**HR Q&A System POC - Complete Requirements & Context Document**

**📋 Project Overview**

**Business Context**

* **Project Name**: HR Q&A System with MongoDB Vector Architecture
* **Type**: Proof of Concept (POC)
* **Scale**: 50 employees initially, designed to scale to 5000+ employees
* **Purpose**: Transform static HR data into an intelligent Q&A system using natural language processing
* **Business Value**: Enable HR teams and employees to query complex HR data using natural language instead of manual spreadsheet analysis

**Problem Statement**

Current HR data management challenges:

* HR data scattered across multiple Excel/CSV files
* No easy way to query complex relationships (employee → projects → leave → training)
* Manual effort required for simple HR analytics
* No semantic search capability for HR policies and procedures
* Difficulty in finding specific employee information quickly
* Complex queries require SQL knowledge or manual data manipulation

**Solution Approach**

Build an AI-powered Q&A system that:

* Converts CSV/Excel HR data into structured MongoDB collections
* Uses Azure AI Search for hybrid (vector + traditional) search capabilities
* Leverages Azure OpenAI for natural language understanding and response generation
* Provides both simple lookups and complex analytical queries
* Enables real-time querying without technical expertise

**🏗️ Technical Architecture**

**Technology Stack**

Database: MongoDB Atlas (Document storage with flexible schema)

Search Engine: Azure AI Search (Hybrid vector + keyword search)

AI/LLM: Azure OpenAI (GPT-4o for responses, Ada-002 for embeddings)

Backend Language: Python 3.10

Framework: FastAPI (for API) + CLI interface

Data Processing: Pandas for CSV processing

Environment: Cloud-first with local development support

**Data Flow Architecture**

CSV Files → Python ETL → MongoDB Collections → Azure AI Search Indexes → Query Engine → Azure OpenAI → Natural Language Response

**Core Components**

1. **Data Processing Pipeline**: CSV → JSON → MongoDB
2. **Search Integration**: MongoDB → Azure AI Search with vector embeddings
3. **Query Engine**: Natural language → search queries → AI responses
4. **Interfaces**: CLI for POC, API ready for future web/mobile interfaces

**📊 Data Model & Structure**

**Source Data (HR CSV Files)**

The system processes these HR data categories:

Employee Master Data:

- Personal info (name, email, birth date, blood group, marital status)

- Employment details (job code, department, business unit, supervisor)

- Location info (onsite/offshore, country, birth place)

- Experience data (months of experience, hire date)

Project Allocations:

- Employee assignments to projects

- Project roles and responsibilities

- Allocation percentages and dates

- Project status and timelines

Leave Management:

- Leave balances by employee

- Leave transaction history

- Leave types and policies

- Approval workflows

Training Records:

- LMS training attempts

- Completion status

- Training types and categories

- Certification tracking

Job Reference Data:

- Job codes and descriptions

- Organizational hierarchy

- Role definitions

Exit Interviews:

- Departure reasons

- Exit interview feedback

- Attrition analysis data

**MongoDB Collections Design**

// employees collection - Primary employee data

{

"employee\_id": "123456",

"personal": {

"first\_name": "John",

"last\_name": "Doe",

"email": "john.doe@company.com",

"birth\_date": "1990-01-15",

"blood\_group": "O+",

"marital\_status": "Single"

},

"employment": {

"job\_code": "SE001",

"business\_unit": "IT",

"department\_id": "DEP001",

"supervisor\_id": "SUP001",

"location\_type": "Offshore",

"hire\_date": "2020-01-15",

"months\_experience": 60

},

"vector\_embedding": [0.1, 0.2, ...], // 1536 dimensions

"metadata": {

"created\_at": ISODate,

"updated\_at": ISODate,

"version": 1

}

}

// allocations collection - Project assignments

{

"employee\_id": "123456",

"project\_id": "PROJ001",

"allocation\_percentage": 80,

"project\_role": "Senior Developer",

"start\_date": "2024-01-01",

"end\_date": "2024-12-31",

"status": "active",

"vector\_embedding": [...]

}

// leave collection - Leave management

{

"employee\_id": "123456",

"leave\_type": "Annual Leave",

"request\_date": "2024-06-01",

"start\_date": "2024-06-15",

"days\_requested": 5,

"status": "approved",

"current\_balance": 18.5

}

// training collection - Learning records

{

"employee\_id": "123456",

"training\_type": "Technical Certification",

"completion\_status": "completed",

"completion\_date": "2024-03-15",

"vector\_embedding": [...]

}

// job\_reference collection - Job definitions

{

"job\_code": "SE001",

"job\_title": "Senior Software Engineer",

"department": "Engineering",

"level": "Senior"

}

// exit\_interviews collection - Attrition data

{

"employee\_id": "123456",

"exit\_date": "2024-05-01",

"exit\_reason": "Better opportunity",

"feedback": "...",

"vector\_embedding": [...]

}

**🎯 Use Cases & Query Examples**

**Simple Queries (Direct Data Lookup)**

Employee Information:

- "Show me details of employee ID 123456"

- "What is John Doe's email address?"

- "Who is the supervisor of employee 789012?"

- "List all employees in the IT department"

Basic Analytics:

- "How many employees do we have?"

- "What are the different job codes?"

- "Show me all offshore employees"

- "List employees hired in 2023"

**Complex Queries (Multi-Collection Analytics)**

Cross-Functional Analysis:

- "Show me offshore employees with more than 5 years experience who have taken more than 10 days leave this year"

- "Which supervisors have the most direct reports?"

- "Find employees working on AI projects who have completed technical certifications"

- "Compare leave patterns between onsite and offshore employees"

Trend Analysis:

- "What are the most common reasons for employee exits?"

- "Which departments have the highest attrition rate?"

- "Show me training completion rates by business unit"

- "Identify employees with low project allocation percentages"

Predictive Insights:

- "Which employees might be at risk of leaving based on patterns?"

- "What projects have the highest team turnover?"

- "Identify skill gaps in specific departments"

**Semantic Search Queries**

Natural Language Understanding:

- "Find developers working on machine learning projects"

- "Show me people similar to John Doe's profile"

- "Who has experience in both frontend and backend development?"

- "Find employees with leadership potential"

**🔧 Technical Implementation Details**

**Data Processing Pipeline**

# ETL Process Flow:

1. CSV Validation → Check data quality, missing fields, format consistency

2. Data Transformation → Convert CSV rows to JSON documents

3. Schema Validation → Ensure data matches MongoDB collection schemas

4. MongoDB Insertion → Upsert documents with employee\_id as unique key

5. Vector Generation → Create embeddings for searchable text fields

6. Azure Search Indexing → Sync MongoDB data to Azure AI Search

7. Index Optimization → Create search indexes for performance

**Query Processing Architecture**

# Query Flow:

1. Natural Language Input → User types question

2. Intent Classification → Determine query type (simple/complex/semantic)

3. Query Planning → Decide search strategy (MongoDB direct/Azure Search/hybrid)

4. Data Retrieval → Execute searches and gather relevant data

5. Context Building → Prepare data context for AI

6. AI Processing → Send context + question to Azure OpenAI

7. Response Generation → GPT-4 generates natural language answer

8. Result Formatting → Present answer with data sources

**Search Strategy**

Hybrid Search Approach:

Vector Search: For semantic similarity and natural language understanding

Keyword Search: For exact matches and structured data queries

Filtering: For demographic and categorical data

Aggregation: For analytics and statistical queries

Search Index Design:

- employees\_search\_index: Personal and employment data

- projects\_search\_index: Project allocation and role data

- training\_search\_index: Learning and certification data

- Combined semantic search across all collections

**🏃‍♂️ Development Phases**

**Phase 1: Foundation (Week 1)**

Goals:

- Set up MongoDB Atlas connection

- Create Azure AI Search service

- Build CSV → MongoDB ETL pipeline

- Validate data processing with 50 employee sample

Deliverables:

- Working data ingestion pipeline

- MongoDB collections populated

- Basic data validation reports

- Environment configuration complete

**Phase 2: Search Integration (Week 2)**

Goals:

- Integrate Azure AI Search

- Generate vector embeddings with OpenAI

- Create search indexes

- Test hybrid search capabilities

Deliverables:

- Azure Search indexes created

- Vector embeddings generated and stored

- Basic search functionality working

- Search performance benchmarks

**Phase 3: Query Engine (Week 3)**

Goals:

- Build natural language query processor

- Integrate Azure OpenAI for responses

- Create CLI interface

- Test complex query scenarios

Deliverables:

- Working Q&A system

- CLI interface for testing

- Sample queries and responses documented

- Performance optimization complete

**Phase 4: Optimization & Testing (Week 4)**

Goals:

- Performance tuning

- Comprehensive testing

- Documentation completion

- Production readiness assessment

Deliverables:

- Performance benchmarks

- Test coverage reports

- Complete documentation

- Deployment guidelines

**🎛️ Configuration & Environment**

**Required Services & Access**

MongoDB Atlas:

- Cluster: M0 (Free tier) or M10 for better performance

- Database: hr\_qna\_poc

- Collections: employees, allocations, leave, training, job\_reference, exit\_interviews

Azure AI Search:

- Service Tier: Free (50MB, 10K documents) for POC

- Indexes: 6 search indexes (one per collection)

- Features: Vector search, semantic search, filtering

Azure OpenAI:

- Models: GPT-4o (for responses), text-embedding-ada-002 (for vectors)

- Deployment: Existing deployment already provisioned

- Usage: Pay-per-token for embeddings and completions

**Development Environment**

Local Setup:

- Python 3.10+

- Virtual environment

- Git repository

- Code editor (VS Code recommended)

Dependencies:

- motor (async MongoDB driver)

- azure-search-documents (Azure AI Search)

- openai (Azure OpenAI client)

- fastapi (API framework)

- pandas (data processing)

- pydantic (data validation)

**📈 Success Criteria & KPIs**

**Functional Requirements**

Data Processing:

✅ Successfully process 50 employee CSV records

✅ Zero data loss during transformation

✅ Handle missing/invalid data gracefully

✅ Maintain data relationships across collections

Search Capabilities:

✅ Support both simple and complex queries

✅ Semantic search with 80%+ relevance accuracy

✅ Response time under 3 seconds for 95% of queries

✅ Handle 10+ concurrent queries

AI Integration:

✅ Natural language understanding for HR domain

✅ Accurate data retrieval and synthesis

✅ Contextually relevant responses

✅ Proper handling of ambiguous queries

**Performance Targets**

Response Time: <3 seconds for 95% of queries

Accuracy: 90%+ correct responses for test query set

Scalability: Handle 500+ employees without performance degradation

Availability: 99%+ uptime during testing period

Cost: <$100/month for POC environment

**🔄 Future Roadmap**

**Immediate Enhancements (Post-POC)**

User Interface:

- Web-based dashboard

- Mobile app integration

- Voice interface capabilities

Advanced Features:

- Real-time data synchronization

- Advanced analytics dashboards

- Predictive HR insights

- Integration with HRIS systems

Scalability:

- Multi-tenant architecture

- Enterprise security features

- Advanced caching strategies

- Load balancing and auto-scaling

**Integration Possibilities**

HR Systems:

- Workday, SAP SuccessFactors integration

- Active Directory synchronization

- Payroll system connections

Communication Platforms:

- Slack/Teams bot integration

- Email query capabilities

- Webhook notifications

Analytics Platforms:

- PowerBI dashboard integration

- Tableau connector

- Custom reporting APIs

**🚀 Quick Start Context for New Sessions**

When starting a new conversation, reference this document and the assistant will understand:

* The complete technical architecture and decisions made
* Current implementation status and next steps
* Data model and collection schemas
* Query capabilities and use cases
* Development environment and tools used
* Performance targets and success criteria

**Current Status**: Ready to begin Phase 1 implementation with foundation setup and data processing pipeline.

**Next Immediate Steps**:

1. Set up project structure
2. Configure MongoDB Atlas connection
3. Create Azure AI Search service
4. Build CSV processing pipeline
5. Test with 50 employee sample data

This document serves as the complete project context that can be referenced in any new conversation to maintain full project understanding and continuity.