Knowledge Management System

This is a test document for the KnowledgeScout RAG system.

This document contains multiple pages with different topics to test document parsing, chunking, embedding generation, and semantic search capabilities.

Page 1 covers the introduction and overview of the system.

System Overview

The KnowledgeScout system is designed to:

- Upload and parse PDF/DOCX documents
- Extract text with page-level granularity
- Generate vector embeddings using Transformers.js
- Store embeddings in MongoDB with Atlas Vector Search
- Provide semantic search with source attribution
- Cache query results for performance

Technical Architecture

Page 2 describes the technical stack and architecture.

Technology Stack

Backend Framework: Hono.js - Fast, lightweight, and works on any JavaScript runtime.

Database: MongoDB with Atlas Vector Search for semantic similarity search.

Embeddings: Transformers.js (Xenova/all-MiniLM-L6-v2) for local embedding generation.

Authentication: JWT-based bearer token authentication with refresh tokens.

Document Parsing: pdf-parse for PDFs, mammoth for DOCX files.

Key Features

The system implements several important features for production use:

- Private document access control with share tokens
- Query caching with 60-second TTL for performance
- Page-level source attribution for accurate citations
- Pagination for document listings
- Real-time indexing status tracking

API Endpoints

Page 3 lists all available API endpoints and their functionality.

Authentication Endpoints

POST /api/auth/register - Register new user account

POST /api/auth/login - Login and receive access tokens

POST /api/auth/refresh - Refresh expired access token

POST /api/auth/logout - Logout and invalidate tokens

Document Management Endpoints

POST /api/docs - Upload new document (multipart/form-data)

GET /api/docs - List documents with pagination

GET /api/docs/:id - Get single document details

DELETE /api/docs/:id - Delete document and associated chunks

Query and Indexing Endpoints

POST /api/ask - Query documents with semantic search

POST /api/index/rebuild - Rebuild document index

GET /api/index/stats - Get indexing statistics

Security & Best Practices

Page 4 covers security considerations and implementation best practices.

Security Measures

The system implements multiple layers of security:

- JWT access tokens expire after 15 minutes
- Refresh tokens stored in database for revocation
- Document access control based on ownership and privacy settings
- Share tokens for controlled document sharing
- Password hashing with bcrypt (10 rounds)
- CORS configuration for trusted origins

Performance Optimization

Query caching reduces duplicate embedding computations by 60 seconds.

MongoDB indexes optimize document and chunk queries.

Vector search uses cosine similarity with efficient HNSW indexing.

Async document processing prevents blocking API responses.

Usage Examples

Page 5 provides practical examples of using the system.

Example Query Scenarios

Question: "What technology stack does the system use?"

Expected Result: Chunks from Page 2 describing Hono.js, MongoDB, and Transformers.js

Question: "How does authentication work?"

Expected Result: Chunks from Page 3 and Page 4 about JWT tokens and security

Question: "What are the API endpoints for documents?"

Expected Result: Chunks from Page 3 listing document management endpoints

Testing Recommendations

Test with various query types: factual, comparative, and exploratory questions.

Verify page numbers in citations match source content.

Check cache behavior with duplicate queries within 60 seconds.

Test private document access control with different users.