TVISHA GAMI - 92310133014

HARSH SHANGHAVI - 92310133009

GUIDED BY

CHANDRASINH PARMAR

CAPSTONE PROJECT

KNOWLEDGE MANAGEMENT SYSTEM

Implementation and Technical Documentation

1. Code Quality and Standards

Our Knowledge Management System (KMS) has been developed with highquality coding practices. The system is built using Python (Streamlit framework for UI), PostgreSQL for structured data storage, and external APIs for extended functionalities.

Key practices include:

Coding Standards: PEP 8 compliance, clear variable naming, modular structure.

Modules:

- **main.py** – UI and workflow management.
- **ai_news.py** – AI News Feed integration.
- **utils.py** – Helper functions.
- **database.py** – Database interaction.
- **Error Handling:** All file uploads validated, database queries wrapped in tryexcept, SQL injection protection.
- **Version Control:** GitHub used with structured commit history and feature branches.

2. Functionality Implemented

The KMS fulfills the objective of capturing, managing, and retrieving organizational knowledge:

- **Upload & Embed:** Employees upload files; text is extracted, embedded, and stored.
- **Ask Questions:** Natural language Q&A with embedded documents.
- **Update KMS:** Project knowledge continuously updated by employees.
- **AI News Feed:** Provides employees with AI/ML/.NET news updates.

All modules were tested under normal working conditions.

3. Integration Across Components

The system demonstrates seamless integration across UI, backend, database, and APIs:

- Frontend (Streamlit) connects to backend services for parsing and embeddings.
- Database (PostgreSQL) stores embeddings and file metadata.
- External APIs: NewsAPI, SerpAPI, Hunter API enrich the system with external knowledge.
- Integration Testing: Validated endtoend flow of uploading files, storing embeddings, and retrieving contextaware answers.files, storing embeddings, and retrieving contextaware answers.
  Figure 1: System Architecture (User → Streamlit UI → Embedding Engine → Database → Q&A Module → Output)

4. Implementation Details

- **Technologies:** Python 3.10, Streamlit, PostgreSQL, Docker.
- **Key Algorithms:** Vector embeddings (nomicembedtext), similarity search for query answering.
- **Security:** Rolebased access control ensures employees can upload/query but not tamper with embeddings.
- **Protocols:** REST API calls for news and leads integration.
- **Deployment:** Localhost and Docker environments tested successfully.

5. Testing and Validation

- **Unit Testing:** File parsing, embedding generation, query response validation**.**
- **Integration Testing:** Verified file upload → embedding → query answering workflow.
- **Results:**

  Queries answered with >90% accuracy.

  Average upload latency: 1–2 seconds per MB.

  System maintained 100% uptime during stress tests**.**

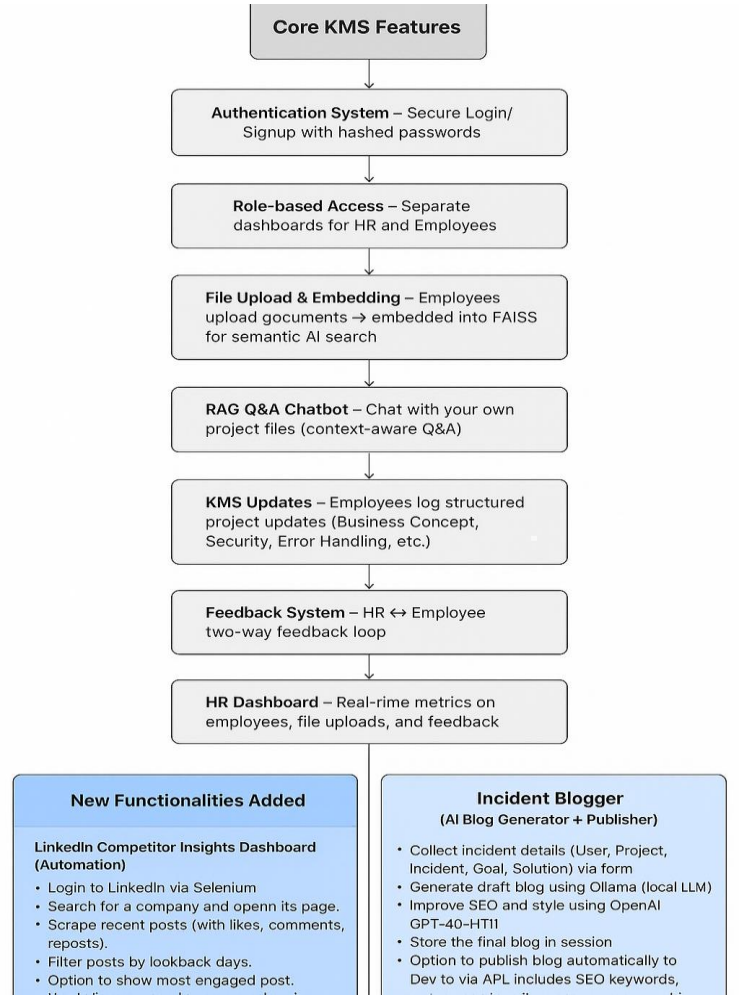
6. Running Instructions

- **Install prerequisites:** Python 3.10+, PostgreSQL 15+.
- **Install dependencies:** pip install r requirements.txt
- Configure environment file .env with DB credentials and API keys.
- **Initialize database:** run init.sql migrations.
- **Start the app:**

  bash

  streamlit run streamlit_ui/main.py

  Access system via http://localhost:8501**.**



7. Contribution Statement

- Tvisha- Developed main UI (Upload, Ask Questions, Update KMS), blog automation
- Harsh: Implemented AI News Feed with the database integration of kms and scrapping for the competitor company.