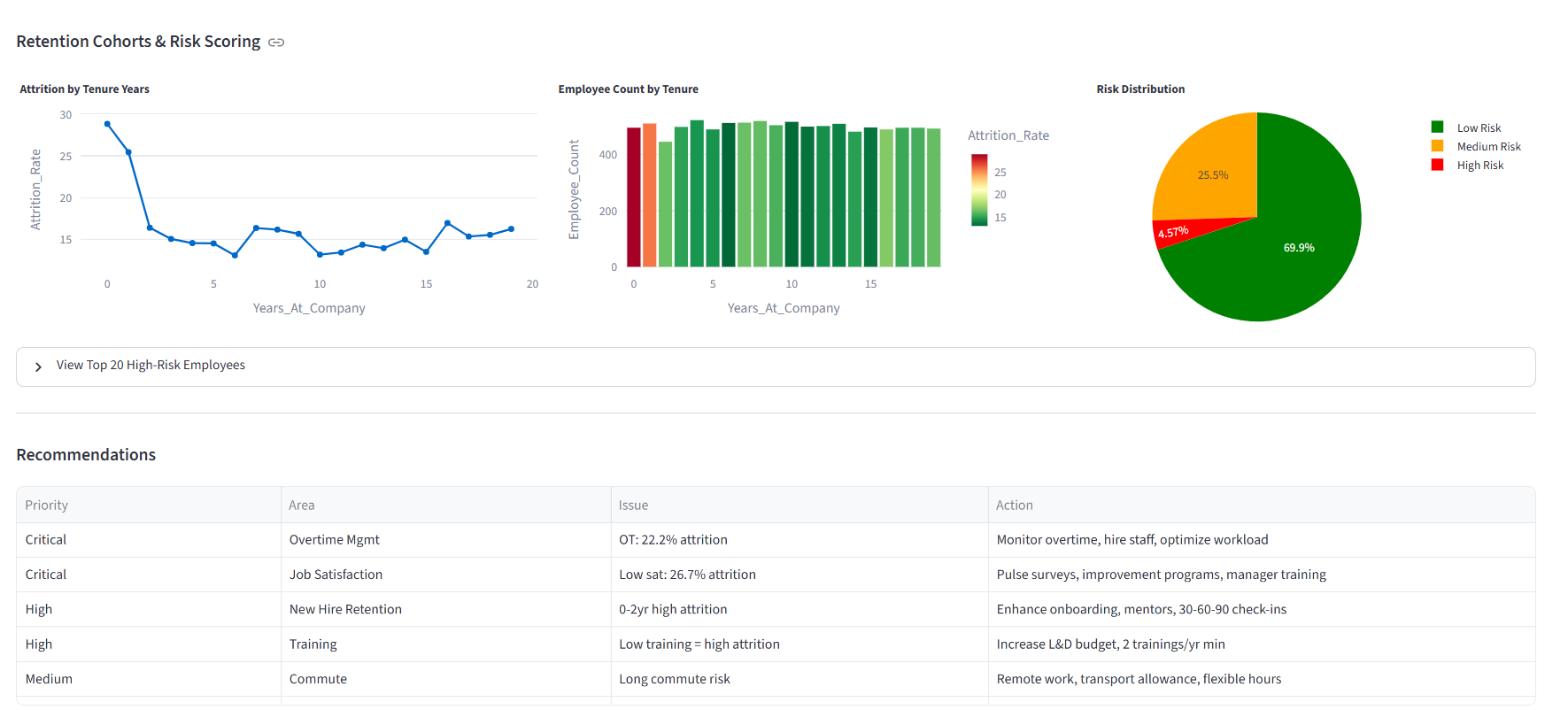












## 5. Part 2: AI-Powered HR Assistant (The "How")

The HR Assistant (hr\_assistant.py) utilizes a **Retrieval-Augmented Generation (RAG)** architecture to answer questions grounded strictly in Acme Corp's policy.

### 5.1. RAG Pipeline Design

1. **Ingestion:** The acme\_hr\_policy.txt is loaded and cleaned.
2. **Chunking:** The text is split into semantic chunks (approx. 600 characters) using RecursiveCharacterTextSplitter to maintain context (e.g., keeping "Leave Policy" rules together).
3. **Embedding:** Text chunks are converted into vector embeddings using sentence-transformers/all-MiniLM-L6-v2. This model was chosen for its balance of speed and semantic accuracy.
4. **Vector Storage:** Embeddings are stored in a local **FAISS** index for millisecond-latency similarity search.
5. **Retrieval & Generation:**
   * When a user asks a question, the system retrieves the top 4 most relevant policy chunks.
   * These chunks are passed as "Context" to the generation layer, ensuring the answer is accurate and sourced.

### 5.2. Performance Optimizations

* **Vector Store Caching:** The embedding generation process (which is computationally expensive) is performed only once. The resulting Vector Store is pickled and saved to disk (.cache/). Subsequent application launches load this pre-computed index instantly.
* **LRU Caching:** An in-memory Least Recently Used (LRU) cache stores answers to frequent questions, providing 0-latency responses for common queries like "What is the notice period?".

## 6. Part 3: Web Application Prototype

The application (app.py) serves as a single entry point for all users.

* **Executive Dashboard:** A high-level view for leadership showing Attrition Rate, Total Cost of Attrition, and Headcount trends.
* **Department View:** Drill-down capabilities for HRBPs to view specific risks in R&D, Sales, or HR.
* **Employee Self-Service:** A dedicated tab for the "Ask HR" chatbot.
* **User Experience (UX):**
  + Clean, sidebar-based navigation.
  + Interactive Plotly charts (hover for details).
  + One-click download of reports (High-risk employee lists, Recommendations).

## 7. Part 4: Cloud Deployment Strategy

To move this prototype to production, the following architecture is recommended:



### 7.1. Containerization

The entire application will be containerized using **Docker**.

* **Dockerfile:** Defines the environment (Python 3.9), installs dependencies (requirements.txt), and exposes the Streamlit port (8501).
* **Benefits:** Ensures consistency across development, testing, and production environments.

### 7.2. Cloud Architecture (Azure/AWS)

We recommend a serverless container approach for scalability and cost-efficiency.

**Option A: Azure (Preferred for Enterprise)**

1. **Hosting:** **Azure Container Apps** or **App Service for Containers**. This allows the app to scale down to zero when not in use.
2. **Vector Database:** While FAISS (local) works for the prototype, production should use **Azure AI Search** or a managed **ChromaDB** instance for persistent vector storage.
3. **Secrets Management:** API keys (if using OpenAI/Azure OpenAI) and database credentials will be stored in **Azure Key Vault**, injected as environment variables at runtime.

**Option B: AWS**

1. **Hosting:** **AWS Fargate** (ECS) for serverless container orchestration.
2. **Storage:** S3 for storing raw policy documents and model artifacts.

### 7.3. CI/CD Pipeline

1. Code is pushed to **GitHub**.
2. **GitHub Actions** triggers a build:
   * Runs unit tests.
   * Builds the Docker image.
   * Pushes the image to **Azure Container Registry (ACR)** or **Amazon ECR**.
3. The cloud service pulls the new image and redeploy the application automatically.

## 8. Strategic Recommendations

Based on the initial analysis capabilities of the tool, we recommend Acme Corp prioritize the following:

1. **Address the "Overtime Burnout":** Data suggests a strong correlation between overtime and attrition. Implement a policy to flag managers whose teams consistently exceed overtime thresholds.
2. **Revamp Onboarding for New Hires:** High attrition in the <2 Year Tenure bracket indicates onboarding failures. Introduce a "30-60-90 Day" mentorship check-in program.
3. **Market Correction for Salaries:** Employees in the lowest income quartile show higher risk. Conduct a compensation benchmarking study for these specific bands.
4. **Promote the HR Assistant:** Launch an internal campaign to drive employees to the chatbot for Level 1 queries, freeing HRPs to focus on complex retention interviews.

## 9. Conclusion

This solution provides Acme Corp with the immediate ability to stop bleeding talent through predictive insights and the long-term capability to scale HR operations through AI. The architecture is modular, secure, and ready for cloud deployment.

## 10. Appendix: Project Artifacts

The solution package submitted contains the following modular components:

|  |  |
| --- | --- |
| **File Name** | **Description** |
| **app.py** | Main Streamlit application entry point handling navigation and UI. |
| **attrition\_analytics.py** | Core data processing engine, ML models, and statistical analysis. |
| **attrition\_config.py** | Centralized configuration file for thresholds, UI settings, and logic parameters. |
| **hr\_assistant.py** | RAG-based AI assistant module for processing HR policy queries. |
| **acme\_hr\_policy.txt** | Source knowledge base for the AI assistant. |
| **employee\_attrition.csv** | Raw dataset used for attrition diagnostics. |