TVISHA GAMI - 92310133014

HARSH SHANGHAVI - 92310133009

GUIDED BY

CHANDRASINH PARMAR

CAPSTONE PROJECT

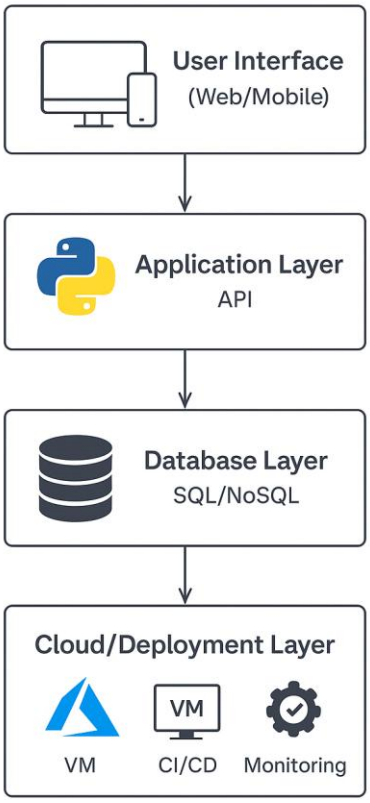
KNOWLEDGE MANAGEMENT SYSTEM

Documentation and Reporting

1. Technical Report

This technical report provides a comprehensive summary of the capstone project, detailing the system's design principles, implementation highlights, and key outcomes. The project was conceived to address the growing need for intelligent knowledge management, competitor insights, and operational efficiency in modern ICT-driven organizations. By leveraging cutting-edge technologies in artificial intelligence, cloud computing, and secure system design, the project demonstrates how an integrated platform can meet both organizational and stakeholder needs effectively.

- A fully functional knowledge management system accessible to employees and HR for centralized project learning and information sharing.
- An AI-powered chatbot capable of performing semantic Q&A over project documents, significantly reducing onboarding time for new employees and improving knowledge accessibility.
- A real-time LinkedIn competitor insights dashboard, empowering HR and management with actionable data-driven strategies.
- A blogging module integrated with Dev.to**,** ensuring smooth dissemination of organizational knowledge and experiences with the wider community.
- A robust monitoring and maintenance framework, ensuring long-term reliability, performance, and security of the system.



2. User Manual

1. Fully Functional Knowledge Management System

The platform provides a centralized repository for all project-related documents, reports, and insights, enabling employees and HR personnel to access critical information on demand. By leveraging advanced document ingestion pipelines, the system automatically indexes and categorizes files, ensuring that no knowledge is lost across projects. Employees can retrieve past project learnings, best practices, and technical references within seconds, fostering a culture of continuous learning and minimizing repetitive work. This also enhances organizational memory, allowing even new team members to quickly gain context about ongoing or completed projects.

2. Model-Powered Chatbot for Semantic Q&A

At the core of the system is an AI-driven chatbot powered by FAISS embeddings and large language models (LLMs), capable of understanding natural language queries and retrieving precise, context-aware answers. Unlike traditional keyword search engines, this chatbot interprets user intent and delivers semantically relevant information, which dramatically reduces time spent searching through lengthy documents. For HR and new employees, the chatbot acts as a virtual mentor, answering onboarding queries, explaining project workflows, and providing instant access to project-specific details. This innovation reduces training costs, accelerates employee productivity, and improves overall organizational efficiency.

3. Real-Time LinkedIn Competitor Insights Dashboard

The system integrates with LinkedIn data sources and APIs to provide a live dashboard that tracks competitor hiring trends, skill demands, and organizational movements. This feature empowers HR managers and decision-makers with data-driven workforce insights, helping them align recruitment strategies with industry benchmarks. For instance, if a competitor is rapidly hiring for AI/ML roles, the dashboard can signal management to anticipate market shifts and adjust talent acquisition plans accordingly. By transforming external workforce data into actionable intelligence, the dashboard ensures that the organization remains competitive in dynamic ICT landscapes.

4. Blogging Module Integrated with Dev.to

Knowledge dissemination extends beyond the organization through an integrated blogging feature. By connecting the system to the Dev.to publishing API, employees and managers can transform internal reports, incident learnings, or research findings into public-facing blogs with minimal effort. This creates a two-fold impact: internally, it encourages a culture of documentation and technical writing; externally, it enhances the company's visibility in developer communities, fostering credibility and potential collaborations. Automated formatting and publishing workflows ensure that technical content is consistently professional, accessible, and aligned with industry standards

5. Robust Monitoring and Maintenance Framework

To ensure long-term reliability, the system incorporates Azure Monitor and Application Insights, which continuously track system health, response times, error rates, and user activity. These metrics are displayed on real-time dashboards, giving system administrators early warning signals of potential issues before they escalate. In addition, the maintenance framework includes weekly backups, monthly security audits, and regular library dependency updates**,** protecting the system from downtime, data loss, or cybersecurity vulnerabilities. By embedding these practices into operations, the project guarantees a sustainable lifecycle, ensuring that the system remains secure, scalable, and performant under evolving business needs.

4. Code Documentation

The codebase follows best practices in documentation with inline comments and Python docstrings. Key modules include:

- **main.py:** Streamlit UI and navigation logic.
- **blog.py:** Blog generate and post to dev.io.
- **utils.py:** Helper functions for preprocessing, embedding, and session management.
- **faiss.index & index.pkl**: vectors storage and db
- **embedding_service.py:** Handles FAISS vector embeddings and retrieval.
- **linkedin_automation.py**: Automates LinkedIn competitor scraping with Selenium and many more files

  Dependencies:

  - Python 3.12 - Streamlit - FAISS - Selenium - Azure SDK - Ollama LLM - PostgreSQL and many more.

Overview of Codebase

The codebase of the Knowledge Management System (KMS) is structured into modular components for ease of maintenance, scalability, and extensibility. Each module is responsible for a specific functionality, ensuring a clean separation of concerns. The key modules are summarized below:

Upload & Embed Module

- Handles uploading of project documents (PDF, DOCX, TXT).
- Extracts text content and generates embeddings using FAISS**.**
- Stores embeddings in the vector database for semantic retrieval.

Question-Answering (Chatbot) Module

- Accepts user queries from employees/HR.
- Performs semantic similarity search on embeddings.
- Returns context-aware answers using an AI LLM model**.**

Blogging & Publishing Module

- Integrates with Dev.to API to automate posting of blogs.
- Formats project insights and publishes them externally.
- Provides a user-friendly UI for blog creation and scheduling.

LinkedIn Competitor Insights Module

- Uses Selenium automation to scrape competitor job postings and trends.
- Generates a real-time dashboard for HR and management.

Monitoring & Maintenance Module

- Integrated with Azure Monitor and Application Insights**.**
- Tracks KPIs such as API response time, uptime, and error logs.
- Automates weekly backups and monthly security checks.

Deployment & CI/CD Module

- Uses GitHub Actions pipelines for continuous integration.
- Deploys to Azure (VM + DNS + SSL)
- Ensures smooth upgrades and rollback options.

Inline Code Documentation Example (Python Snippet)

```
# Importing required librariesimport faissfrom sentence_transformers import SentenceTransformer

# Initialize embedding model

model = SentenceTransformer('all-MiniLM-L6-v2')

def generate_embeddings(texts):

    Generates vector embeddings for a list of text documents.

    Args:
```

```
        texts (list): List of strings (documents or paragraphs).
    Returns:
        ndarray: Vector embeddings for the input texts.
    return model.encode(texts)
# Create FAISS index for semantic search
dimension = 384  # embedding size for 'all-MiniLM-L6-v2'
index = faiss.IndexFlatL2(dimension)
def add_to_index(embeddings):
    Adds embeddings to the FAISS vector index.
    Args:
        embeddings (ndarray): Encoded embeddings to be indexed.
    index.add(embeddings)
def search_query(query, top_k=5):
    Performs semantic search over indexed embeddings.
    Args:
        query (str): The search query entered by the user.
        top_k (int): Number of top results to retrieve.
    Returns:
        List of indices representing the most relevant documents.
    query_vector = model.encode([query])
    distances, indices = index.search(query_vector, top_k)
    return indices
```

4. Contribution Statement

Tvisha was primarily responsible for database design, vectorization, and retrieval mechanisms**.** She prepared and optimized the project database, ensuring efficient schema design for scalability and robustness. She also implemented the vector conversion pipelines using FAISS embeddings, enabling semantic document retrieval with high accuracy. Additionally, Tvisha handled the data scraping tasks, collecting and preprocessing external datasets (e.g., competitor insights, domain-specific knowledge) for integration into the system. Harsh was responsible for designing and developing the blogging module and the KMS upload mechanism**.** He built the seamless integration with Dev.to's publishing API**,** ensuring automated content formatting and external dissemination of organizational knowledge. For the knowledge management system, Harsh implemented the document upload and ingestion workflows**,** allowing employees to upload PDFs, DOCX, and TXT files directly into the system for embedding and indexing. Additionally, he played a key role in the deployment process on Microsoft Azure**,** setting up CI/CD pipelines and ensuring that the system was accessible beyond localhost.