TVISHA GAMI - 92310133014

HARSH SHANGHAVI - 92310133009

GUIDED BY

CHANDRASINH PARMAR

CAPSTONE PROJECT

KNOWLEDGE MANAGEMENT SYSTEM

System Design and Architecture

1. Introduction

The system design and architecture for this project emphasize modularity, scalability, and efficient use of ICT tools. This section details the modular structure, technology stack, and scalability planning to ensure the proposed solution is robust, adaptable, and future-ready.

2. Modular Design

1. Authentication & Role Management

- Handles secure login/signup using hashed passwords (bcrypt or SHA-256).
- Provides role-based access: HR Dashboard vs. Employee Dashboard**.**
- Includes session management to track active users.
- Justification: Authentication ensures data security, while role separation improves usability for diverse stakeholders.

2. File Upload & Embedding Module

- Employees upload documents (PDF, DOCX, TXT).
- Files are parsed, and embeddings are generated using FAISS for semantic search.
- Supports real-time document indexing to ensure newly added files are available for querying.
- Justification: Enables AI-powered document search, allowing employees to retrieve precise information quickly.

3. RAG Q&A Chatbot

- Uses Retrieval-Augmented Generation (RAG).
- Retrieves relevant context from embedded documents and generates answers using an LLM (Ollama locally).
- Maintains session-based chat history for continuity.
- Justification: Empowers employees to interact with stored knowledge conversationally

4. KMS Update Logger

- Structured forms for employees to record project updates: Business Concept, Security, Error Handling, Progress Notes**.**
- Updates stored in PostgreSQL for long-term reference.
- Justification: Ensures project continuity even when employees leave

5. HR Dashboard

- Provides summary metrics:
  - Total employees
  - Number of uploaded files
  - Feedback received
  - Justification: Offers HR an overview of organizational knowledge flow.

6. Feedback System

- Enables two-way feedback between HR and employees.
- Feedback is tagged to projects and employees for traceability
- Justification: Improves collaboration and communication between stakeholders.

7. AI News Feed

- Fetches real-time AI/ML/.NET news articles using NewsAPI.
- Filters and displays updates in a scrolling footer across all pages.
- Justification: Keeps employees updated with industry trends, encouraging learning.

8. Competitor Insights (LinkedIn Automation)

- Automates LinkedIn search using Selenium.
- Extracts competitor posts with engagement metrics (likes, comments, shares).
- Displays most-engaged content in styled cards.
- Justification: Provides market intelligence for decision-making.

9. Incident Blogger

- Employees log incidents (problem, solution, learning).
- Generates AI-enhanced blogs using Ollama + GPT.
- Option to publish to **Dev.to** automatically via API.
- Justification: Transforms internal knowledge into public-facing blogs, enhancing organizational reputation.

10. Database & Storage

- PostgreSQL stores user data, feedback, logs, and structured project updates.
- FAISS stores embeddings for fast semantic retrieval.
- Redis can be used for caching frequently accessed queries.
- Justification: Ensures high-performance, structured, and secure data management.

3. Technology Stack

The technology stack is selected to ensure performance, scalability, and reliability.

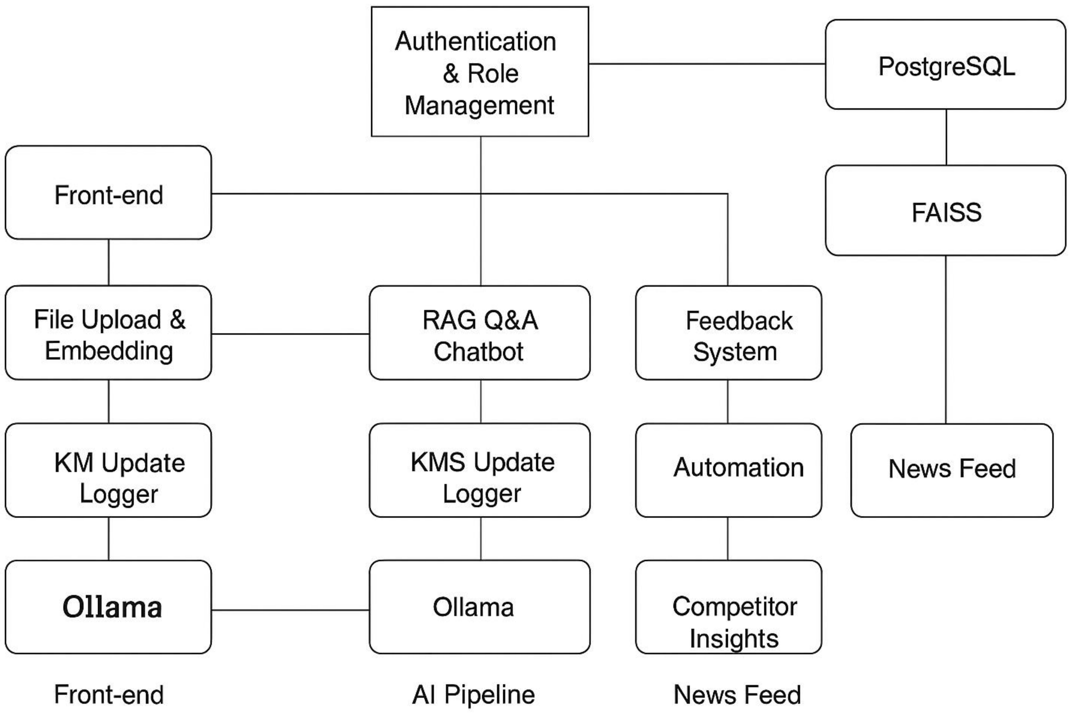
- **Front-End & UI:** Streamlit and flask (lightweight, Python-based rapid prototyping).
- **Back-End Processing:** Python (for NLP, embedding generation, RAG pipeline).
- **Database:** PostgreSQL (structured data), FAISS (vector embeddings)
- **AI/ML Models:** Ollama (local LLM2 and 3, pie-2, many more), SentenceTransformers for embeddings.
- **Automation:** Selenium, Beatifulsoap (LinkedIn scraping).
- **Cloud/Deployment:** Docker for containerization, potential AWS/GCP and grage server for hosting windows server 2022.
- **Version Control:** Git/GitHub for collaborative development.
- **Libraries:** langchain, psycopg2, python-docx, streamlit, requests many more

Justification: This stack ensures compatibility with modern AI frameworks, security for enterprise data, and fast deployment options.

4. Scalability Plan

To handle growth in data, users, and traffic, the following strategies are planned:

- **Horizontal Scaling (Cloud Containers):**
  - Docker for container orchestration.
  - Scale out multiple instances of Streamlit + API back-end.
- **Database Scaling:**
  - PostgreSQL → partitioning, sharding for large datasets.
  - Use connection pooling to reduce overhead.
  - Replication for high availability.
- **Vector Store Optimization:**
  - FAISS supports distributed search for large embeddings.
  - Index compression for memory efficiency.
- **Cost & Performance Consideration:**
  - Cloud scaling only enabled under high load to reduce cost.
  - Hybrid approach (local for embeddings, cloud for scaling Q&A).



UML Diagram

5.Contribution

Tvisha played a pivotal role in designing the database schema and implementing the vector storage and retrieval architecture using FAISS. She also contributed to data scraping workflows (e.g., LinkedIn competitor insights) and ensured that the modular components (front-end, back-end, database) interacted seamlessly in the architecture design.

Harsh was responsible for developing the blogging module integrated with Dev.to and designing the KMS upload mechanism for documents. He also worked on defining the overall modular structure of the system (front-end, back-end, and APIs), ensuring scalability and maintainability in the architecture.

6. Conclusion

The modular system design of the KMS ensures separation of concerns, ease of maintenance, and flexibility in adding new features. The chosen technology stack balances efficiency, reliability, and stakeholder needs. Scalability strategies future-proof the system, ensuring it can support growing user bases, large datasets, and complex AI-powered functionality. This design establishes the foundation for a sustainable and impactful ICT solution.