AI-Powered Knowledge Management System (RAG)

1. RAG Architecture

```
[Documents: PDF/Word/PPT/Emails]
            │
   [Document Ingestion & Preprocessing]
            │
            ▼
     [Text Chunking + Metadata]
            │
            ▼
      [Embedding Model / Vectorization]
            │
            ▼
       [Vector Database]
            │
            ▼
        [RAG Engine / LLM]
            │
            ▼
    [Web + Mobile UI: Queries & Responses]
            │
            ▼
     [Source Citations & Feedback]
```

2. Technology Stack

Cloud Provider: AWS (US regions)

Document Ingestion: Lambda, S3, Textract, Apache Tika

Vector Database: Pinecone or Weaviate
Embedding Model: OpenAI Embeddings or Cohere
RAG Engine / LLM: GPT-4 API or LLaMA 2 via Sagemaker
Web/Mobile UI: React, React Native
Authentication: AWS Cognito (SSO)
Logging/Audit: CloudWatch, S3, DynamoDB
Orchestration: Step Functions
Caching: Redis / ElastiCache
CI/CD: GitHub Actions / AWS CodePipeline

3. Security Architecture

Authentication: SSO with MFA
Authorization: RBAC, document-level permissions
Data Protection: Encryption at rest (S3 SSE-KMS), encryption in transit (TLS)
Audit: Query logs in S3/DynamoDB

4. Scaling Strategy

Document ingestion: event-driven via Lambda
Vector DB: sharding or managed scaling
Concurrent users: autoscaling ECS/Fargate, caching
Uptime: Multi-AZ deployment, health checks, backups

5. Cost Strategy

LLM calls: precompute embeddings, batch processing
Vector DB: Pinecone pay-per-usage or self-host FAISS
Storage: S3, Glacier for old docs
Compute: Lambda/ECS Fargate
Monitoring: CloudWatch with retention limits

6. Implementation Phases

Phase 1 (MVP, Month 1-2): PDF/Word ingestion, embedding storage, basic chat UI, SSO login
Phase 2 (Month 3-4): PowerPoint/email ingestion, metadata filters, source citations
Phase 3 (Month 5): Real-time ingestion (SQS/Lambda), caching, audit logs
Phase 4 (Month 6): Mobile UI, advanced RBAC, feedback loop, scaling, multi-AZ deployment