# Software Requirement Specifications Document

**Project Title:-** COLLEGE AI

**Team:-** SPARROWSPIDY & TEAM

**Project Instructor:-** DR. DHAKSHAYANI J

# Table Of Content

## 1. Introduction:-

This Software Requirements Specification (SRS) document describes the requirements of the AI Assistant of IIITK or simply College-AI system. College-AI is an **AI-powered assistant** designed to answer queries strictly related to IIIT Kottayam using **Retrieval-Augmented Generation (RAG).**

### *1.1 Purpose:-*

College-AI is an intelligent, domain-specific AI assistant designed exclusively for **IIIT Kottayam**.
It provides accurate, contextual, and up-to-date information about academics, campus life, rules, procedures, and student guidance by leveraging **open-source language models**, **vector databases**, and **secure backend architecture**.

### *1.2 Scope:-*

The scope of the **College-AI: AI-Powered College Information Assistant for IIIT Kottayam** project is to design and develop a **college-specific intelligent question-answering system** that provides **accurate, reliable, and document-grounded responses** to user queries related exclusively to **IIIT Kottayam**.

The system focuses on delivering verified information by leveraging **Retrieval-Augmented Generation (RAG)** and **open-source Large Language Models (LLMs)** running locally through **Ollama**, thereby avoiding dependency on paid or external APIs.

### *1.3 Overview:-*

College-AI is an AI-powered, college-specific assistant designed to provide accurate and reliable information related to IIIT Kottayam. The system uses open-source Large Language Models (LLMs) running locally through Ollama and follows a Retrieval-Augmented Generation (RAG) approach to ensure that responses are strictly based on official institutional documents, thereby reducing misinformation and hallucinations.

The project aims to assist students and peers by offering quick access to academic, administrative, and campus-related information through a chat-based interface. By leveraging vector embeddings, FAISS-based similarity search, and a modular backend architecture, College-AI ensures efficiency, privacy, and scalability while remaining completely free from paid APIs.

## 2. Overall Description:-

### *2.1 Product Perspective:-*

College-AI is a standalone web-based AI system that integrates document retrieval and language model generation. It operates without relying on paid APIs.

### *2.2 Product Functions:-*

• Answer IIIT Kottayam-specific queries
• Retrieve relevant document chunks
• Generate grounded AI responses
• Ensure security and reliability

### *2.3 User Classes and Characteristics:-*

Students: Seek academic and administrative information
Developers: Maintain and enhance the system
Administrators (future): Manage documents

### *2.4 Operating Environment:-*

Operating System: Windows/Linux
Backend: Python 3.10+, FastAPI
LLM Runtime: Ollama
Vector Store: FAISS

### *2.5 Design and Implementation Constraints:-*

• Use only open-source tools
• No paid APIs
• Offline-first LLM execution

### *2.6 Assumptions and Dependencies:-*

• Ollama must be installed
• Institutional data is accurate
• Users have network access

## 3. System Features and Functional Requirements:-

### *3.1 Document Ingestion:-*

- The system shall allow uploading of official documents.
- The system shall extract and chunk text from PDFs.

### *3.2 Embedding Generation:-*

The system shall convert text chunks into vector embeddings using transformer models.

### *3.3 Vector Storage and Retrieval:-*

- The system shall store embeddings in FAISS.
- The system shall retrieve top-k relevant chunks for each query.

### *3.4 Query Processing:-*

- The system shall accept natural language queries.
- The system shall preprocess queries for retrieval.

### *3.5 Response Generation:-*

- The system shall generate responses using a local LLM.
- The system shall restrict output to retrieved context.

## 4. Non-Functional Requirements:-

### *4.1 Performance:-*

- The system shall provide responses within a reasonable time frame.
- Efficient vector search and optimized LLM inference shall be used to minimize latency.

### *4.2 Security:-*

- The system shall ensure that internal college documents are not exposed to unauthorized access.
- The backend APIs shall be protected against misuse and excessive requests.

### *4.3 Reliability:-*

- The system shall operate consistently without unexpected failures.
- The system shall handle invalid inputs and runtime errors gracefully.

### *4.4 Maintainability:-*

- The user interface shall be simple, intuitive, and easy to navigate.
- The system shall require minimal user training.

### *4.5 Portability:-*

- The system shall be deployable across different operating systems such as Windows and Linux.
- The system shall support different environments with minimal configuration changes.

### *4.6 Data Integrity:-*

- The system shall ensure accuracy and consistency of stored documents and embeddings.
- The system shall prevent accidental data loss during updates.

### *4.7 Availability:-*

- The system shall be available for use during normal operational hours.
- Planned maintenance shall not significantly disrupt system usage.

### *4.8 Usability:-*

- The user interface shall be simple, intuitive, and easy to navigate.
- The system shall require minimal user training.

### *4.9 Scalability:-*

- The system shall support the addition of new documents and knowledge sources without major architectural changes.
- The vector storage mechanism shall scale efficiently as data volume increases.

## 5. External Interface Requirements:-

### *5.1 User Interface:-*

Chat-based web interface.

### *5.2 Software Interfaces:-*

FastAPI for backend
Ollama for LLM inference
FAISS for vector similarity search

## 6. User Interface Requirements:-

### *6.1 General Interface Design:-*

- The system shall provide a **web-based, chat-style user interface** similar to common conversational AI platforms.
- The interface shall be **simple, intuitive, and user-friendly** to ensure ease of use for students and new users.
- The design shall follow a **minimalistic layout** with clear visual separation between user queries and system responses.

### *6.2 Input Interface:-*

- The system shall allow users to **enter queries using a text input field**.
- The input field shall support **natural language queries** related to IIIT Kottayam.
- The interface shall include a **send button** and also support **keyboard-based submission (Enter key)**.

### *6.3 Output Interface:-*

- The system shall display responses in a **chat bubble format**.
- AI responses shall be **clearly distinguishable** from user messages.
- The interface shall support **formatted text** (paragraphs, bullet points) for better readability.
- The system may display a **loading indicator** while generating responses.

### *6.4 Feedback and Error Handling:-*

- The interface shall display **informative error messages** in case of:
    - Invalid queries
    - System unavailability
    - Internal processing errors
- The system shall notify users when a query is **outside the supported scope**.

### *6.5 Responsiveness and Accessibility:-*

- The user interface shall be **responsive**, functioning correctly on desktops and laptops.
- The interface shall maintain usability across different screen resolutions.
- Basic accessibility practices such as readable font sizes and contrast shall be followed.

### *6.6 Security and Privacy Indicators:-*

- The interface shall not request sensitive personal information from users.
- The system shall clearly indicate that responses are **based on official IIIT Kottayam documents**.

## 7. Performance Requirements:-

### *7.1 Response Time:-*

- The system shall generate responses to user queries within an acceptable time frame.
- For standard queries, the system should respond within **3–10 seconds**, depending on model size and hardware capability.

### *7.2 Throughput:-*

- The system shall be capable of handling **multiple user requests sequentially** without failure.
- The backend shall efficiently process concurrent requests within resource limitations.

### *7.3 Scalability:-*

- The system shall support an increase in the number of documents and embeddings without significant degradation in performance.
- The vector retrieval mechanism shall efficiently handle large embedding datasets using FAISS.

### *7.4 Resource Utilization:-*

- The system shall optimize CPU and memory usage during embedding generation and inference.
- The system shall operate effectively on systems with **limited hardware resources**, as it uses locally hosted LLMs.

### *7.5 Availability:-*

- The system shall remain operational during normal usage hours.
- Temporary unavailability due to maintenance or model updates shall not cause data loss.

### *7.6 Reliability Under Load:-*

- The system shall continue functioning correctly under moderate load conditions.
- The system shall handle invalid or malformed queries without crashing.

## 8. Schedule and Budget:-

### *8.1 Project Schedule:-*

The development of the **College-AI** project is planned over a period of **8–6 months**, following a structured and phased approach. Each phase focuses on specific deliverables to ensure systematic development and timely completion.

### *8.2 Project Timeline:-*

| Phase | Activity | Duration |
|---|---|---|
| Phase 1 | Requirement Analysis & SRS Preparation | 2 Weeks |
| Phase 2 | System Design & Architecture Planning | 2 Weeks |
| Phase 3 | Data Collection & Document Preparation | 3 Weeks |
| Phase 4 | RAG Pipeline Development (Embeddings + Retrieval) | 4 Weeks |
| Phase 5 | LLM Integration using Ollama | 3 Weeks |
| Phase 6 | Backend API Development (FastAPI) | 3 Weeks |
| Phase 7 | Frontend Development | 3 Weeks |
| Phase 8 | Security & Performance Enhancements | 2 Weeks |
| Phase 9 | Testing & Validation | 2 Weeks |
| Phase 10 | Documentation & Final Submission | 2 Weeks |

**Total Estimated Duration: 6–8 Months**

### *8.3 Project Budget:-*

The **College-AI** project is designed to be **cost-effective** by relying entirely on **open-source software and free tools**, making it suitable for academic and student-led development.

| Component | Cost |
|---|---|
| Open-source LLMs (via Ollama) | ₹0 |
| Python Libraries (FAISS, LangChain, FastAPI) | ₹0 |
| Development Tools (VS Code, GitHub) | ₹0 |
| Operating System | ₹0 |
| Hardware (Personal Laptops) | Already available |
| Internet Usage | Minimal / Existing |
| Cloud Services | Not used |

## 9. Appendices:-

### *Appendix A: Glossary:-*

| Term | Description |
|---|---|
| LLM | Large Language Model used for generating natural language responses |
| RAG | Retrieval-Augmented Generation technique that combines document retrieval with text generation |
| FAISS | Facebook AI Similarity Search, used for efficient vector similarity search |
| Vector Database | A storage system for vector embeddings used in semantic search |
| Embeddings | Numerical representations of text used for similarity comparison |
| Ollama | Local runtime for executing open-source LLMs |
| FastAPI | Python web framework for building RESTful APIs |

### *Appendix B: Technology Stack:-*

- **Programming Language:** Python 3.10+
- **Backend Framework:** FastAPI
- **AI Models:** Open-source LLMs (Mistral, LLaMA-based models)
- **Embedding Model:** Sentence Transformers
- **Vector Search Engine:** FAISS
- **LLM Runtime:** Ollama
- **Version Control:** Git, GitHub
- **Dependency Management:** UV

### *Appendix C: System Requirements:-*

**Hardware Requirements:-**

- Minimum 8 GB RAM
- Multi-core CPU
- 20 GB free disk space

**Software Requirements:-**

- Windows/Linux OS
- Python 3.10 or higher
- Ollama installed
- Web browser

### *Appendix D: Assumptions:-*

- Official IIIT Kottayam documents are accurate and up to date.
- Users will query only within the supported scope.

- Local system resources are sufficient for model execution.

### *Appendix E: Future Enhancements:-*

- Role-based authentication
- Multilingual support
- Mobile application
- Admin dashboard

## 10. Conclusion:-

The **College-AI** project successfully demonstrates the application of modern Artificial Intelligence techniques to solve real-world problems in an academic environment. By leveraging **Retrieval-Augmented Generation (RAG)** and **open-source Large Language Models**, the system provides accurate, reliable, and college-specific information while minimizing misinformation and dependency on external paid services.

The project emphasizes **privacy, scalability, and cost-effectiveness** by executing AI models locally using Ollama and organizing institutional data through efficient vector-based retrieval. With its modular design and clear architectural approach, College-AI lays a strong foundation for future enhancements such as authentication, multilingual support, and integration with official institutional systems. Overall, this project reflects a practical and sustainable approach to deploying AI solutions within educational institutions and highlights the potential of open-source AI in building intelligent, trustworthy systems.

## 11. Contributors:-


1. Dr. Dhakshayani J ( Project Supervisor )
   dhakshayani@iiitkottayam.ac.in

2. Vivek Ediga
   vivek772256l@gmail.com

3. Himasai Vihar
   himasaisuraboina24bcs0225@iiitkottayam.ac.in

4. Vinay V
   secretlongplan@gmail.com

5. Adithya M
   madapatijyothiradithyamadapati@gmail.com

6. Joyal Kumar J
   janampallyjoyalkumar99@gmail.com

7. Akilesh K
   akhileshkontheti@gmail.com